Bounded Degree Interval Sandwich Problems*

Haim Kaplan† Ron Shamir‡

Abstract

The problems of Interval Sandwich (IS) and Intervalizing Colored Graphs (ICG) have received a lot of attention recently, due to their applicability to DNA physical mapping problems with ambiguous data. Most of the results obtained so far on the problems were hardness results. Here we study the problems under assumptions of sparseness, which hold in the biological context. We prove that both problems are polynomial when either (1) the input graph degree and the solution graph clique size are bounded, or (2) the solution graph degree is bounded. In particular, this implies that ICG is polynomial on bounded degree graphs for every fixed number of colors, in contrast with the recent result of Bodlaender and de Fluiter.

1 Introduction

A graph is an *interval graph* if one can assign an interval on the real line to each vertex so that two vertices are adjacent if and only if their intervals have a nonempty intersection. Consider the following problem:

> INTERVAL SANDWICH (IS):
> **INSTANCE:** A triple $S = (V, E, F)$, where V is a set of vertices, and E and F are disjoint sets of edges on V.
> **QUESTION:** Decide whether there exists an interval graph $G' = (V, E')$ so that $E \subseteq E'$ and $E' \cap F = \emptyset$.

E and F are called the sets of *mandatory* and *forbidden* edges, respectively. The graph G', if it exists, is called a *sandwich graph* for the instance S. Without loss of generality,

*A Preliminary version of this work was presented at the 4th Israeli Symposium on Theory of Computing and Systems (ISTCS'96) [14].

†AT&T-labs research, 180 Park Ave, Florham Park, NJ 07932 USA. hkl@research.att.com

‡Department of Computer Science, Sackler Faculty of Exact Sciences, Tel Aviv University, Tel-Aviv 69978 ISRAEL. Research supported in part by the Minstry of Science and the Arts, Israel, Grant No. 4911294. shamir@math.tau.ac.il

$G = (V, E)$ (which we call the *mandatory graph*, or simply the *input graph*) is assumed to be connected.

The interval sandwich problem was introduced by Golumbic and Shamir [10] in the context of temporal reasoning, and was shown to be NP-complete. See also [9] for a simpler proof. The following special case of the problem, which is motivated by molecular biology, was studied in [6] and [9]: Recall that a *coloring* of a graph $G = (V, E)$ is a function $c : V \rightarrow \{1, \ldots, k\}$ such that for every edge $(x, y) \in E$, $c(x) \neq c(y)$.

> INTERVALIZING COLORED GRAPHS (ICG):
> **INSTANCE:** A graph $G = (V, E)$ and a coloring c of it.
> **QUESTION:** Is G a subgraph of an interval graph G' that is properly colored by c?

Clearly ICG is a restriction of IS where $F = \{(i, j)|c(i) = c(j)\}$. ICG was shown to be NP-complete independently in [6] and [9]. Parameterized hardness results for the problem, where the parameter is the number of colors, were given in [6] and strengthened in [3]. Recently, Bodlaender and de Fluiter [2] strengthened these results further by showing that ICG is NP-complete even when the number of colors k is fixed to any constant $k \geq 4$. On the other hand they showed that ICG is polynomial for $k \leq 3$, by giving a rather involved algorithm and proof for three colors.

In this note we show that two parameterized versions of IS (and hence also of ICG) are tractable. Both problems deal with bounded degree instances. The first problem asks for the existence of an interval sandwich graph with maximum clique size no greater than k in a sandwich instance in which the degree of any vertex in the input graph G is no greater than d. For fixed k and d we give an $O(n^{k-1})$ algorithm for the problem. Hence, if the degree is bounded, IS is polynomial whenever the clique size is bounded. In particular this implies that ICG is polynomial *for every fixed number of colors*, on bounded-degree input graphs. This is in contrast to the recent negative result of [2].

The second problem asks for the existence of an interval sandwich graph in which the degree of any vertex is no greater than d. For fixed d we give a $O(n^{d-1})$ algorithm for that problem. Again, this implies that ICG is polynomial when the interval graph must have bounded degree.

One of the motivations to studying IS and ICG is constructing physical DNA maps. In physical mapping in molecular biology, several copies of a long chain of DNA are cut, and fragments of the chains are extracted. Each fragment is a contiguous chain of DNA, called a *clone*. The order of the clones is lost, and the goal is to reconstruct that order, based on information on pairwise overlaps of clones. Biological experiments allow the determination whether two clones overlap (have a nonempty intersection). The experimental data of clone-clone overlap can be presented as a graph G whose vertices corresponding to clones, overlaps corresponding to edges in E, and non-overlaps corresponding to edges in F. As is often the case in practice, for some pairs of clones, the information whether they overlap or not is unknown, due to unperformed or inconclusive experiments. In this situation, deciding if the data is consistent with some clone order is equivalent to the IS problem. If clones can be partitioned into sets where in each set all clones originate from a single copy of the chromosome, then each such set forms an independent set in the graph, which can be assigned a single color. In that case the decision problem is equivalent to ICG. See [4, 19, 1, 9, 7] for more on the biology and computational aspects of this problem.

Our motivation to studying bounded degree problems is from physical mapping, although similar sparse problems arise naturally in many other applications. We have observed in [13] that most biological maps are very sparse: The size of the largest set of mutually overlapping clones in such maps is typically between 5 and 15, even when the maps contain thousands of clones [16, 17]. The question is whether by restricting the problems to satisfy this property, their complexity will be polynomial. In [13], sparsity was modeled by bounding the clique size of the sandwich graph, for the case of equal length clones. It was shown that the unit interval sandwich problem is polynomial whenever the clique size in the resulting realization is bounded by a constant [13, 15]. Here we deal with clones of arbitrary lengths. The problem discussed in Section 3.1 models sparsity by bounding the maximum clique size of the sandwich graph, and additionally, by bounding the maximum degree in the input graph. The first requirement means that the largest set of mutually overlapping clones in the solution map should be bounded. The second means that at the input, the maximum number of clones known to intersect a particular one is bounded. The problem discussed in Section 3.2 models sparsity by bounding the maximum degree of the required sandwich graph. In biological terms it means bounding the number of clones intersecting any single clone in the map we seek. The same modeling of sparsity was exploited in [11] for studying other mapping problems. The algorithms used here build on a dynamic programming scheme, motivated by ideas from [18] and [12].

For basic facts and terminology on graph theory and interval graphs, see [8]. For a description of parameterized complexity theory, see [5].

2 The basic scheme

In this section we describe an algorithm for the Interval Sandwich problem. The algorithm is exponential in the worst case but likely to behave better on practical instances with many forbidden edges. In the next section we show how to derive from this general algorithm polynomial algorithms for the two restrictions of IS defined in the previous section.

The basic idea is very simple: Generate solutions for subproblems induced on increasing number of vertices. A method of extending a solution for a subproblem by adding a single vertex is given. The key to the relative efficiency of our algorithm is that each subproblem even though may have many different solutions (interval sandwich graphs) is handled only once.

Let $S = (V, E, F)$ be a sandwich instance. A *cut* of S is a subset $X \subseteq V$. The *edge set*

$\delta(X)$ of a cut X consists of all edges in E that have one endpoint in X and the other in $V-X$. The *active region* of X, $\Delta(X)$, is a subset of X consisting of all the vertices incident with at least one edge in $\delta(X)$. A *layout* $L(X)$ of a cut X is a function that assigns an interval $I(v)$ for every $v \in X$ such that the following three conditions hold.

1. $I(v) \cap I(w) \neq \emptyset$ if $(v, w) \in E$.
2. $I(v) \cap I(w) = \emptyset$ if $(v, w) \in F$.
3. If $v \in \Delta(X)$ then $r(I(v)) = \max\{r(I(w))|w \in X\}$ where $r(I(u))$ denotes the right endpoint of the interval assigned to u.

We shall say that a cut is *feasible* if it has a layout.

We say that a feasible cut Y *extends* a feasible cut $X \neq Y$ if $Y = X \cup \{v\}$ and one can obtain a layout for Y from a layout $L(X)$ of X by adding $I(v)$ to the right of all the intervals in $L(X)$ and changing the right endpoint of the intervals of the vertices in $\Delta(X)$ to be $r(I(v))$. Figure 2 shows a sandwich instance together with a sequence of feasible cuts $X_1, \ldots, X_6$ such that X_i extends X_{i-1} for every $2 \leq i \leq 6$. The following simple lemma gives an alternative characterization of the extension relation between cuts. We omit its proof which is straightforward.

Lemma 2.1 *Let X be a feasible cut and let $Y = X \cup \{v\}$. Y is a feasible cut and Y extends X if and only if $(v, w) \notin F$ for every $w \in \Delta(X)$.* ∎

It follows that if a cut Y extends a cut X a layout for Y can be obtained from any layout of X by the method described above. Two other simple observations about the extension relation between cuts are in order at this point. We prove the first, and omit the proof of the second which is straightforward.

Observation 2.2 *Every feasible cut extends at least one smaller feasible cut.*

Proof. let Y be a feasible cut, L a layout of Y, and v a vertex in Y such that $l(I(v))$, the left endpoint of $I(v)$, is maximum among all the left endpoints of intervals in L. If we change the right endpoint of any interval in L that crosses $l(I(v))$ to be $l(I(v))$ and delete $I(v)$ we obtain a layout for the cut $Y - \{v\}$ that Y extends. ∎

Observation 2.3 *There exists a solution to the instance $S = (V, E, F)$ if and only if V is a feasible cut.* ∎

These observations suggest the algorithm for the interval sandwich problem outlined in Figure 2. The algorithm essentially generates all feasible cuts vin increasing cardinality order as follows. It maintains a queue Q of feasible cuts yet to be extends. In each iteration it pops a cut X from Q and generates all feasible cuts that extend X. Each extension that is detected for the first time is injected into Q. There exists a solution to the given instance if and only if the cut V is generated before Q gets empty. We assume that the algorithm maintains a table storing every cut already detected whose size matches the current cardinality. This is in order to decide for each cut generated whether it is already

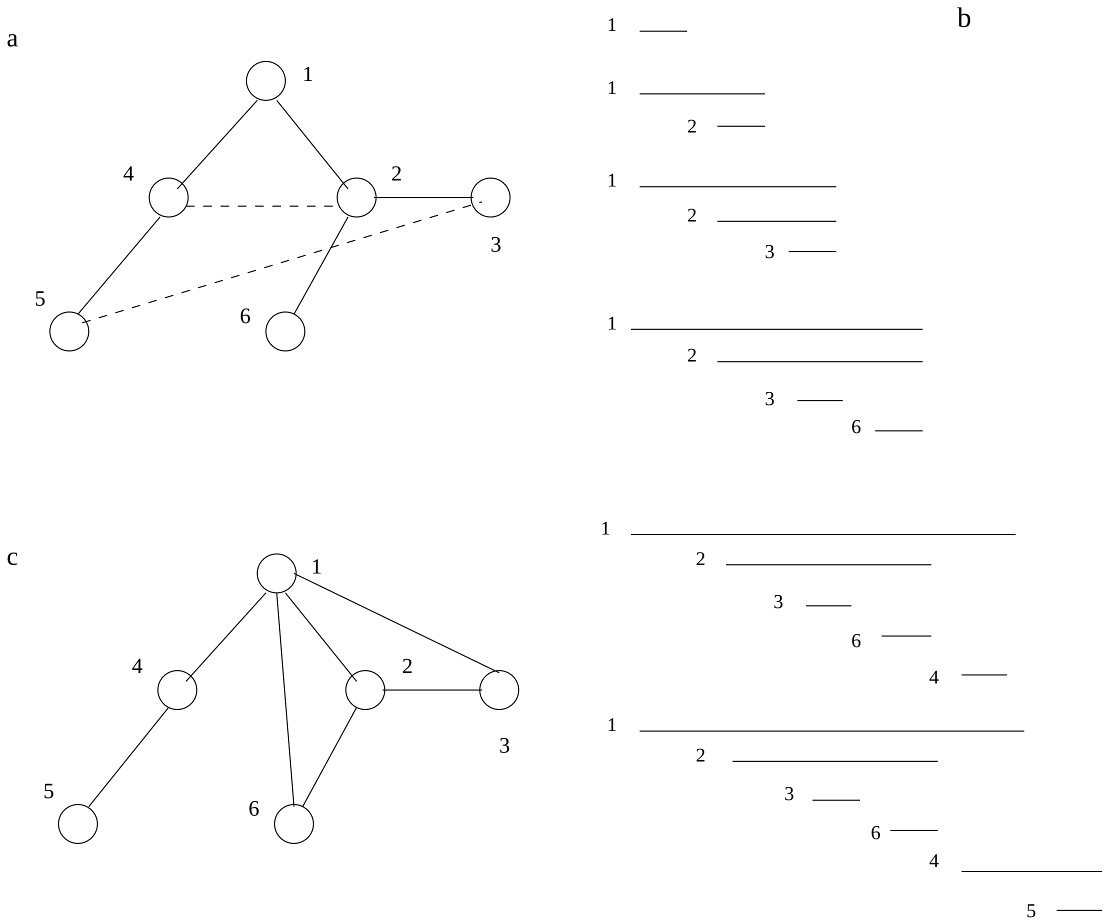

Figure 1: a) A sandwich instance. Solid edges are mandatory. Broken edges are forbidden. b) A sequence of feasible cuts $X_1 = \{1\}$, $X_2 = \{1,2\}$, $X_3 = \{1,2,3\}$, $X_4 = \{1,2,3,6\}$, $X_5 = \{1,2,3,4,6\}$, $X_6 = \{1,2,3,4,5,6\}$, where X_i extends X_{i-1}. For X_i we show a layout obtained from a layout of X_{i-1} as specified by the definition of extension. c) The sandwich graph corresponding to the final layout.

in Q or not in constant time. We also assume that for any pair of vertices the existence of a forbidden edge between them can be checked in constant time.

Under the assumptions above, the running time of the algorithm we describe is proportional to the number of feasible cuts, which is exponential in the worst case but likely to be much smaller in practice. In the next section we show how to adapt the scheme to be polynomial for the problems we have defined in Section 2.

algorithm SANDWICH(S);

/* $S = (V, E, F)$ is a connected sandwich instance */
/* initialization */
1. initialize an empty queue Q.
2. **for** each $v \in V$, add the cut $\{v\}$ to Q.
begin
    while $Q \neq \emptyset$ **do** :
    /* iteration */
    begin
        3. Remove some feasible cut X from Q.
        4. **for** every $z \notin X$ **do** :
        begin
            /* extension attempt */
            5. Try to extend X by z (using Lemma 2.1)
            6. Suppose a feasible cut Y is generated.
            if Y has domain V **then** output "success" and stop.
            Otherwise, **if** Y is new, **then** add it to Q.
        end
    end
    7. Output "failure" and stop.
end

Figure 2: A schematic algorithm for the IS problem

3 Polynomial problems

In this section we present two polynomial restrictions of the general Interval Sandwich problem. To obtain a polynomial algorithm for each of these problems we restrict the set of feasible cuts to be of polynomial size and generate them using a version of algorithm SANDWICH in Figure 2.

3.1 Bounded degree input graph, bounded width

Recall that in this version of the problem we are given a sandwich instance where the degrees of all vertices in the mandatory graph $G = (V, E)$ are bounded by d. An interval sandwich graph with maximum clique size at most k is required. In order to adapt the framework of section 1 such that it polynomially solves this restriction of IS we redefine a cut $X \subseteq V$ to be *feasible* if $|\Delta(X)| \leq k-1$ and it has a layout $L(X)$ satisfying the three conditions given in Section 1 and the following fourth condition.

4. Every $k+1$ intervals of $L(X)$ have an empty intersection.

It is straightforward to verify that Observations 2.2 and 2.3 still hold with the modified definition of a feasible cut. Lemma 2.1 needs to modified slightly as follows.

Lemma 3.1 *Let X be a feasible cut and let $Y = X \cup \{v\}$. Y is a feasible cut and Y extends X if and only if $(v, w) \notin F$ for every $w \in \Delta(X)$ and $|\Delta(Y)| \leq k-1$.* ∎

The algorithm in Figure 2 still applies as well. One change necessary is that while trying to extend a feasible cut X whose active region is of size $k-1$ we check only vertices that are adjacent to at least one vertex in the active region of X. Since the mandatory graph is connected, extending a cut whose active region is of size $k-1$ by a vertex that is not adjacent to the active region necessarily results in a cut with active region of size k and thus infeasible. Also, the decision whether an extension of a feasible cut X by z results in a feasible cut is done using Lemma 3.1.

Theorem 3.2 *For fixed k and d, algorithm SANDWICH can be implemented to detect whether an interval sandwich graph with clique size at most k exists in a degree-d sandwich instance in $O(n^{k-1})$ time using $O(n^{k-1})$ space.*

Proof. One can specify a feasible cut X by giving its active region $\Delta(X)$ and its edge set $\delta(X)$. The set X itself can then be recovered as follows. Delete the edges in $\delta(X)$ from $G = (V, E)$, the cut X is the union of the connected components of the resulting graph that have a nonempty intersection with $\Delta(X)$.

Since $|\Delta(X)| \leq k-1$ the number of possible active regions is $O(n^{k-1})$. Moreover, since the degrees of the vertices in $G = (V, E)$ are bounded by d the number of possible edge sets incident to each active region is at most $2^{d(k-1)}$. Thus, it follows that the total number of feasible cuts is $O(n^{k-1})$.

If X is a cut with an active region of maximum size $k-1$ then the only possible vertices that can extend X to a bigger feasible cut are those vertices in $V - X$ adjacent in E to vertices in X. The number of vertices of this kind is bounded by the maximum size of the edge set of X that is $d(k-1)$. For each vertex checking whether it extends X takes $O(k)$ time. Thus the total time spent on extending cuts whose regions are of size $k-1$ is $O(n^{k-1})$. A cut whose active region is smaller than $k-1$ may be extended by $\Omega(n)$ vertices but since the number of such cuts is $O(n^{k-2})$ the total time spent on extending them is also $O(n^{k-1})$ and the time bound stated in the theorem follows.

The space complexity of the algorithm is dominated by the table we maintain to determine whether a newly generated cut hasn't been seen before. We assume that the vertices are labeled $1, \ldots, n$ and the the edges incident with vertex v are labeled $1, \ldots, d(v)$. The table we maintain is indexed by a feasible cut specified using the pair $\Delta(X), \delta(X)$. The active region $\Delta(X)$ is represented as a sequence of $k-1$ numbers between 0 and n. The first $|\Delta(X)|$ numbers are the labels of the vertices in $\Delta(X)$ and the rest $k-1-|\Delta(X)|$ numbers are zeros. The edge set $\delta(X)$ is represented by a sequence of $k-1$ bit strings of length d each. The ith bit string has 1 in position j if and only if the jth edge incident to the ith vertex of $\Delta(X)$ (the order on $\Delta(X)$ is induced by the vertex labels) is in $\delta(X)$. Since for each cut one bit of information suffices the size of the table is equal to the the number of entries in it, that is $n^{k-1} * 2^{d(k-1)}$ bits ■

Corollary 3.3 *For every fixed number of colors, Intervalizing Colored Graphs is polynomial if the degrees in the input graph are bounded by a constant.*

3.2 Bounded degree

Recall that in this version of the problem an interval sandwich graph with maximum degree at most d is required. Hence we may assume that all the degrees in our mandatory graph $G = (V, E)$ are at most d. Note that an interval sandwich solution if exists is connected and hence it can contain a clique of size greater than d only if the whole graph is a clique with $d+1$ vertices. By checking for this special case separately we may assume that our interval sandwich solution does not contain cliques of size greater than d.

As in Section 3.1, in order to adapt the framework of section 1 such that it polynomially solves this restriction of IS we redefine a cut $X \subseteq V$ to be *feasible* if $|\Delta(X)| \leq d-1$ and it has a layout $L(X)$ satisfying the three conditions given in Section 1 and the following two additional conditions.

4. Each interval of $v \notin \Delta(X)$ intersects at most d other intervals.
5. If $v \in \Delta(X)$ and $dangling(v, X) = \{(v, w) | w \notin X\}$ then $I(v)$ intersects at most $d - |dangling(v, X)|$ other intervals.

Lemma 2.1 does not hold any more but it is easy to prove the following lemma instead.

Lemma 3.4 *Let X be a feasible cut and let $Y = X \cup \{v\}$. Y is a feasible cut and Y extends X if and only if $(v, w) \notin F$ for every $w \in \Delta(X)$, and X has a layout L such that for every $u \in \Delta(X)$, $(u, v) \notin E$, $I(u)$ intersects strictly less than $d - |dangling(u, X)|$ other intervals.* ■

It follows that Y may extend X while some layout of X may not be extendable to a layout of Y. For a layout $L(X)$ of X define $degree_{L(X)}(v)$ to be the number of intervals that $I(v)$ intersects. Lemma 3.4 implies that two different layouts of X with the same degree for every vertex in $\Delta(X)$ either can both be extended by a vertex v or they both cannot. Note that Observation 2.2 and 2.3 carry over to our new setup here.

To adapt the generic algorithm to solve our problem in this section we maintain in Q pairs, each consisting of a feasible cut X and a degree function defined on $\Delta(X)$ such that there exists a layout of X with that particular degree function. When expending a pair (X, f) then we generate every possible pair (Y, g) where Y extends X and a layout of Y with degree function g is obtained by extending a layout of X with degree function f. The following theorem proves a time bound on the modified algorithm

Theorem 3.5 *For fixed d, algorithm SANDWICH can be implemented to detect whether an interval sandwich graph with maximum degree d exists in a given sandwich instance in $O(n^{d-1})$ time using $O(n^{d-1})$ space.*

Proof. The same reasoning as in the proof of Theorem 3.2 shows that the number of feasible cuts that the algorithm has to consider is $O(n^{d-1})$. The number of possible degree functions whose domain is the active region of each cut is d^d and hence the total number of pairs that have to be extended is $O(n^{d-1})$. The number of pairs that potentially extend a cut with active region of size $d-1$ is independent of n so that the total time spend extending such pairs is $O(n^{d-1})$. The number of pairs that can extend a pair whose cut has an active region of size at most $d-2$ is $\Omega(n)$ but since the number of such pairs is $O(n^{d-2})$ the total time spent extending these pairs is also $O(n^{d-1})$ and the time bound of the theorem follows. The space follows by the same argument as for Theorem 3.2. ■

Corollary 3.6 *Intervalizing Colored Graphs is polynomial if the degrees in the interval graph G' are bounded by a constant.*

4 Conclusions

We have presented an algorithm that finds in $O(n^{k-1})$ time and space an interval sandwich graph whose clique size is at most k when the mandatory input graph is of bounded degree. We have also presented an $O(n^{d-1})$ time and space algorithm that finds an interval sandwich graph with maximum degree at most d if one exists.

Our worst case time guarantees are not practical even when k and d are quite small. An intriguing open question is whether faster algorithms for these problems exist. In particular, is there an algorithm for either one of these problems whose complexity is $O(n^{\alpha})$ where α is a fixed constant independent of k and d? In other words the dependency on k and d is only through multiplicative and additive constants. We conjecture that neither of these two problems has such an algorithm. To support this conjecture one may try to prove that these problems are hard for some level of the W-hierarchy [5]. Some of the ideas in the parameterized reduction of [13] may prove useful to obtain such a result.

The enumeration arguments given in the proofs of Theorems 3.2 and 3.5 for upper bounding the number of feasible cuts are crude and do not take into account many other constraints that are harder to analyze. It is very likely that for many problem instances the number of feasible cuts would be much smaller and hence the running time would be much faster.

References

[1] F. Alizadeh, R. M. Karp, L. W. Newberg, and D. K. Weisser. Physical mapping of chromosomes: A combinatorial problem in molecular biology. In *Proc. fourth annual ACM-SIAM Symp. on Discrete Algorithms (SODA 93)*, pages 371–381. ACM Press, 1993.

[2] H. L. Bodlaender and B. L. E. de Fluiter. Intervalizing k-colored graphs. In *Proc. ICALP 95*. LNCS, 1995.

[3] H. L. Bodlaender, M. R. Fellows, and M. T. Hallet. Beyond NP-Completeness for problems of bounded width: Hardness for the W hierarchy (extended abstract). In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 449–458. ACM Press, New York, 1994.

[4] A. V. Carrano. Establishing the order of human chromosome-specific DNA fragments. In A. D. Woodhead and B. J. Barnhart, editors, *Biotechnology and the Human Genome*, pages 37–50. Plenum Press, New York, 1988.

[5] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer Verlag, 1997.

[6] M. R. Fellows, M. T. Hallet, and H. T. Wareham. DNA physical mapping: Three ways difficult. In *Proc. First European Symp. on Algorithms (ESA '93)*, pages 157–168. Springer, 1993. LNCS 726.

[7] P. W. Goldberg, M. C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. *Journal of Computational Biology*, 2(1):139–152, 1995.

[8] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[9] M. C. Golumbic, H. Kaplan, and R. Shamir. On the complexity of DNA physical mapping. *Advances in Applied Mathematics*, 15:251–261, 1994.

[10] M. C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *J. ACM*, 40:1108–1133, 1993.

[11] D. Greenberg and S. Istrail. The chimeric mapping problem: algorithmic strategies and performance evaluation on synthetic genomic data. *Computers Chem.*, 18(3):207–220, 1994.

[12] E. Gurari and I. H. Sudborough. Improved dynamic programming algorithms for the bandwidth minimization and the mincut linear arrangement problem. *J. Algorithms*, 5:531–546, 1984.

[13] H. Kaplan and R. Shamir. Pathwidth, bandwidth and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing*, 25(3):540–561, 1996.

[14] H. Kaplan and R. Shamir. Physical maps and interval sandwich problems: Bounded degrees help. In *Proc. ISTCS*, pages 195–201, 1996.

[15] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping. In *Proceedings of the 35th Symposium on Foundations of Computer Science*, pages 780–791. IEEE Computer Science Press, Los Alamitos, California, 1994.

[16] Y. Kohara, K. Akiyama, and K. Isono. The physical map of the whole *E. coli* chromosome: application of a new strategy for rapid analysis and sorting of large genomic libraries. *Cell*, 50:495–508, 1987.

[17] M. V. Olson et al. Random-clone strategy for genomic restriction mapping in yeast. *Proc. Nat. Acad. Sci. USA*, 83:7826–7830, 1986.

[18] J. B. Saxe. Dynamic programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM J. Algebraic Discrete Methods*, 1:363–369, 1980.

[19] J.D. Watson, M. Gilman, J. Witkowski, and M. Zoller. *Recombinant DNA*. W.H. Freeman, New York, 2nd edition, 1992.