Scheduling Split Intervals *

Reuven Bar-Yehuda[†‡] Magnús M. Halldórsson[§] Joseph (Seffi) Naor[†]
Hadas Shachnai[†] Irina Shapira[†]

July 5, 2005

Abstract

We consider the problem of scheduling jobs that are given as *groups* of non-intersecting segments on the real line. Each job J_j is associated with an interval, I_j, which consists of up to t segments, for some $t \geq 1$, and a weight (profit), w_j; two jobs are in conflict if their intervals intersect. Such jobs show up in a wide range of applications, including the transmission of continuous-media data, allocation of linear resources (e.g. bandwidth in linear processor arrays), and in computational biology/geometry. The objective is to schedule a subset of non-conflicting jobs of maximum total weight.

Our problem can be formulated as the problem of finding a *maximum weight independent set* in a *t-interval* graph (the special case of $t = 1$ is an ordinary interval graph). We show that, for $t \geq 2$, this problem is APX-hard, even for highly restricted instances. Our main result is a $2t$-approximation algorithm for general instances. This is based on a novel *fractional* version of the Local Ratio technique. One implication of this result is the first constant factor approximation for non-overlapping alignment of genomic sequences. We also derive a bi-criteria *polynomial time approximation scheme* for a restricted subclass of t-interval graphs.

1 Introduction

1.1 Problem Statement and Motivation

We consider the problem of scheduling jobs that are given as *groups* of non-intersecting segments on the real line. Each job J_j is associated with a *t-interval*, I_j, which consists of up to t disjoint segments, for some $t \geq 1$, and a weight (profit), w_j; two jobs are in conflict

*A preliminary version of this paper appeared in the proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, 2002.

†Computer Science Dept., Technion, Haifa 32000, Israel. E-mail: {reuven,naor,hadas,csira}@cs.technion.ac.il.

‡This research was supported by the fund for the promotion of research at the Technion.

§Dept. of Computer Science, University of Iceland, IS-107 Reykjavik, Iceland, E-mail: mmh@hi.is.

if any of their segments intersect. The objective is to schedule on a single machine a subset of non-conflicting jobs whose total weight is maximum.

An instance of our problem can be modeled as the intersection graph of t-intervals, known as a *t-interval graph*. Each vertex in the graph corresponds to an interval that has been *"split"* into t parts, or segments, such that two vertices u and v are adjacent if and only if some segment in the interval corresponding to u intersects with some segment in the interval corresponding to v (see Figure 1). In the special case where intersections can occur only between the i-th segments of two intervals, $1 \leq i \leq t$, we get the subclass of *t-union graphs*.[1] Note that 1-interval graphs are precisely interval graphs. Our problem can be viewed as the *maximum weight independent set problem* (MWIS) restricted to a weighted t-interval graph $G(V, E)$, where we seek a subset of non-adjacent vertices $U \subseteq V$, such that the weight of U is maximized.

We describe below several practical scenarios involving t-interval graphs.

Transmission of Continuous-media Data. Traditional multimedia servers transmit data to the clients by *broadcasting* video programs at pre-specified times. Modern systems allow to replace broadcasts with the allocation of video data streams to individual clients *upon request*, for some time interval (see, e.g., [34, 6]). In this operation mode, a client may wish to take a break, and resume viewing the program at some later time. This scenario is natural, e.g., for video programs that are used in remote education [15].

Suppose that a client starts viewing a program at time t_0. At time t_1 the client takes a break, and resumes viewing the program at t_2, till the end of the program (at t_3). This scenario can be described by a *split interval*, I, that consists of two segments: $I^1 = (t_0, t_1)$ and $I^2 = (t_2, t_3)$.

The scheduler may get many requests formed as split intervals; each request is associated with a profit which is gained by the system only if *all* of the segments corresponding to the request are scheduled. The goal is to schedule a subset of non-overlapping requests that maximizes the total profit, i.e., find a MWIS in the intersection graph of the split intervals.

Most of the previous work in this area describe analytic models (e.g., [32]) or experimental studies, in which VCR-like operations can be used by the clients (see [6, 13, 34, 46]); however, these studies focus on the efficient use of system resources while supporting such operations, rather than on the scheduling problem.

Linear Resource Allocation. Another application is allocation of multiple linear resources [24]. Requests for a linear resource can be modeled as intervals on a line; two requests for a resource can be scheduled together unless their intervals overlap. A disk drive is a linear resource when requests are for contiguous blocks [38]. A linear array net-

[1] We give the precise definition in Section 2.1.

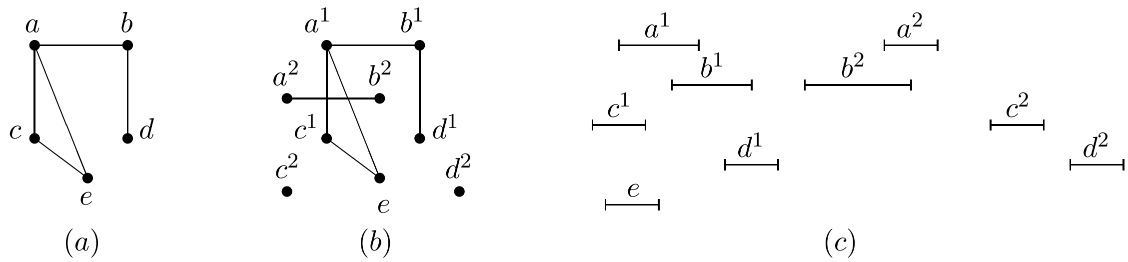

Figure 1: A 2-interval graph (a), corresponding interval (segment intersection) graph (b), and interval system (c).

work is a linear resource, since a request for bandwidth between processors i and j requires that bandwidth be allocated on *all* intervening edges. Consider a computer system that consists of a linear array network and a large disk, shared by a set of processors. A scheduler must decide when to schedule requests, where each request may comprise of distinct requests to these two linear resources, e.g., "a certain amount of bandwidth between processors 4 and 7, and a lock on blocks 1000-1200 of the disk". Two requests are in conflict if they overlap on the disk or in their bandwidth requirements. Thus, when the goal is to maximize the amount of requests satisfied by the system, we get an instance of the MWIS problem on the subclass of *2-union* graphs. Indeed, each segment in a 2-interval represents an allocation of one of the resources (e.g., first segment is bandwidth allocation, and the second segment is the allocation of blocks on the disk to a given request. In general, with t different resources we get an instance of MWIS on t-union graph.

Genomic Sequence Similarity. One of the more fundamental problems in computational biology is to determine the similarity of substructures. We consider here genomic sequences (DNA, protein) and define the substructures to be contiguous subsequences. The similarity score of a substructure is generally related to the local alignment, or editing distance, between the two subsequences.

When considering the total similarity of two whole sequences, we can view this as being made up by the combination of individual substructures. Due to genomic rearrangements, the order of the subsequences need not be preserved between the two genomes. The non-overlapping local alignment problem seeks a collection of substructures, each corresponding to pairs (S_i, T_i) of subsequences of the genomes S and T, where none of the subsequences overlap (neither in S nor in T). The objective is to maximize the sum of the similarity scores of the substructures.

As an example, suppose $S = xxxAxxBC$ and $T = C'yyA'zzB'$, where A, B, and C are sequences with similarity score of 15, 20, and 11 to sequences A', B', and C', respectively. Then, the total similarity of S and T would be 46.

We may assume that the input sequences have been preprocessed to give subsequence pairs with non-zero similarity. Each such substructure (S_i, T_i) corresponds to a 2-interval, formed by the interval that S_i forms with S one on hand, and the interval that T_i forms with T, on the other hand. The non-overlapping restriction of the problem implies that the set of 2-intervals that we find needs to be mutually independent. Hence, the non-overlapping local alignment problem corresponds to MWIS in 2-union graphs.

The common total similarity of t sequences simultaneously can similarly be modelled as the MWIS problem in t-union graphs. Previously, the problem was only considered in the case where the projections of input boxes did not contain one another, i.e., the case of proper t-union graphs. While making the problem easier, this restriction is not intrinsic to the biological problem.

More generally, the multiple alignment of t sequences corresponds to the MWIS problem in t-union graphs. Previously, the problem was only considered in the case where the projections of input boxes did not contain one another, i.e., the case of proper t-union graphs. While making the problem easier, this restriction is not intrinsic to the biological problem.

Computational Geometry. The problem of finding an independent set among a set of multi-dimensional axis-parallel boxes is of independent interest in computational geometry. It corresponds to the MWIS problem in t-union graphs.

1.2 Our Results

We provide a comprehensive study of the MWIS problem in t-interval graphs. In Section 2, we show that the problem is APX-hard even on highly-restricted instances, namely, on $(2, 2)$-union graphs (defined in Section 2.1). In Section 3 we discuss some structural properties of t-interval graphs. In particular, we derive a bound on the inductiveness of a t-interval graph. As a corollary, we extend the best bound known on the chromatic number of t-interval graphs of Gyárfás [20]. We show this bound to be asymptotically optimal.

In Section 3.2, we study the MWIS problem on 2-interval graphs. We show that a simple greedy algorithm achieves the factor $O(\min\{\log R, \log n\})$, where R is the ratio between the longest and shortest segment in the instance.

Our main result (in Section 4) is a $2t$-approximation algorithm for MWIS in any t-interval graph, for $t \geq 2$, which is based on a novel *fractional* version of the Local Ratio technique. (The Local Ratio technique was first developed in [4] and later extended by [3, 5].) We use the fractional Local Ratio technique to round a fractional solution obtained from a linear programming relaxation of our problem. We expect that our non-standard use of the Local Ratio technique will find more applications. Indeed, recently, this technique was used for obtaining improved bounds for MWIS in the intersection graph of axis parallel rectangles in the plane [31].

As we shall see, the MWIS in t-interval graphs properly includes the k-dimensional matching problem. For this unweighted problem the best approximation factor known is $k/2 + \epsilon$, for any $\epsilon > 0$ [26]. Hazan, Safra, and Schwartz [39] have recently shown that it is hard to approximate the k-dimensional matching problem within an $O(k/\log k)$ factor unless P = NP. Thus, our results are close to best possible.

For the class of t-union graphs, we develop (in Section 5) a $(1, 1+\epsilon)$-approximation scheme with respect to the optimal profit and the latest completion time of any interval in some optimal solution. In particular, our scheme gets as parameter $T_{\mathcal{O}}$, the latest completion time of an interval in some optimal schedule, and outputs a subset of intervals of optimal profit, in which the latest completion time of any interval is at most $T_{\mathcal{O}}(1+\epsilon)$.

1.3 Related Work

We mention below several works that are related to ours.

Split interval graphs. Many NP-hard problems, including MWIS [17, 18], can be solved efficiently in interval graphs. Split interval graphs have a long history in graph theory [44, 19, 40, 45], and more recently, union graphs have been studied under the name of *multitrack* interval graphs [30, 21, 29]. We mention some of the main results. For any fixed $t \geq 2$, determining whether a given graph is a t-interval (t-union) graph is NP-complete [45] ([21], respectively). 2-union graphs contain trees [44, 30] and more generally all outerplanar graphs [29], while 3-interval graphs contain the class of planar graphs [40]. Graphs of maximum degree Δ are $\lceil \frac{1}{2}(\Delta+1) \rceil$-interval graphs [19]. The complete bipartite graph, $K_{m,n}$, is a t-interval and t-union graph for $t = \lceil (mn+1)/(m+n) \rceil$ [44, 21].

Union graphs, which constitute a sub-family of split interval graphs, were also considered in several papers. Bafna *et al.* [2] considered the problem of finding a weighted independent set in t-union graphs in the context of an application coming from computational biology. The union graphs considered in [2] are *proper*, i.e., there is no containment between segments. For the weighted independent set problem in proper t-union graphs, the paper [2] shows that the problem is NP-hard and gives a $(2^t - 1 + 1/2^t)$-approximation algorithm. This is obtained by mapping the problem to MWIS in (2^t+1)-claw free graphs, noting that proper t-union graphs are (2^t+1)-claw free. Recently, Chlebík and Chlebíkova [12] showed that proper t-union graphs are $2t+1$-claw free. Using an algorithm of Berman [7] this gives a $t + 1/2$-approximation of MWIS in proper t-union graphs. Berman et al. showed in [8] that a simple $O(n \log n)$ algorithm (based on the local ratio technique) yields a factor of 3 for proper 2-union graphs.

Coupled-tasks and flow shop scheduling. The problem of scheduling 2-intervals (known as *coupled-task scheduling*) was considered in the area of machine scheduling, with the objective of minimizing the overall completion time, or *makespan* (see e.g. [35, 42]).

Relaxed versions of the problem, that require only a lower bound on the time that elapses between the schedules of the two tasks of each job (also called *time-lag problems*) were studied, e.g., in [37, 14, 11].

An instance of our problem can be viewed as an instance of the *flow shop* problem, in which the segments and break times are represented by *tasks* that need to be processed on a set of $m = 2t+1$ machines. (The precise transformation is given in Section 5.) In general, the flow shop problem, where the objective is to minimize the makespan, is NP-complete even on three machines ([16]). The best result known is $O(\log^2(m\mu)/\log\log(m\mu))$-approximation algorithm, where μ is the maximum number of operations per job, and m is the number of machines ([41, 43]). Hall [22] gave a PTAS for this problem in the case where m is fixed (but arbitrary).

2 Preliminaries

2.1 Definitions and Notation

Given a t-interval graph, $G = (V, E)$, we assume that each vertex $v \in V$ is mapped to a set of at most t segments, and we call v a *split interval*. Suppose that segment I is one of the segments that vertex v is mapped to, then we say that I belongs to v and denote it by (v, I). We denote by $\mathcal{I}(\mathcal{G})$ the collection of segments (or intervals) on the real line, partitioned into disjoint *groups*, where each group is associated with a split interval. A t-interval graph is *proper* if no segment properly contains another segment.

In the subfamily of t-union graphs, the segments associated with each vertex can be labeled in such a way that for any two vertices u and v, the ith segment of u and the ℓth segment of v never intersect for $1 \leq i, \ell \leq t$, and $i \neq \ell$. Union graphs correspond also to certain geometric intersection graphs. The t segments are viewed as intervals on orthogonal axes, corresponding to a t-dimensional box; two boxes intersect if their projections on any of the t axes do. We further define subclasses of union graphs, where coordinates are all integral and segments are half-open. In (a, b)-union graphs, a subclass of 2-union graphs, all x-segments are of length a and y-segments of length b.

Given a graph $G = (V, E)$, we denote by $N(v)$ the set of neighbors of $v \in V$, and by $N[v]$ the closed neighborhood of v, $\{v\} \cup N(v)$. A $(k+1)$-*claw* is a graph consisting of a center vertex adjacent to $k+1$ mutually non-adjacent vertices. A graph is called $(k+1)$-*claw free* if it contains no k-claw as an induced subgraph.

Finally, we define our performance measures. Denote by OPT an optimal algorithm. The *approximation factor* of an algorithm $\mathcal{A}$ is r if for every finite input instance I, $\mathcal{A}(I)/OPT(I) \geq 1/r$, where $\mathcal{A}(I)$ and $OPT(I)$ are the values of $\mathcal{A}$ and OPT on I. A *polynomial time approximation scheme (PTAS)* is an algorithm which takes as input both the instance I and an error bound ϵ, has performance guarantee $R(I, \epsilon) \leq (1 + \epsilon)$, and runs

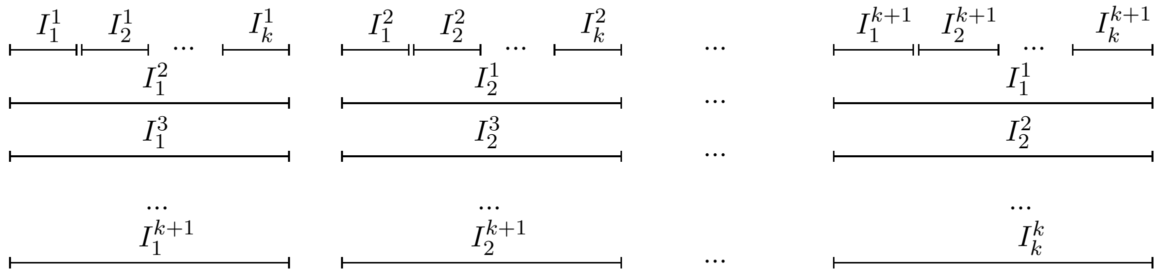

Figure 2: A 2-interval graph in which every vertex has k independent neighbors.

in time polynomial in $|I|$. A (β, ϵ) *bi-criteria PTAS* is a PTAS which is a β-approximation in one optimization criterion, and a $(1+\epsilon)$-approximation in the other criterion.

2.2 Hardness Results

The independent set problem in interval graphs is easy to solve exactly, since interval graphs always contain a *simplicial* vertex, i.e., a vertex whose neighborhood is a clique. In fact, most approximation algorithms for independent sets on geometric intersection graphs are based on a related relaxed property: there always exists a vertex whose neighborhood does not contain a large independent set. We first show that for general t-interval graphs this property does not hold.

Observation 2.1 *For any $n \geq 2$, there exists a 2-interval graph G on n vertices, in which every vertex has $\Omega(\sqrt{n})$ independent neighbors.*

Proof: We show how to construct a 2-interval graph, in which every vertex has k independent neighbors. We construct the graph from $(k+1)$ sets of intervals; each set consists of k intervals, and each interval is composed of two segments: a *short* and a *long* segment. All the short segments in the input are of the same length, and likewise all the long segments.

The graph is constructed as follows. For $\ell = 1, \ldots, k$, all the short segments of the intervals in the ℓth set, $I_1^\ell, \ldots, I_k^\ell$, intersect the long segments of the ℓth intervals in the sets $1, \ldots, k+1$, i.e., $I_\ell^1, \ldots, I_\ell^{k+1}$, excluding I_ℓ^ℓ. Finally, all the short segments of the intervals in the $(k+1)$-th set intersect the long segment of the interval I_ℓ^ℓ, $1 \leq \ell \leq k$ (see Figure 2). Thus, we get that any interval I_j^ℓ with $\ell \neq j$, intersects k non-intersecting intervals in the jth set, and I_ℓ^ℓ intersects k non-intersecting intervals in the $(k+1)$-th set.

Note that since $k(k+1) \leq n$, we may have some remaining intervals, which are not contained in any set. We can place the segments of each such interval on the line such that its long segment intersects all the intervals in the ℓth set, for some $1 \leq \ell \leq k+1$, providing that interval k independent neighbors. $\square$

We note that the above construction can be modified to hold for 2-union graphs. We now give a structural result that implies hardness of approximation for a highly restricted class of proper 2-union graphs. A *degree-3 graph* is one of maximum degree at most three.

Theorem 2.2 *The class of* $(2,2)$*-union graphs includes the class of degree-3 graphs.*

Proof: A *linear forest* is a collection of disjoint paths. A path can be represented as a collection of length-2 half-closed intervals between integral endpoints, e.g., $[0,2), [1,3), [2,4)$, etc. Thus, a union of a pair of linear forests can be represented as a $(2,2)$-union graph. Akiyama, Exoo, and Harary [1] showed that degree-3 graphs can be represented as a union of two linear forests. Namely, they showed that for a degree-3 graph G, we have that $la(G) = 2$, where $la(G)$ denotes the *linear arboricity* of G or the minimum number of classes in a partition of $E(G)$ such that each class induces a linear forest. □

It follows that the MWIS problem is APX-hard on unweighted $(2,2)$-union graphs, since the (unweighted) MIS problem is APX-hard on degree-3 graphs (see [9, 25]). It also implies equivalent hardness results for other optimization problems that are hard to approximate on degree-3 graphs.

Corollary 2.3 *The MWIS problem is APX-hard on unweighted* $(2,2)$*-union graphs.*

Segments of unit size, whose start points are integral, are called *unit segments*. t-interval graphs of unit segments can be characterized precisely.

For some $k > 1$, let $S = \{1, 2, \ldots, n\}$, and let C be a collection of subsets of S, where each subset is of size at most k. The *k-set packing* problem is that of finding a maximum cardinality sub-collection $C' \subseteq C$, such that the intersection of any two sets in C' is empty. In the *weighted* version, each subset has a weight, and we seek a sub-collection C' of maximum weight.

Lemma 2.4 *The k-set packing problem is equivalent to MWIS in the special class of k-interval graphs of unit segments.*

Proof: There is a bijective mapping between unit segments and the set S, where $[i, i+1)$ maps to i, for all values of i. Thus, there is a bijective mapping between sets of up to k elements from S and sets of up to k unit segments. □

A special case of k-set packing is the *k-dimensional matching* problem. Here, S is partitioned into subsets $S_1, S_2, \ldots, S_k$, and each set in C contains exactly one element from each S_i. The k-dimensional matching problem is similarly equivalent to MWIS in the special class of k-union graphs of unit segments. The former problem is NP-hard to approximate within factor $O(k/\log k)$ [39], while the best factor known is $k/2+\epsilon$, for any $\epsilon > 0$ [26]. We note that the 2-set packing problem is equivalent to the (polynomially solvable) edge cover problem, while 3-dimensional matching is APX-hard [36].

Corollary 2.5 *MWIS in $(1,1)$-interval graphs is polynomial solvable. MWIS in $(1,1,1)$-union graphs is APX-hard.*

The correspondence between $(1,1)$-union graphs to line graphs of bipartite graphs, and the resulting polynomial solvability of MWIS, was shown by Halldórsson *et al.* [24].

3 Greedy Algorithms

3.1 Coloring t-Interval Graphs

For a t-interval graph G (see in Figure 1(a) for $t = 2$), let G^* denote the graph formed by the intersection of the segments of the intervals (Figure 1(b)). The clique number, $\omega(G^*)$, denotes the maximum number of segments crossing a point on the real line.

Theorem 3.1 *For any t-interval graph G, there is a vertex v in G such that*

$$d(v) \le 2t(\omega(G^*) - 1) - 1.$$

Proof: Since each vertex in G corresponds to up to t vertices of G^*, $|V(G^*)| \le t \cdot |V(G)|$, and since each edge in G corresponds to one or more edges in G^*, $|E(G)| \le |E(G^*)|$. Since G^* is an interval graph, there is a simplicial ordering of the graph so that each vertex v_i has at most $\omega(G^*) - 1$ neighbors among the vertices $v_{i+1}, \ldots$. Thus, the number of edges in G^* is at most $(\omega(G^*) - 1)|V(G^*)|$; in fact, it must be strictly less, since the last vertex has later neighbors. It follows that the average degree of G is bounded by

$$\overline{d}(G) = \frac{2|E(G)|}{|V(G)|} \le 2t\frac{|E(G^*)|}{|V(G^*)|} < 2t(\omega(G^*) - 1).$$

Hence, the minimum degree of G is at most $2t(\omega(G^*) - 1) - 1$. □

This leads to a simple coloring algorithm: find a vertex v satisfying the lemma, color the remaining graph $G \setminus v$, and finally color v with the smallest color not used by previously colored neighbors. This results in a $2t(\omega(G^*) - 1)$-coloring.

The above gives a $2t$-approximation for coloring t-interval graphs via a greedy algorithm. Gyárfás [20] showed that the chromatic number of a t-interval graph G is at most $2t(\omega(G) - 1)$, where $\omega(G)$ is the clique number of the graph.

Corollary 3.2 *A greedy algorithm colors G using $2t(\omega(G^*) - 1)$ colors.*

Observe that this bound is obtained without knowledge of the underlying interval representation of G^*; this is important since deducing the representation is known to be NP-hard [45]. We show that this is about the best bound on $\chi(G)$ one can obtain in terms of $\omega(G^*)$, within a constant factor.

Lemma 3.3 *For infinitely many t, there is a proper t-interval graph G such that $\omega(G) = (t-1)\omega(G^*)$.*

Proof: Let p be any prime number and $\mathbb{Z}_p$ be the finite field over $\{0, 1, \ldots, p-1\}$. Let $t = p+1$. Let $C_{i,j}$ and D_i, $i, j \in \mathbb{Z}_p$, be any disjoint unit segments. We shall construct a system of t-intervals $I_{x,y}$, $x, y \in \mathbb{Z}_p$, and show that the t-intervals are pairwise overlapping, i.e. that any pair contains a common segment.

Let $I_{x,y} = \{C_{i,ix+y \bmod p} : i \in \mathbb{Z}_p\} \cup \{D_x\}$, for each $x, y \in \mathbb{Z}_p$. Clearly, the sets contain t segments each. Consider a pair of t-intervals $I_{x,y}$, $I_{x',y'}$. If $x = x'$, then both t-intervals contain the segment $D_x = D_{x'}$. Otherwise, there exists an $i \in \mathbb{Z}_p$ that is a solution to the modular equation $i(x - x') \equiv (y' - y)(\text{mod} p)$. Then, both t-intervals $I_{x,y}$ and $I_{x',y'}$ contain the segment $C_{i,ix+y \bmod p} = C_{i,ix'+y' \bmod p}$.

It follows that the intersection graph on these t-intervals is a clique on $p^2 = (t-1)^2$ vertices. On the other hand, each segment $C_{i,j}$ is contained in exactly $(t-1)$ t-intervals $I_{x,y}$ (namely those for which $j = ix + y \bmod p$) and the same holds for each D_i. Thus, the clique number of G^* is $t - 1$. □

3.2 Greedy Independent Set Algorithms

In this section, we study a greedy algorithm for the special case where $t = 2$, in order to motivate the use of more complicated techniques in later sections.

Recall from Observation 2.1 that, in a 2-interval graph, the neighborhood of every vertex may include many independent vertices. Thus, purely greedy methods are bound to fail. Consider, for instance, the optimal greedy algorithm for independent sets in interval graphs that iteratively adds the interval with the leftmost right endpoint. An analogous method for 2-interval graphs could be to iteratively select the 2-interval with the leftmost right endpoint of the *first segment*, among all 2-intervals that do not intersect previously chosen 2-intervals. This algorithm, which we call Sort-and-Select, cannot be expected to perform well on all 2-interval graphs. However, it performs well under certain circumstances, which allows us to partition the instance into well-solvable subcases.

Theorem 3.4 *Let G be a 2-interval graph where*

- *the first segment is no shorter than the second, and*
- *the ratio between the length of the shortest and longest second segment is at most 4.*

Then, the approximation factor of Sort-and-Select is 6*.*

Proof: Let I be the interval chosen first by Sort-and-Select. We claim that I intersects at most 6 independent intervals. Namely, the second segment of I is at most four times the length of the shortest segment in the graph; as a result, it intersects at most five independent segments/vertices. Also, since the first segment is furthest to the left of all segments in the graph, it does not intersect two independent vertices. Thus, among the intervals eliminated by the addition of I to the solution, the optimal solution can contain at most 6. By induction, the algorithm then achieves an approximation factor of 6. □

Using the Local-Ratio technique, which is discussed in depth in the next section, one can obtain the same factor for the weighted case. Also, by a similar argument, one can argue a factor of 3 for the case of proper 2-interval graphs.

Given a general 2-interval graph, we first divide the intervals into those where the first segment is shorter than the second segment and those where the first segment is at least as long as the second. This gives us two instances, which can be viewed as symmetric by reversing the direction of the real line. Thus, by increasing the approximation factor by a factor of 2, we can assume that in our instance the first segments are no shorter than the second segments.

We can partition the instance into $(\lg R)/2$ sub-instances, or *buckets*, where R is the ratio between the longest to shortest (S) second segment. The bucket G_i consists of intervals with second segments in the range $[4^{i-1}S, 4^i S)$, for $i = 1, 2, \ldots, \lceil(\lg R)/2\rceil$. Each bucket satisfies the conditions of Theorem 3.4; thus, the largest of the independent sets found in each bucket by Sort-and-Select is a $6 \lg R$ approximation.

Note that we can represent the n second segments in the input by $2n$ endpoints on the line, and define the length of each segment as the number of endpoints that lie between its left and right endpoints plus one. Then, the maximal possible length of a segment is $2n-1$, and the number of buckets is $B = min\{\lg R, \lg(2n-1)\}$. Hence, we obtain the following result.

Theorem 3.5 *There is a greedy partitioning algorithm that approximates the maximum independent set in 2-interval graphs within a factor of* $O(\min\{\log R, \log 2n\})$*.*

4 A $2t$-approximation Algorithm

In this section we describe a $2t$-approximation algorithm for the maximum weight independent set problem in a t-interval graph $G = (V, E)$. The algorithm is based on rounding a fractional solution derived from a linear programming relaxation of the problem. The standard linear programming relaxation of the maximum weight independent set problem

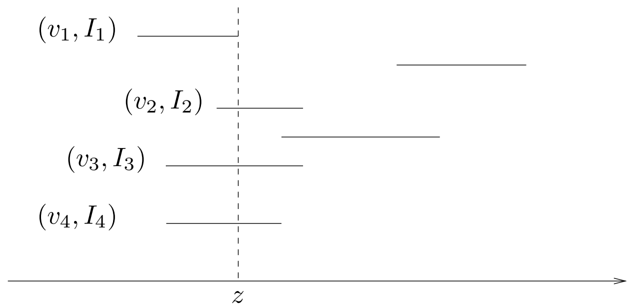

Figure 3: The interval clique (v_1, v_2, v_3, v_4) is defined by z, the right endpoint of (v_1, I_1).

is the following. For each $v \in V$, let $x(v)$ be the linear relaxation of the indicator variable for v, i.e., whether v belongs to the independent set. Let $\mathbf{w}, \mathbf{x} \in \mathbb{R}^{|V|}$ be a weight vector and a relaxed indicator vector, respectively.

$$\begin{array}{ll} \text{maximize} \quad \mathbf{w} \cdot \mathbf{x} & \textit{subject to}: \\ \text{for each clique } \mathcal{C} \in G: & \sum_{v \in \mathcal{C}} x(v) \leq 1 \end{array}$$

A feasible solution for the above linear program, whose value is an upper bound on the maximum weight independent set problem in the graph, can be obtained from the Lovász ϑ-function [33]. However, as we shall see, it is not necessary to optimize over all cliques in the case of t-interval graphs. We say that a clique $\mathcal{C}$ in the graph is an *interval clique* if for every vertex $v \in \mathcal{C}$, there is a segment $I \in v$ such that the intersection of $((v, I) | v \in \mathcal{C})$ is non-empty. It is easy to see that the interval cliques are defined by the set of right endpoints of the segments in $\mathcal{I}(\mathcal{G})$ as follows. Each right endpoint z corresponds to the clique defined by the vertices containing z. (See Figure 4 for an example.) Therefore, the number of interval cliques in a t-interval graph is linear in the number of segments.

We now further relax the maximum weight independent set problem and consider only interval cliques. In the integral case, let $x(v)$ denote the indicator variable of vertex v, and for each $I \in v$, $x(v, I) = x(v)$. In the linear relaxation (P), for each interval clique $\mathcal{C}$, we require that the sum of the variables $(v, I) \in \mathcal{C}$ is at most 1. It suffices to require, for each vertex v and $I \in v$, $x(v, I) \geq x(v)$, since only $x(v)$ participates in the objective function, and therefore in an optimal solution, without loss of generality, $x(v, I) = x(v)$.

$$
\begin{array}{|lrcl|}
\hline
& \text{(P)} \quad \text{maximize} \quad \mathbf{w}\cdot\mathbf{x} \quad \textit{subject to}: & & \\
\text{for each interval clique } \mathcal{C}\text{:} & \displaystyle\sum_{(v,I)\in\mathcal{C}} x(v,I) & \leq & 1 \\
\text{for each } v\in V \text{ and } I\in v\text{:} & x(v,I)-x(v) & \geq & 0 \\
\text{for each } v\in V \text{ and } I\in v\text{:} & x(v),x(v,I) & \geq & 0 \\
\hline
\end{array}
$$

Since the number of interval cliques in a t-interval graph is linear in the number of segments, an optimal solution to (P) can be computed in polynomial time.

The heart of our rounding algorithm is the following lemma. It can be viewed as a fractional analog of Theorem 3.1.

Lemma 4.1 *Let* $\mathbf{x}$ *be a feasible solution to* (P)*. Then, there exists a vertex* $v \in V$ *satisfying:*

$$\sum_{u\in N[v]} x(u) \leq 2t$$

Proof: For two adjacent vertices u and v, define $y(u,v) = x(v)\cdot x(u)$. Define $y(u,u) = x(u)^2$. For a segment I, let $R(I)$ be the interval clique defined by the right endpoint of I ($I \in R(I)$). We prove the claim using a *weighted* averaging argument, where the weights are the values $y(u,v)$ for all pairs of adjacent vertices, u and v.

Consider the sum $\sum_{v\in V}\sum_{u\in N[v]} y(u,v)$. An upper bound on this sum can be obtained as follows. For each $v \in V$, consider all segments $I \in v$, and for each (v,I), add up $y(u,v)$ for all (u,J) that intersect with (v,I) (including (v,I)). In fact, it suffices to add up $y(u,v)$ only for segments (u,J) such that $(u,J) \in R(I)$, and then multiply the total sum by 2. This suffices since: (a) If, for segments (v,I) and (u,J), the right endpoint of I precedes the right endpoint of J, then (v,I) "sees" (u,J) and vice-versa. Since $y(u,v) = y(v,u)$, each of them contributes the same value to the other. (b) For segments (v,I) and (u,J), the constraints of (P) imply that $x(v,I) = x(v)$ and $x(u,J) = x(u)$. Hence, it follows from (a) and (b) that the mutual contribution of two segments (u,J) and (v,I) that intersect depends only on u and v, i.e., it is $y(u,v)$. Thus,

$$\sum_{v\in V}\sum_{u\in N[v]} y(u,v) \leq 2\cdot\sum_{v\in V}\sum_{I\in v}\sum_{(u,J)\in R(I)} y(u,v)$$

Since

$$\sum_{(u,J)\in R(I)} y(u,v) \leq \; x(v)\cdot \sum_{(u,J)\in R(I)} x(u) \leq x(v)$$

we get that

$$\sum_{v\in V}\sum_{u\in N[v]} y(u,v) \leq 2t\cdot\sum_{v\in V} x(v).$$

Hence, there exists a vertex v satisfying

$$\sum_{u \in N[v]} y(u,v) \leq 2t \cdot x(v). \tag{1}$$

If we factor out $x(v)$ from both sides of (1) we obtain the statement of the lemma. □

We now define a fractional version of the Local Ratio technique. The proof of the next lemma is immediate.

Lemma 4.2 *Let* $\mathbf{x}$ *be a feasible solution to* (P)*. Let* $\mathbf{w_1}$ *and* $\mathbf{w_2}$ *be a decomposition of the weight vector* $\mathbf{w}$ *such that* $\mathbf{w} = \mathbf{w_1} + \mathbf{w_2}$*. Let* $r > 0$*. Suppose that* $\mathbf{y}$ *is a feasible integral solution vector to (P) satisfying:* $\mathbf{w_1} \cdot \mathbf{y} \geq r(\mathbf{w_1} \cdot \mathbf{x})$ *and* $\mathbf{w_2} \cdot \mathbf{y} \geq r(\mathbf{w_2} \cdot \mathbf{x})$*. Then,*

$$\mathbf{w} \cdot \mathbf{y} \geq r(\mathbf{w} \cdot \mathbf{x}).$$

The rounding algorithm will apply a Local Ratio decomposition of the weight vector $\mathbf{w}$ with respect to an optimal solution $\mathbf{x}$ to linear program (P). The algorithm proceeds as follows.

1. If no vertices remain, return the empty set. Otherwise, proceed to the next step.
2. Define $V_0 = \{v \in V | w(v) < 0\}$. If V_0 is non-empty, return $\mathcal{I}$ - the recursive solution for $V \setminus V_0$. Otherwise, proceed to the next step.
3. Let $v' \in V$ be a vertex satisfying $\sum_{u \in N[v']} x(u) \leq 2t$. Decompose $\mathbf{w}$ by $\mathbf{w} = \mathbf{w_1} + \mathbf{w_2}$ as follows:
$$w_1(u) = \begin{cases} w(v') & \text{if } u \in N[v'], \\ 0 & \text{otherwise.} \end{cases}$$
(In the decomposition, the component $\mathbf{w_2}$ may be non-positive.)
4. Solve the problem recursively using $\mathbf{w_2}$ as the weight vector. Let $\mathcal{I}'$ be the independent set returned.
5. If $\mathcal{I}' \cup \{v'\}$ is an independent set, return $\mathcal{I} = \mathcal{I}' \cup \{v'\}$. Otherwise, return $\mathcal{I} = \mathcal{I}'$.

Clearly, the set $\mathcal{I}$ is an independent set. We now analyze the quality of the solution produced by the algorithm.

Theorem 4.3 *Let* $\mathbf{x}$ *be an optimal solution to linear program* (P)*. Then, it holds for the independent set* $\mathcal{I}$ *computed by the algorithm that* $w(\mathcal{I}) \geq \frac{1}{2t} \cdot \mathbf{w} \cdot \mathbf{x}$

Proof: The proof is by induction on the number of vertices having positive weight. Any vertex deleted by the algorithm is considered to have zero weight. At the basis of the induction (Step (1)), the inductive hypothesis holds, since the weight vector is considered to be zero. We now prove the inductive step.

In Step (2), if V_0 is non-empty, by the inductive hypothesis, $w(\mathcal{I}) \geq \frac{1}{2t} \cdot \mathbf{w} \cdot \mathbf{x}$. Extending $\mathbf{w}$ to include the non-positive components that were deleted in Step (2) can only decrease the RHS, and therefore the inequality still holds.

In Steps (3)-(5), let $\mathbf{y}$ and $\mathbf{y'}$ be the indicator vectors of the sets $\mathcal{I}$ and $\mathcal{I}'$, respectively. By the decomposition in Step (3), weight vector $\mathbf{w_2}$ has less positive weight vertices than $\mathbf{w}$. Therefore, by the inductive hypothesis, $\mathbf{w_2} \cdot \mathbf{y'} \geq (1/2t) \cdot \mathbf{w_2} \cdot \mathbf{x}$. Since $w_2(v') = 0$, it also holds that $\mathbf{w_2} \cdot \mathbf{y} \geq (1/2t) \cdot \mathbf{w_2} \cdot \mathbf{x}$. From Step (5) of the algorithm it follows that at least one vertex from $N[v']$ belongs to $\mathcal{I}$. Hence, $\mathbf{w_1} \cdot \mathbf{y} \geq (1/2t) \cdot \mathbf{w_1} \cdot \mathbf{x}$. Thus, by Lemma 4.2, it follows that

$$\mathbf{w} \cdot \mathbf{y} \geq \frac{1}{2t} \cdot \mathbf{w} \cdot \mathbf{x}$$

We have thus proved that $\mathcal{I}$ is a $2t$-approximate solution to the MWIS problem. □

We now outline an alternative way of using Lemma 4.1 to obtain the same approximation factor. Given an optimal solution $\mathbf{x}$ to linear program (P), a *multicoloring* of V by a set X is a mapping $\psi : V \rightarrow 2^X$ such that $|\psi(v)| = x(v)$ for each vertex v, and $\psi(v) \cap \psi(u) = \emptyset$ for each edge $(u, v) \in E(G)$. Since $\mathbf{x}$ is a feasible solution to (P), a repeated application of Lemma 4.1 results in a multicoloring with values in the closed interval $[0, 2t]$.

To view this as a multicoloring, it may be easier to discretize the instance within any desired precision by multiplying the $x(v)$'s by a sufficiently large integer L. Then the values assigned are positive integers in the range $1, \ldots, 2tL$. A continuous viewpoint is to assign each vertex a collection of contiguous segments; if we use Lemma 4.1 to assign the values one by one, we can always guarantee that a vertex v can be mapped to segments from $[0, 2t]$ of combined length $x(v)$ without overlapping any of the segments to which its neighbors are mapped to. In fact, by always mapping a vertex to the smallest available values, we need never use more than n disjoint segments for any vertex.

Let $0 = z_0 < z_1 < \cdots < z_{k-1}$ denote the values where the multicoloring changes, and let $z_k = 2t$. Thus, the coloring remains unchanged in the segment $[z_i, z_{i+1})$, $i = 0, \ldots, k-1$. Consider the sets $S_i = \{v \in V : x_i \in \psi(v)\}$, for $i = 0, \ldots, k-1$. Since ψ is a multicoloring, the S_i's are independent sets in G. Let $\mathcal{I}$ be the set S_i of maximum weight, $\sum_{v \in S_i} w(v)$.

Theorem 4.4 *$w(\mathcal{I})$ is a $2t$-approximate independent set.*

Proof: Observe that the amount of color values to which vertex v is mapped is $x(v)$, and we can represent them by $\sum_{S_i \ni v}(z_i - z_{i+1}) = x(v)$. We have that

$$\begin{aligned}\sum_{v\in V} w(v)x(v) &= \sum_{v\in V} w(v) \sum_{S_i \ni v}(z_{i+1} - z_i) = \sum_{S_i}(z_{i+1} - z_i) \sum_{v\in S_i} w(v) \\ &= \sum_{i=0}^{k-1}(z_{i+1} - z_i)w(S_i) \leq \sum_{i=0}^{k-1}(z_{i+1} - z_i)w(\mathcal{I}) = 2tw(\mathcal{I}).\end{aligned}$$

□

5 A Bi-criteria Approximation Scheme for Union Graphs

Recall that MWIS is APX-hard already on $(2,2)$-union graphs. We consider below the larger subclass of t-union graphs in which the possible number of segment lengths is bounded by some constant. For this subclass we develop a *bi-criteria* PTAS, which finds an MWIS by allowing some delays in the schedule.

Let c_i denote the number of distinct lengths of the i-th segment, $1 \leq i \leq t$, where t is some constant. Recall that, in the flow shop problem, we are given a set of n jobs, $J_1, \ldots, J_n$ that need to be processed on m machines, $M_1, \ldots, M_m$; each job, J_j, consists of m operations, $O_{j,1}, \ldots, O_{j,m}$, where $O_{j,i}$ must be processed without interruptions on the machine M_i, for $p_{j,i}$ time units. Any machine, M_i, can either process a *single* operation at a time, or an *unbounded* number of operations; in the latter case we call M_i a *non-bottleneck* machine. Each job may be processed by at most one machine at any time. For a given schedule, let C_j be the completion time of J_j. The objective is to minimize the *maximum completion time* (or makespan), given by $C_{max} = \max_j C_j$. Denote by C^*_{max} the optimal makespan.

An instance of our problem can be transformed to an instance of the flow shop problem, where each job has $2t + 1$ operations, and the machines M_{2i+1}, $0 \leq i \leq t-1$, are non-bottleneck machines. In our transformation, we apply some ideas from [27, 23, 28]. We represent each t-interval, I_j, as a job J_j, where each segment is associated with an "operation" of the job. In addition, we simulate the breaks with operations of the same lengths that need to be processed on non-bottleneck machines. Similarly, to include the release time r_j of I_j, we add to J_j the operation $O_{j,1}$, whose length is equal to r_j; the machine M_1 is a non-bottleneck machine. Thus, if I_j has t segments, J_j has $2t$ operations.

Recall that, in a union graph, each interval has a due date, d_j, that is equal to its release time plus the sum of its processing times and break times. To simulate these due dates we define a *delivery time*, q_j, for each job, J_j. Let $q_j = -d_j$. We add to J_j the operation $O_{j,(2t+1)}$, where $p_{j,(2t+1)} = q_j$, and M_{2t+1} is a non-bottleneck machine. Our objective then is to minimize the maximum *delivery completion time*, given by $\max_j\{C_j + q_j\} =$

$\max_j\{C_j - d_j\}$. This is equivalent to minimizing the maximum *lateness* of any job, given by $L_j = C_j - d_j$. Hence, our objective can be viewed as minimization of $L_{max} = \max_j L_j$.

Denote by $T_{\mathcal{O}}$ the maximum completion time of an optimal solution for the MWIS instance. Since we look for a MWIS that can be scheduled with maximum lateness at most $\epsilon T_{\mathcal{O}}$, we slightly modify the definition of *lateness*. Let $\tilde{d}_j = d_j - T_{\mathcal{O}}$; then, for any j, $\tilde{d}_j \leq 0$. By setting $q_j = -\tilde{d}_j$, we get that all the delivery times are positive. The maximum lateness is now given by $L_{max} = \max_j\{C_j - d_j + T_{\mathcal{O}}\}$. Indeed, for any job J_j, $C_j \geq d_j$, therefore $L_{max} \geq T_{\mathcal{O}}$, and since in any optimal schedule there are no "late" jobs, the minimal lateness is $L^*_{max} = T_{\mathcal{O}}$.

Our scheme uses as procedure a PTAS for finding a $(1+\epsilon)$-approximation for the flow shop makespan problem with a fixed number of machines (see, e.g., [22]). We represent a t-interval I_j by a $(2t+1)$-vector $(p_{j,1}, \ldots, p_{j,2t+1})$, where $p_{j,1}$ is the release time, $p_{j,2i}$ $(p_{j,2i+1})$, is the length of the i-th segment (break), $1 \leq i < t$, and $p_{j,2t+1}$ $(= q_j)$ is the delivery time of the corresponding job, J_j.

We summarize below the steps of our scheme, which gets as parameters the value of $T_{\mathcal{O}}$ and some $\epsilon > 0$.

1. We scale the parameter values for J_j; that is, we divide the processing and release times by $T_{\mathcal{O}}$, and round each release time down and each break time up to the nearest multiple of ϵ.

2. We guess $\mathcal{O}$, the number of intervals scheduled by OPT;

3. We guess the subset $S_{\mathcal{O}}$ of $\mathcal{O}$ intervals of maximal weight, scheduled by OPT. This is done by guessing the set of vectors representing $S_{\mathcal{O}}$, among which we choose the subset of intervals of maximum weight.

4. Using a PTAS for minimizing the makespan in the flow shop instance of $S_{\mathcal{O}}$, we find a schedule of $S_{\mathcal{O}}$ for which $L_{max} \leq (1+\epsilon)L^*_{max}$.

Note that due to the above rounding, we need to add ϵ to the release times; also, each break time may delay the optimal completion time by ϵ, therefore, by taking $\epsilon' = \epsilon/2t$ we guarantee that the delay of each interval is at most $(1+\epsilon)$ times $T_{\mathcal{O}}$. Finally, we set $L_j = L_j - T_{\mathcal{O}}$; thus, the maximum lateness of any job in our schedule is equal to at most $\epsilon T_{\mathcal{O}} = \epsilon C^*_{max}$.

For the complexity of the scheme, note that Steps 1. and 2. take linear time, and since the possible number of vectors $(p_{j,1}, \ldots p_{j,2t+1})$ is $(2t/\epsilon)^t \prod_{i=1}^t c_i$, we can guess $S_{\mathcal{O}}$ in $O(n^{(t^t \prod_{i=1}^t c_i)/\epsilon^t})$ steps. This is multiplied by the complexity of the PTAS for flow shop.

Theorem 5.1 *Let $t \geq 1$ be some fixed constant. Given a t-union graph with constant number of distinct segment lengths, let $\mathcal{W}$ be the weight of an optimal MWIS, whose latest* *completion time is $T_{\mathcal{O}}$. Then, for any $\epsilon > 0$, there is a PTAS that schedules an independent set of weight at least $\mathcal{W}$, such that* any *interval is late by at most $\epsilon T_{\mathcal{O}}$.*

Acknowledgments. We thank Yossi Azar for many helpful comments on this paper. We also thank two anonymous referees for insightful comments and suggestions.

References

[1] J. Akiyama, G. Exoo, F. Harary. Covering and packing in graphs III: Cyclic and acyclic invariants. *Math. Slovaca*, vol. 30 (1980), 405–417.

[2] V. Bafna, B. Narayanan, and R. Ravi. Nonoverlapping Local Alignments (Weighted Independent Sets of Axis Parallel Rectangles). *Discrete Applied Mathematics*, vol. 71, Special issue on Computational Molecular Biology, 1996, pp. 41–53.

[3] V. Bafna, P. Berman, and T. Fujito. A 2-approximation Algorithm for the Undirected Feedback Vertex Set Problem, *SIAM J. on Disc. Mathematics,* vol. 12, pp. 289–297, 1999.

[4] R. Bar-Yehuda and S. Even. A Local Ratio Theorem for Approximating the Weighted Vertex Cover Problem, *Annals of Discrete Mathematics,* vol. 25, pp. 27–46, 1985.

[5] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A Unified Approach to Approximating Resource Allocation and Scheduling. *Journal of the ACM*, 48:1069-1090, 2001.

[6] P. Basu, A. Narayanan, R. Krishnan, and T. D. C. Little. An Implementation of Dynamic Service Aggregation for Interactive Video Delivery. In *Proc. of SPIE - Multimedia Computing and Networking*, San Jose, CA, January 1998.

[7] P. Berman. A d/2 Approximation for Maximum Weight Independent Set in d-Claw Free Graphs. *Nordic Journal of Computing*, vol. 7, 2000, p. 178.

[8] P. Berman, B. DasGupta, S. Muthukrishnan. Simple approximation algorithm for nonoverlapping local alignments. In *Proc. of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms* 2002, pp. 677–678.

[9] P. Berman, and T. Fujito. Approximating Independent Sets in Degree 3 Graphs. In *Proc. of the 4th Workshop on Algorithms and Data Structures (WADS'95)* Lecture Notes in Computer Science, 955, Springer-Verlag, 1995, pp. 449–460.

[10] P. Berman, M. Karpinski. On Some Tighter Inapproximability Results (Extended Abstract). In *Proc. 26th International Colloquium on Automata, Languages and Programming, ICALP*, 1999, Lecture Notes in Computer Science, 1644, pp. 200–209.

[11] P. Brucker, T. Hilbig, and J. Hurink. A Branch and Bound Algorithm for a Single-Machine Scheduling problem with Positive and Negative Time-Lags. *Discrete Applied Mathematics*, vol. 94, 1999, pp. 77–99.

[12] M. Chlebík and J. Chlebíkova. Approximation Hardness of Optimization Problems in Intersection Graphs of d-Dimensional Boxes. In *Proc. of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms* 2005, pp. 267–276.

[13] A. Dan, P. Shahabuddin, and D. Sitaram. Channel Allocation Under Batching and VCR Control in Movie-On-Demand Servers, IBM Research Report RC19588, May 1994.

[14] M. Dell'Amico. Shop Problems with Two machines and Time Lags. *Operations Research*, vol. 44, no. 5, 1996, pp. 777–787.

[15] G. Hoyle. Distance Learning on the Net. `http://www.hoyle.com/distance.html`.

[16] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, 1979.

[17] F. Gavril. Algorithms for Minimum Coloring, Maximum Clique, Minimum Coloring by Cliques, and Maximum Independent Set of a Chordal Graph. *SIAM J. Computing*, vol. 1, No. 2, 1972, pp. 180–187.

[18] M. Golumbic. Algorithmic Graph Theory and Perfect Graphs. Academic Press, 1980.

[19] J. R. Griggs, and D. B. West. Extremal Values of the Interval Number of a Graph. *SIAM J. Algebraic and Discrete Methods*, 1980, vol. 1, No. 1, pp. 1–7.

[20] A. Gyárfás. On the chromatic number of multiple interval graphs and overlap graphs. *Discrete Math.* 55 (1985), 161–166.

[21] A. Gyárfás and D. B. West. Multitrack Interval Graphs. *Congr. Numer.* 109 (1995), 109–116.

[22] L. A. Hall. Approximability of Flow Shop Scheduling. *Mathematical Programming*, 1998, vol. 82, pp. 175-190.

[23] L. A. Hall, and D. B. Shmoys. Approximation Algorithms for Constrained Scheduling Problems. In *Proc. of the IEEE 30th Annual Symposium on Foundations of Computer Science*, 1989, pp. 134–139.

[24] M. M. Halldórsson, S. Rajagopalan, H. Shachnai, and A. Tomkins. Scheduling Multiple Resources. Manuscript, 1999.

[25] M. M. Halldórsson, K. Yoshihara. Approximation Algorithms for Maximum Independent Set Problem on Cubic Graphs. In *Proc. of Int'l Symposium on Algorithms and Computation, ISAAC '95*, LNCS 1004, 152–161.

[26] C. A. J. Hurkens, and A. Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM J. Discrete Math.*, vol. 2, 1989, pp. 68–72.

[27] K. Jansen, R. Solis-Oba, and M. Sviridenko. Makespan Minimization in Job Shops: A Polynomial Time Approximation Scheme. In *Proc. of the 31th Annual ACM Symposium on Theory of Computing*, 1999, pp. 394–399.

[28] D. Karger, C. Stein, and J. Wein. Scheduling Algorithms. In *Algorithms and Theory of Computation Handbook*, CRC Press, 1998.

[29] A. V. Kostochka and D. B. West. Every outerplanar graph is the union of two interval graphs. *Congr. Numer.* 139 (1999), 5–8.

[30] N. Kumar and N. Deo. Multidimensional interval graphs. *Congr. Numer.* 102 (1994), 45–56.

[31] L. Lewin-Eytan, J. Naor and A. Orda. Routing and admission control in networks with advance reservations", In *Proc. 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, 2002, pp. 215-228.

[32] M. Y. Y. Leung, C. S. Lui, L. Golubchik. Use of Analytical Performance Models for System Sizing and Resource Allocation in Interactive Video-on-Demand Systems Employing Data Sharing Techniques". *IEEE Trans. Knowl. Data Eng*, 14(3), 2002, pp. 615–637.

[33] L. Lovász. On the Shannon capacity of a graph". *IEEE Transactions on Information Theory*, vol. 25, 1979, pp. 1-7.

[34] C. Martin, P. S. Narayanan, B. Ozden, R. Rastogi, and A. Silberschatz. The Fellini Multimedia Storage Server. In *Multimedia Information Storage and Management*, Kluwer Academic Publishers, 1996.

[35] A.J. Orman, and C.N. Potts. On the Complexity of Coupled-Task Scheduling. *Discrete Applied Mathematics*, vol. 72, 1997, pp. 141–154.

[36] C. H. Papadimitriou, and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Computer and System Sciences*, vol. 43, 1991, pp. 425–440.

[37] A. H. G. Rinnooy Kan. Machine Scheduling Problems. Martinus Nijhoff, The Hague, 1976.

[38] D. Rotem, and S. Seshadri. Analysis of Disk Arm Movement for Retrieval of Large Objects. In *Proc. of Principles of Database Systems*, 1992.

[39] E. Hazan, S. Safra, and O. Schwartz. On the Hardness of Approximating k-Dimensional Matching. Electronic Colloquium on Computational Complexity, TR03-020, 2003.

[40] E. R. Scheinerman and D. B. West. The interval number of a planar graph – three intervals suffice. *J. Combin. Theory (B)* 35 (1983), 224–239.

[41] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding Bounds for Applications with Limited Independence. *SIAM J. Discrete Math.*, vol. 6, 1995, pp. 223–250.

[42] R. D. Shapiro. Scheduling Coupled Tasks. *Naval Research Logistics Quarterly*, vol. 27, 1980, 489–498.

[43] D. B. Shmoys, C. Stein, and J. Wein. Improved Approximation Algorithms for Shop Scheduling Problems. *SIAM J. Computing*, vol. 23, 1994, pp. 617–632.

[44] W. T. Trotter, Jr. and F. Harary. On Double and Multiple Interval Graphs. *Journal of Graph Theory*, vol. 3, 1979, pp. 205–211.

[45] D. B. West, and D.B. Shmoys. Recognizing Graphs with Fixed Interval Number is NP-Complete. *Discrete Applied Mathematics*, vol. 8, 1984, pp. 295–305.

[46] P. S. Yu, J. L. Wolf, and H. Shachnai. Design and Analysis of a Look_ahead Scheduling Scheme to Support Pause-Resume Video-on-Demand Applications. In *ACM Multimedia Systems Journal*, vol. 3, 1995, pp. 137–149.