Scheduling Split Intervals

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

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 $t \geq 1$, a positive weight, w_j, and two jobs are in conflict if any of their segments 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.

In a single machine environment, 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, based on a novel *fractional* version of the Local Ratio technique. Previously, the problem was considered only for proper union graphs, a restricted subclass of t-interval graphs, and the approximation factor achieved was $(2^t - 1 + 1/2^t)$. A bi-criteria polynomial time approximation scheme (PTAS) is developed for the subclass of *t-union* graphs.

In the online case, we consider uniform weight jobs that consist of at most two segments. We show that when the resulting 2-interval graph is *proper*, a simple greedy algorithm is 3-competitive, while any randomized algorithm has competitive ratio at least 2.5. For general instances, we give a randomized $O(\log^2 R)$-competitive (or $O((\log R)^{2+\epsilon})$-competitive) algorithm, where R is the known (unknown) ratio between the longest and the shortest segment in the input sequence.

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 segments, for some $t \geq 1$, and a positive weight, w_j; two jobs are in conflict 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 intersect if and only if some segment in the interval corresponding to u intersects with some segment in the interval corresponding to v. Note that 1-interval graphs are precisely interval graphs (an example is given in Figure 1). Thus, for a given instance of our problem, we seek to find a *maximum weight independent set* (MWIS) in the resulting weighted t-interval graph, that is, 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., [33, 9]). 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.

Suppose that a client starts viewing a program at time t_0. At time t_1 the client takes a break, and resumes

[*]Computer Science Dept., Technion, Haifa 32000, Israel. E-mail: reuven@cs.technion.ac.il.

[†]Computer Science Dept., University of Iceland, IS-107 Reykjavik, Iceland, and Iceland Genomics Corp. E-mail: mmh@hi.is.

[‡]Computer Science Dept., Technion, Haifa 32000, Israel. E-mail: *naor@cs.technion.ac.il.*

[§]Computer Science Dept., Technion, Haifa 32000, Israel. Currently on leave at Bell Laboratories, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974. E-mail: hadas@research.bell-labs.com.

[¶]Computer Science Dept., Technion, Haifa 32000, Israel. E-mail: csira@cs.technion.ac.il.

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 experimental studies, in which VCR-like operations can be used by the clients (see [9, 13, 33, 46]); however, these studies focus on the efficient use of system resources while supporting such operations, rather than the scheduling problem.

Linear Resource Allocation. Another application is linear resource allocation [25]. 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 [37]. A linear array network 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. A scheduler must decide when to schedule requests, where each request may comprise 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 a subclass of 2-interval graphs, known as *2-union* graphs (See Section 2.1.)

Genomic Sequence Similarity. Bafna *et al.* [5] consider determining the similarity between genetic sequences under large-scale mutational operations including reversal and transposition. The problem is modeled as that of determining a maximum weight independent set in an intersection graph of axis-parallel boxes: the boxes are in a t-dimensional space, where t is the number of sequences. A pair of boxes is independent (or non-adjacent in the graph) if their projections in all t axes are disjoint. The non-negative weight of a box corresponds to the similarity of the substrings derived from their local alignment.

Computational Geometry. This 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, a subclass of t-interval graphs.

1.2 Our Results.

We give a comprehensive study of the MWIS problem in t-interval graphs. In Section 2, we show that MWIS is APX-hard even on highly-restricted instances, namely, on $(2,2)$-union graphs. Our main result (in Section 3) 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. (This technique was first developed [7] and later extended by [6, 8].) Previously, the problem was considered only on proper union graphs [5], a restricted subclass of t-interval graphs, and the approximation factor achieved was $(2^t - 1 + 1/2^t)$. Note that our approximation factor is almost the best possible, given that any graph G can be represented as a $\lceil(\Delta + 1)/2\rceil$-interval graph [20] (where Δ is the maximum degree), and that the maximum independent set problem cannot be approximated better than within a factor of $\Delta/2^{O(\sqrt{\log \Delta})}$ [43]. The approximation factor can also be argued to be within a constant of best currently possible, since the problem properly includes the MWIS problem in $(t+1)$-claw free graphs (see Section 2), which is not known to be approximable within $t/2$ in polynomial time.

For the class of t-union graphs, we develop (in Section 3.1) a *bi-criteria* PTAS. Given a value $T_{\mathcal{O}}$ and $\epsilon > 0$, our scheme finds a subset of intervals of optimal weight and a schedule where each interval is delayed by at most $\epsilon T_{\mathcal{O}}$, assuming that there exists an optimal solution, whose latest completion time is $T_{\mathcal{O}}$.

In the online case, we consider the MIS (the *unweighted* version) problem on 2-interval graphs. We show (in Section 4) that when the graph is *proper*, any randomized algorithm has competitive ratio at least 2.5; a simple greedy algorithm is 3-competitive. For general instances, we distinguish between two cases. Let R denote the ratio between the longest and the shortest segment in the input sequence. When R is known we employ the *bounded capacity* approach of [1] to obtain a simple 8-competitive algorithm for inputs that consist of small number of segment lengths. This algorithm is used as a procedure to yield an $O(\log^2 R)$-competitive randomized algorithm for inputs with arbitrary segment lengths. When R is unknown in advance we use the method of *randomized virtual selection* of [32] for developing an $O(\log R)^{2+\epsilon})$-competitive algorithm.

Our results contain two technical contributions. Our first contribution is a *fractional* extension of the Local Ratio technique. This enables us to apply the technique for rounding a fractional solution obtained for an LP relaxation of our problem. We expect that this non-standard use of the Local Ratio technique will

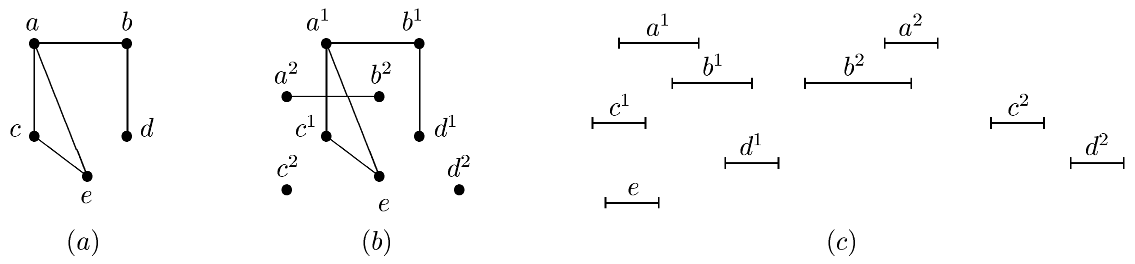

Figure 1: An example of a 2-interval graph (a), corresponding interval (segment intersection) graph (b), and interval system (c).

find more applications. Our second contribution (in Theorem 2.2) is a bound on the *inclusive inductiveness* of a weighted t-interval graph. As a corollary, we extend the best bound known on the chromatic number of t-interval graphs of Gyárfás [21]. Our bound can be shown to be asymptotically optimal.

1.3 Related Work.

We briefly mention several works that are related to ours.

Split interval graphs. Many NP-hard problems including MIS [16, 19] can be solved efficiently in interval graphs. Split interval graphs have a long history in graph theory [44, 20, 38, 45], and more recently union graphs have been studied under the name of *multitrack* interval graphs [31, 22, 30]. We mention some of the main results. For any fixed $t \geq 2$, determining whether a given graph is a t-interval graph is NP-complete [45], and so is determining if a graph is a 2-union graph [22]. 2-union graphs contain trees [44, 31] and more generally all outerplanar graphs [30], while 3-interval graphs contain the class of planar graphs [38]. Graphs of maximum degree Δ are $\lceil \frac{1}{2}(\Delta + 1) \rceil$-interval graphs [20].

Coupled-tasks and flow shop scheduling. The problem of scheduling 2-intervals (known as *coupled-task scheduling*) was considered in the area of offline machine scheduling with the objective of minimizing the *makespan* (see e.g. [34, 41]). 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-leg problems*) were studied, e.g., in [36, 14, 12].

Any instance of our scheduling 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 3.1.) In general, the flow shop problem, where the objective is to minimize the makespan, is NP-complete even on three machines ([15]). 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 ([39, 42]). When m is fixed (but arbitrary) Hall [23] gave a PTAS for this problem.

Online scheduling of intervals. Lipton and Tomkins [32] considered the problem of online scheduling of intervals on a single machine (resource), where the objective is to maximize the resource *utilization*. They gave an $O(\log R)$-competitive ($O((\log R)^{1+\epsilon})$-competitive) randomized algorithm for some $\epsilon > 0$, where R is the (unknown) ratio of longest to shortest interval. Later works [17, 18] consider a variant of the problem, where each interval (job) offers a slack, i.e., the maximal possible delay from the time it arrives until it is scheduled.

Call admission. Interval scheduling can be viewed as a call admission problem on a line, where the objective is to maximize the number of accepted calls. Awerbuch et al. [4] showed a lower bound of $\Omega(\log R)$ on the competitive ratio of any algorithm for calls of arbitrary duration on a *single* link; R is the ratio between the longest and shortest possible durations. Their proof implies a lower bound of $\Omega(\log R)$ for online scheduling of (non-split) intervals, where R is the ratio between the longest and shortest intervals. This lower bound carries over to online scheduling of split intervals.

2 Preliminaries

2.1 Definitions and Notation.

Let $\mathcal{I}$ be a collection of segments (or intervals) on the real line partitioned into disjoint *groups* containing at most t segments, where $t \geq 1$. A t-interval graph $G = (V, E)$ is the intersection graph of the groups of segments. Each vertex in V corresponds to a group of segments, and $(u, v) \in E$ if one of the segments belonging to the group of u intersects some segment belonging to the group of v. We call a vertex in a t-interval graph a *split interval.* Given a t-interval graph, we assume that each vertex can be mapped to a set of segments, i.e., we can say that a segment I belongs to a vertex v and denote it by (v, I). 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 jth segment of v never intersect for $1 \leq i, j \leq t$, and $i \neq j$. 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. In an (a, b)-union graph, all x-segments are of length a and y-segments of length b.

Finally, in 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, $\{v\} \cup N(v)$.

2.2 Hardness Results. Interval graphs are easy to solve exactly since they always contain a vertex whose neighborhood is a clique. In general t-interval graphs, this property fails strongly, as stated in our next result.

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

Proof. For a given $n \geq 2$, let $k = \lfloor (\sqrt{4n+1} - 1)/2 \rfloor$. We show how to construct a 2-interval graph, in which every vertex has k independent neighbors. We construct the graph from $(k + 1)$ subsets of intervals; each subset consists of k intervals, and each interval is composed of two segments. We denote the jth interval in subset i by $I_{i,j}$.

The graph is constructed as follows. Proceeding from left to right, we place under the intervals of subset i, $I_{i,1}, \ldots, I_{i,k}$ the ith intervals of subsets $1, \ldots, k + 1$, i.e., $I_{1,i}, \ldots, I_{k+1,i}$, excluding $I_{i,i}$. This is repeated for $i = 1, \ldots, k$. Finally, under the intervals of the $(k + 1)$-th subset, we place the intervals $I_{i,i}$, $1 \leq i \leq k$ (see Figure 2).

Thus, we get that any interval $I_{i,j}$ with $i \neq j$, intersects k non-intersecting intervals of subset j, and $I_{i,i}$ intersects k non-intersecting intervals of subset $k+1$.

Note that since $k(k + 1) \leq n$, we may have some remaining intervals, which are not contained in any subset. We can place each such interval I under any of the subsets i, $1 \leq i \leq k + 1$, providing that interval k independent neighbors. ∎

We can modify the above construction to hold for 2-union graphs.

We now give a hardness result for a highly restricted class of proper 2-union graphs.

Theorem 2.1. *The MWIS problem is APX-hard on $(2, 2)$-union graphs.*

The omitted proof proceeds by embedding degree-3 graphs in the plane as $(2, 2)$-union graphs. Since the MWIS problem is APX-hard on degree-3 graphs [11, 26] the theorem then follows.

Unit segments are segments of unit size whose start points are integral. Let $S = \{1, 2, \ldots, n\}$ and C be a collection of subsets of S. The *k-set packing* problem is that of finding a maximum cardinality sub-collection C' of C such that the intersection of any two sets in C' is empty. It properly contains the *k-dimensional matching* problem.

Lemma 2.1. *The k-set packing problem is equivalent to the MWIS problem in the special class of k-interval graphs of unit segments.*

Proof. There is a bijective mapping between unit segments and the set S, with $[i, i + 1)$ mapping to i etc. Thus, there is a bijective mapping between sets of up to k elements from S and sets of up to k unit segments. ∎

Similarly, the k-dimensional matching problem (k-DM) is equivalent to the MWIS problem in the special class of k-union graphs of unit segments. In spite of considerable research, the best approximation ratio known for k-dimensional matching is still $k/2 + \epsilon$ [27]. The 2-set packing problem is equivalent to the polynomial solvable Edge Cover problem, while 3-DM is APX-hard [35].

Corollary 2.1. *MWIS in $(1, 1)$-interval graphs is polynomial solvable. MWIS in $(1, 1, 1)$-union graphs is APX-hard.*

The correspondence of $(1, 1)$-union graphs to line graphs of bipartite graphs, and the resulting polynomial solvability of MWIS, was shown by Halldórsson *et al.* [25].

2.3 Structural Properties. Recall that the *inductiveness* of a graph G is defined as $D(G) = \max_{H \subset G} \min_{v \in V(H)} d(v)$. The weighted analog is the *inclusive inductiveness* of G. Let $d^+(v) = \sum_{u \in N[v]} w(u)$ denote the *inclusive degree* of $v \in V$, and $\delta^+(G)$ be the minimum inclusive degree of G. The *inclusive inductiveness* of G is given by $D^+(G) = \max_{H \subseteq G} \delta^+(H)$. Finally, the *weighted clique number* of G is given by $\omega(G) = \max_{C \subseteq G} \sum_{v \in C} w(v)$, where C is a clique. In the following we derive a bound on the inclusive inductiveness of t-interval graphs.

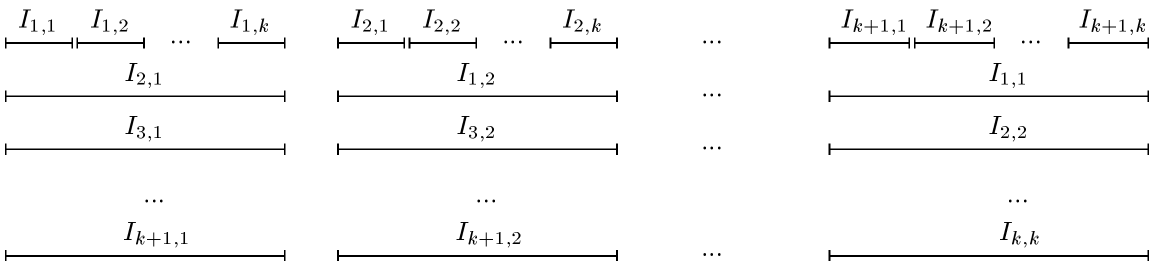

Figure 2: An example of a 2-interval graph, in which every vertex has k independent neighbors.

THEOREM 2.2. *Let G^* be a weighted t-interval graph and let G be the underlying interval graph, i.e. the intersection graph of the segments in G^*. Then, $D^+(G^*) \le 2t \cdot \omega(G)$.*

Proof. Let S^* be an arbitrary subgraph of G^*, and S the corresponding induced subgraph of G. Since each vertex in S^* corresponds to at most t vertices in S, $|V(S)| \le t \cdot |V(S^*)|$, and since each edge in S^* corresponds to one or more edges in S, $W(S^*) \le W(S)$. Thus,

$$\delta^+(S^*) \le \frac{W(S^*)}{|V(S^*)|} \le t\frac{W(S)}{|V(S)|}.$$

If we remove minimum-degree vertices one by one from S, each has inclusive degree at most $D^+(S)$. The weight of the subgraph is at most twice this sum, i.e., $W(S) \le 2D^+(S)$. Since S is an interval graph, $D^+(S) = \omega(S) \le \omega(G)$. Thus,

$$\delta^+(S^*) \le t\frac{W(S)}{|V(S)|} \le 2t \cdot D^+(S) \le 2t\omega(G).$$

This holds for any subgraph S^* of G^*, hence the theorem follows from the definition of inductiveness. ∎

The above gives a $2t$-approximation for coloring t-interval graphs via a greedy algorithm. Gyárfás [21] 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. Our new bound (with slight improvements in the unweighted case) replaces $\omega(G^*)$ by $\omega(G)$ in the latter expression, where G is the underlying interval graph.

COROLLARY 2.2. *A greedy algorithm colors G^* using $2t(\omega(G) - 1)$ colors.*

3 The Offline Case

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 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.

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

Unfortunately, it is not clear how to optimize in the above over all cliques in a t-interval graph. 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 (v, I) such that the intersection of $((v, I)|v \in \mathcal{C})$ is non-empty. We now further relax the independent set problem and consider only interval cliques. For each vertex $v \in V$ and segment $I \in v$, $x(v, I)$ denotes the value of segment I.

$$\boxed{\begin{array}{lrcl} \text{(P)} \quad \text{maximize} & \mathbf{w} \cdot \mathbf{x} & \multicolumn{2}{l}{\textit{subject to}:} \\ \text{for each interval clique } \mathcal{C}: & \sum_{(v,I) \in \mathcal{C}} x(v, I) & \le & 1 \\ \text{for each } v \in V \text{ and } I \in v: & x(v, I) - x(v) & \ge & 0 \\ \text{for each } v \in V \text{ and } I \in v: & x(v), x(v, I) & \ge & 0 \end{array}}$$

The heart of our rounding algorithm is the following lemma. In fact, it can be viewed as a fractional analog of Theorem 2.2.

LEMMA 3.1. *Let $\mathbf{x}$ be a feasible solution to (P). Then, there exists a vertex $v \in V$ satisfying:*

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

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* average 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" (v,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, 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,

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

Since

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

we get that

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

Hence, there exists a vertex v satisfying

$$\sum_{u\in N[v]} y(u,v) \le 2t\cdot x(v). \tag{3.1}$$

If we factor out $x(v)$ from both sides of (3.1) 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 3.2. *Let* $\mathbf{x}$ *be a feasible solution vector to* (P). *Let* $\mathbf{w_1}$ *and* $\mathbf{w_2}$ *be a decomposition of the weight vector* $\mathbf{w}$ *such that* $\mathbf{w} = \mathbf{w_1} + \mathbf{w_2}$. *Suppose that* $\mathbf{y}$ *is a feasible integral solution vector to (P) satisfying:* $\mathbf{w_1}\cdot\mathbf{y} \ge r(\mathbf{w_1}\cdot\mathbf{x})$ *and* $\mathbf{w_2}\cdot\mathbf{y} \ge r(\mathbf{w_2}\cdot\mathbf{x})$. *Then,*

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

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. Delete all vertices with non-positive weight. If no vertices remain, return the empty set.
2. Let $v' \in V$ be a vertex satisfying $\sum_{u\in N[v']} x(u) \le 2t$. 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.)
3. Solve the problem recursively using $\mathbf{w_2}$ as the weight vector. Let $\mathcal{I}'$ be the independent set returned.
4. 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 3.1. *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* $w(\mathcal{I}) \ge \frac{1}{2t}\cdot\mathbf{w}\cdot\mathbf{x}$

Proof. The proof is by induction on the number of recursive calls. At the basis of the recursion, the independent set returned is optimal (and hence a $2t$-approximation), since no vertices remain. Clearly, the first step in which vertices of non-positive weight are deleted cannot decrease the above RHS. We now prove the inductive step. Let $\mathbf{y}$ and $\mathbf{y}'$ be the indicator vectors of the sets $\mathcal{I}$ and $\mathcal{I}'$, respectively. Assume that $\mathbf{w_2}\cdot\mathbf{y}' \ge (1/2t)\cdot\mathbf{w_2}\cdot\mathbf{x}$. Since $w_2(v') = 0$, it also holds that $\mathbf{w_2}\cdot\mathbf{y} \ge (1/2t)\cdot\mathbf{w_2}\cdot\mathbf{x}$. From Step (4) of the algorithm it follows that at least one vertex from $N[v']$ belongs to $\mathcal{I}$. Hence, $\mathbf{w_1}\cdot\mathbf{y} \ge (1/2t)\cdot\mathbf{w_1}\cdot\mathbf{x}$. Thus, by Lemma 3.2, it follows that

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

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

Extension. We note that our approximation results can be generalized to k machines. That is, the goal is to find a maximum weight k-colorable subgraph of a t-interval graph. The approximation factor obtained for this case is $2t+1$.

3.1 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. More specifically, 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 equals 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, which equals 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 $L_{max} = \max_j L_j$. Note that the q_j's are negative. Given that the latest completion time of the optimal solution is $T_{\mathcal{O}}$, we define $\tilde{d}_j = d_j - T_{\mathcal{O}}$. Then for any j, $\tilde{d}_j \leq 0$. By setting $q_j = -\tilde{d}_j$ we get that all the processing times are positive. The objective is to minimize the maximum lateness, given by $L_{max} = \max_j\{C_j - d_j + T_{\mathcal{O}}\}$. Note that since in an optimal schedule there are no "late" jobs, the minimal lateness is $L^*_{max} = T_{\mathcal{O}}$.

In the following we use some ideas from [28, 24, 29]. 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., [23]). 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 then scale all parameter values as follows. 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 $\epsilon/(2t)$. Thus, the number of vectors $(p_{j,1}, \ldots p_{j,2t+1})$ is $\prod_{i=1}^{t} c_i (2t/\epsilon)^t$.

We now summarize the steps of our scheme, which gets as parameter the value of $T_{\mathcal{O}}$ and some $\epsilon' > 0$. (i) Guess $\mathcal{O}$, the number of intervals scheduled by OPT; (ii) Guess the subset $S_{\mathcal{O}}$ of $\mathcal{O}$ intervals of maximal weight, scheduled by OPT. (iii) Using a PTAS for minimizing the makespan in the flow shop instance of $S_{\mathcal{O}}$, 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 $\epsilon/2t$ to the release times; also, each break time may delay the optimal completion time by $\epsilon/2t$, therefore, taking $\epsilon' = \epsilon/2t$ we guarantee that the delay of each interval is at most $(1 + \epsilon)$ times $T_{\mathcal{O}}$. Finally, we add $T_{\mathcal{O}} = L^*_{max}$ to each due date; thus, the maximum lateness of any job in our schedule equals to ϵC^*_{max}.

For the complexity of our scheme, note that step (i) takes $O(n)$. The number of possible guesses of $S_{\mathcal{O}}$ is $O(n^{(t^t \prod_{i=1}^{t} c_i)/\epsilon^t})$. (In each guess of a set of vectors representing a subset of intervals, we take the subset whose weight is maximal.) This is multiplied by the complexity of the PTAS for flow shop. We summarize in the next result.

THEOREM 3.2. *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 $\epsilon T_{\mathcal{O}}$.*

4 The Online Case

In the online version of our problem, the set of input intervals is not known to the scheduler in advance. We focus here on the case where the jobs are uniform, and each job consists of at most two segments. Thus, our goal is to select a subset of non-overlapping jobs of maximum size, i.e., we deal with the MIS problem in a 2-interval graph. We assume that all the endpoints of the segments that belong to a 2-interval are known upon its arrival. For simplicity, assume that the shortest segment has length 1. Let k be the claw number of the graph.

Consider the Greedy algorithm that schedules every arriving interval if it does not conflict with any of the previously scheduled intervals. Recall that R is the ratio between the longest and the shortest segment in the input sequence. We omit the proof of the next result.

THEOREM 4.1. *The Greedy algorithm is strongly* $min\{2R+1, k\}$*-competitive.*

4.1 Proper graphs. When the graph is proper, the above greedy algorithm has competitive ratio 3. In the following we show that this is almost the best possible, even when we use randomization.

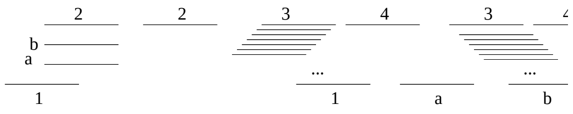

Figure 3: Construction for a randomized lower bound

THEOREM 4.2. *The performance ratio of any randomized online MIS algorithm on proper 2-interval graphs is at least 2.5.*

Proof. We construct a family graph, whose interval representation is shown in Figure 3. The graph contains vertices 1 through 4, a and b, and ℓ vertices shown following vertex 3. Here ℓ is a parameter chosen at random from the range 1 to n. The intervals are presented in the order shown. Only the order among 2, a, and b is not known, and is chosen at random.

The optimal solution consists of intervals 2,3, and 4. The reason why the algorithm is bound to obtain a non-optimal solution is three-fold: (i) it must choose interval 1 with significant probability, in order to be competitive on the prefix instance consisting of that interval alone, (ii) when trying to choose interval 2, it may wind up with either a or b, in which case it can include no more intervals, and (iii) it cannot guess with non-trivial probability the value of ℓ, meaning that it is unlikely to add interval 3 and then 4.

Let p_1 be the probability that the algorithm adds interval 1. The competitiveness on the instance consisting of a single interval is then $1/p_1$. The algorithm can only get two intervals if it either selects 2, or selects 3, and three only if it adds both. The probability of selecting interval 3 is only $1/n$, which is negligible. The probability of selecting 2 is $1/3$. It follows that the expected size A of the set obtained by the algorithm is at most

$$E[A] \le p_1 + (1-p_1)(\frac{2}{3}\cdot 1 + \frac{1}{3}\cdot 2) = \frac{4-p_1}{3}.$$

Thus, the performance ratio of the algorithm is the worse of the performance on the single-interval instance and of the instance above, or $\rho = \max(\frac{1}{p_1}, \frac{3}{4/3-p_1/3}) = \max(\frac{1}{p_1}, \frac{9}{4-p_1}) = \frac{5}{2}$. ■

4.2 General split interval graphs

4.2.1 Known segment lengths. Consider the case where R is known in advance. We first present an algorithm for scheduling intervals, in which the first segment is of length 1 and the second of length l, for some $l \ge 1$ ($(1,l)$ intervals). Then we show how this algorithm can be used as a procedure by an online algorithm that accepts as input interval with segments of arbitrary lengths.

Suppose that we can schedule the intervals on two servers (i.e., the capacity of the resource is 2). The algorithm Double Select (DS) proceeds as follows. Upon arrival of an interval I, if there is a resource available for I, we schedule the interval on some available resource; on the other resource, we schedule no interval whose long segment intersects I's long segment. Finally, we choose randomly, with probability 1/2, one of the two servers.

It can be shown that (i) if the length of the first segment is in the range $[1,2]$, and the second segment is in the range $[l/2, l]$, then the algorithm DS is 8-competitive; (ii) if the length of the first segment is in the range $[l/2, l]$, and the second segment is in the range $[1,2]$, then the Greedy algorithm is 5-competitive.

For the general case, assume for simplicity that $R = 2^k$, where $k \ge 1$ is an integer. We partition the intervals to $\lg^2 R$ barrels: the barrel (i,j), $1 \le i,j \le \lg R - 1$. contains the intervals I, in which $2^i \le |I^1| < 2^{i+1}$ and $2^j \le |I^2| < 2^{j+1}$.

Denote by $\mathcal{A}$ the set of barrels in which the first segment is longer, i.e., $\forall I \in \mathcal{A}\ |I^1| > |I^2|$. The algorithm Select-from-Barrel (SB) proceeds as follows. We randomly choose one barrel $(i,j)^*$. If $(i,j)^* \in \mathcal{A}$, we schedule the intervals in this barrel greedily; else we apply the algorithm DS on this barrel. We reject all other intervals.

Note that for the barrels in $\mathcal{A}$, the greedy schedule achieves the competitive ratio 5 from (i), while on the other barrels DS has the ratio 8. Since half of the barrels are in $\mathcal{A}$ we get that the competitive ratio of SB is $6.5 \log^2 R$. Thus, we have shown the following.

THEOREM 4.3. *The algorithm* SB *is* $O(\log^2 R)$ *competitive.*

4.2.2 Unknown segment lengths.

Now we describe an algorithm for the case where R is unknown in advance. Using ideas similar to those of [32], we define the algorithm HalfLengths (HL). For each arriving interval, I, we examine separately the two segments of I, $I^i, i \in \{1,2\}$. The *depth* of I^i, denoted by $depth(I^i)$, is d, if there is a set of segments S_j, $1 \le j < d$, such that (*i*) $S_1, \ldots, S_{d-1}$ intersect with I^i, (*ii*) $|S_j| \le \frac{1}{2}|S_{j-1}|$, $1 < j < d$; (*iii*) S_j, $1 \le j < d$, was "selected", but the resource may be allocated to another segment (satisfying certain properties) before S_j completes. (We say that S_j was *virtually taken.*)

We use for coin tosses a probability distribution derived from the *Riemann zeta function* (see, e.g., [40]): for a complex number $z \in \mathcal{C}$, $\zeta(z) = \sum_{n \ge 1} \frac{1}{n^z}$; $\zeta(z)$ converges when $|z| > 1$, thus we choose $z = 1 + \epsilon$ for some small $\epsilon > 0$. Let $c_d = 1/(d^{1+\epsilon}\zeta(1+\epsilon))$, and $p_d = c_d / \prod_{j=1}^{d-1}(1 - p_j)$. Then c_d denotes the probability that a segment in depth d will be taken; p_d is the *conditional* probability that a segment of depth d will be taken, given that $S_1, \ldots, S_{d-1}$ were *not* taken.

We consider scheduling I^i in depth d, only if $|I^i| \le \frac{1}{2}|S_{d-1}|$, in which case I^i is taken with probability p_d, and *virtually* taken with probability $1 - p_d$. This guarantees that if we virtually take an interval I, then we do not consider any other interval, J, that intersects I^i, $i \in \{1,2\}$, unless the segment of J which intersects with I^i is at least twice shorter than I^i.

Finally, we schedule the interval I only if both I^1 and I^2 were scheduled; if one of these segments was only virtually taken, then I is virtually taken, and if any of the segments was rejected, then I is rejected.

THEOREM 4.4. *The competitive ratio of algorithm HalfLengths is* $O((\log R)^{2+\epsilon})$.

Proof. Let O be some interval scheduled by OPT. There are two possible cases: if HL has an opportunity to schedule O, (i.e., HL schedules O with some probability c_O), then we are done, since the maximal depth of any segment is $\log R$; thus, $c_O \ge c^2_{\log R}$; otherwise, HL has an opportunity to schedule some other interval I, that blocks O, i.e., I intersects O and either $|O^1| > \frac{|I^i|}{2}$ or $|O^2| > \frac{|I^i|}{2}$, $i \in \{1,2\}$. But in this case, OPT can schedule at most 5 intervals instead of I. For HL the probability to schedule any interval (if this interval is not blocked) is at least $c^2_{\log R}$, since each unblocked segment is scheduled with probability at least $c_{\log R}$, and scheduling of different segments of the same interval is independent. Therefore, for any interval that belongs to the optimal schedule, the gain of HL is at least $\frac{c^2_{\log R}}{5}$.

Let $OPT(\sigma)$ and $HL(\sigma)$ denote the total gains of OPT and HL, for some input sequence σ, and $OPT(O)$ and $HL(O)$ denote the gain of OPT and HL on the interval O respectively. Then,

$$\begin{aligned} E[HL(\sigma)] &\ge \sum_{O \in OPT(\sigma)} HL(O) \\ &\ge \sum_{O \in OPT(\sigma)} \frac{c^2_{\log R}}{5} OPT(O) = \frac{c^2_{\log R}}{5} OPT(\sigma) \end{aligned}$$

This completes the proof. ∎

Acknowledgments. We thank Yossi Azar for many helpful comments on this paper.

References

[1] R. Adler, and Y. Azar. "Beating the Logarithmic Lower Bound: Randomized Preemptive Disjoint Path and Call Control Algorithms". In *SODA '99*, 1–10.

[2] A. Aggarwal, J. Garay, and A. Herzberg. "Adaptive Video on Demand". In *ESA '95*, 538–553.

[3] B. Awerbuch, Y. Azar, A. Fiat, and T. Leighton. "Making commitments in the face of uncertainty: how to pick a winner almost every time". In *STOC '96*, 519–530.

[4] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén. "Competitive Non-Preemptive Call Control". In *SODA '94*, 312–320.

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

[6] 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.

[7] 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.

[8] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. "A Unified Approach to Approximating Resource Allocation and Scheduling". In *STOC '00*, 735–744.

[9] P. Basu, A. Narayanan, R. Krishnan, and T.D.C. Little. "An Implementation of Dynamic Service Aggregation for Interactive Video Delivery". In *SPIE '98.*

[10] 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.

[11] P. Berman, and T. Fujito. "Approximating Independent Sets in Degree 3 Graphs". In *WADS '95*, LNCS 955, 449–460.

[12] P. Brucker, T. Hilbig, and J. Hurink. "A Branch and Bound Algorithm for a Single-Machine Scheduling

problem with Positive and Negative Time-Lags". In *Discrete Applied Mathematics*, vol. 94, 1999, pp. 77–99.

[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". In *Operations Research*, vol. 44, no. 5, 1996, pp. 777–787.

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

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

[17] S. Goldman, J. Parwatikar, and S. Suri. "On-line Scheduling with Hard Deadlines". In *WADS '97*, 258–271.

[18] M. Goldwasser. "Patience is a Virtue: The Effect of Slack on Competitiveness for Admission Control". In *SODA '99*, 396–405.

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

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

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

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

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

[24] L.A.Hall, and D.B. Shmoys. "Approximation Algorithms for Constrained Scheduling Problems". In *FOCS '89*, 134–139.

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

[26] M. M. Halldórsson, K. Yoshihara. "Approximation Algorithms for Maximum Independent Set Problem on Cubic Graphs", In *ISAAC '95*, LNCS 1004, 152–161.

[27] 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.

[28] K. Jansen, R. Solis-Oba, and M. Sviridenko. "Makespan Minimization in Job Shops: A Polynomial Time Approximation Scheme". In *STOC '99*, 394–399.

[29] D. Karger, C. Stein, and J. Wein. "Scheduling Algorithms". Algorithms and Theory of Computation Handbook, CRC Press, 1997.

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

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

[32] R. Lipton and A. Tomkins. "Online Interval Scheduling". In *SODA '94*, 302–311.

[33] 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.

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

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

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

[37] D. Rotem, and S. Seshadri. "Analysis of Disk Arm Movement for Retrieval of Large Objects". In *PODS '92.*

[38] 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.

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

[40] R. Sedgewick and P. Flajolet. An Introduction to the Analysis of Algorithms. Addison-Wesley, 1996.

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

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

[43] L. Trevisan. "Non-approximability results for optimization problems on bounded degree instances". In *STOC '01*, 453–461.

[44] W.T. Trotter, Jr. and F. Harary. "On Double and Multiple Interval Graphs". In *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". In *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.