Strip Graphs: Recognition and Scheduling[⋆]

Magnús M. Halldórsson and Ragnar K. Karlsson

Dept. of Computer Science
University of Iceland
IS-107, Reykjavik, Iceland
{mmh, rkk1}@hi.is

Abstract. We consider the class of strip graphs, a generalization of interval graphs. Intervals are assigned to rows such that two vertices have an edge between them if either their intervals intersect or they belong to the same row. We show that recognition of the class of strip graphs is $\mathcal{NP}$-complete even if all intervals are of length 2. Strip graphs are important to the study of job selection, where we need an equivalence relation to connect multiple intervals that belong to the same job.

The problem we consider is Job Interval Selection (JISP) on m machines. In the single-machine case, this is equivalent to Maximum Independent Set on strip graphs. For m machines, the problem is to choose a maximum number of intervals, one from each job, such that the resulting choices form an m-colorable interval graph. We show the single-machine case to be fixed-parameter tractable in terms of the maximum number of overlapping rows. We also use a concatenation operation on strip graphs to reduce the m-machine case to the 1-machine case. This shows that m-machine JISP is fixed-parameter tractable in the total number of jobs.

1 Introduction

1.1 Strip Graphs

Strip graphs are a generalization of interval graphs. They are defined by an interval graph combined with an equivalence relation on the intervals. For example, we can map different jobs assigned to a machine as an interval graph and allow each equivalence class to be made up of jobs belonging to the same user. In this case, two vertices have an edge between them if either their intervals intersect or they belong to the same equivalence class. We can look at this as the union of two graphs: an interval graph and a graph of equivalence classes, representable as a set of disjoint cliques. Note that we can represent a set of disjoint cliques as an interval graph as well: if we enumerate the cliques, the interval $[i-1, i)$ can be assigned to each vertex of clique i. By using this representation we see that any such graph can be defined by taking the union of two interval graphs, one of which is a set of disjoint cliques. Such a graph is called a *strip graph*, which refers to the *rectangle graph* representation of graphs formed by the union of two

[⋆] Supported by a grant from RANNÍS — The Icelandic Centre for Research.

interval graphs, defined by Bar-Yehuda *et al.* in [BYHN$^+$06]. In this representation, each vertex is represented on the 2-dimensional plane by an axis parallel rectangle, the vertical side of which is of length 1. The two interval graphs are formed by projecting the sides of the rectangles onto each axis, such that two vertices are adjacent if their corresponding intervals in either of these projections intersect. Throughout this paper, we will assume that all intervals are half-open (open on top).

Gyárfás and West defined a t-track (t-union) graph to be the edgewise union of t interval graphs. Since strip graphs can be represented by two interval graphs, they form a subclass of the 2-union graphs. In [GW95], it was shown that recognizing 2-union graphs is $\mathcal{NP}$-complete. Since we are dealing with only a restricted class of the 2-union graphs, we are interested in seeing if recognition is $\mathcal{NP}$-complete for this subclass as well. To form the sharpest bound possible between $\mathcal{P}$ and $\mathcal{NP}$, we will define a graph as a *k-strip graph* if it is representable as a strip graph such that all strips are of length k and each interval has integral start- and endpoints. Clearly, k-strip graphs are a subclass of strip graphs. We will show that the class of 1-strip graphs is precisely equivalent to the class of line graphs of bipartite graphs and is therefore recognizable in polynomial time, but recognition of strip graphs is $\mathcal{NP}$-complete, even when restricted to k-strip graphs, for any $k \geq 2$.

The reduction is from the problem of determining whether a triangle-free cubic graph is Hamiltonian, which was shown in [WS84] to be $\mathcal{NP}$-complete. We prove in Sect. 3 that the removal of an edge e from any triangle-free cubic graph will form a strip graph if and only if the original graph has a Hamiltonian cycle going through edge e.

1.2 Strip Graph Applications

Interval graphs are often used in scheduling, but they only give information in the time dimension. Modeling more complex relations requires more complex structures. One classic scheduling problem is scheduling the maximum number of jobs in a non-conflicting manner on a single machine, given multiple possible run-times for each job. This problem is commonly referred to as the Job Interval Selection Problem (JISP) and has results going as far back as a 1982 paper by Nakajima and Hakimi [NH82]. By considering each possible run-time for each job as a vertex, and defining equivalence classes of intervals contained in the same job, we can model this problem with a strip graph. While JISP and the generalizations we consider in this paper have been recently studied, most of the research looks at JISP as being a problem on sets of intervals. We take a different route by studying the problem from the perspective of strip graphs and using structural observations of strip graphs to gain new insight into the problems.

Since each equivalence class is a clique, any independent set of this strip graph can only contain one interval from each job. Therefore, the maximum independent set of the strip graph is the maximum number of jobs that can be run on each machine. We show in Sect. 4 that finding the maximum independent set of a strip graph is fixed-parameter tractable in the maximum number of jobs with overlapping windows. This idea of job windows is similar to the one used in Chuzhoy *et al.* [COR01], where they showed that the MIS is computable in pseudo-polynomial time if the size of the job window is small in comparison to the size of the job.

A simple extension to this problem is scheduling the maximum number of jobs in a non-conflicting manner on multiple machines. We will show in Sect. 5 how this is reduced to the single-machine case by generating an instance of single-machine JISP involving concatenated copies of the input graph. We also analyze what effect this has on the running time of our JISP algorithm.

2 Preliminaries

2.1 Definitions and Notation

Given a strip graph G, we define G to be the union of two interval graphs, G_1 and G_2, such that G_1 contains only intervals of length 1 and G_2 is a regular interval graph. For any subgraph $H \subset G$, we will use the standard notation $V(H)$ to refer to the vertex set of H. For any set $V' \subset V(G)$, we will use $G(V')$ to refer to the subgraph of G induced by V'.

Let v be a vertex in a strip graph. We need to know three properties to define v's position. We define, for any vertex v, ρ_v to be an integer such that v is represented by the interval $[\rho_v, \rho_v + 1)$ in G_1. We define s_v and f_v to be the start- and endpoint of v's interval — that is, values such that v is represented by the interval $[s_v, f_v)$ in G_2. When represented as a rectangle graph, we can think of ρ_v as being the "row" containing the rectangle v. We can then define v as being adjacent to another vertex w if either $\rho_v = \rho_w$ — in which case we say v is a *1-neighbor* of w — or $[s_v, f_v) \cap [s_w, f_w) \neq \emptyset$, in which case there are four options, outlined below.

Assume $[s_v, f_v) \cap [s_w, f_w) \neq \emptyset$. We say v is a *left neighbor* of w and w is a *right neighbor* of v if $s_v < s_w < f_v < f_w$. Otherwise, we say v is an *internal neighbor* of w and w is an *external neighbor* of v if $s_w \leq s_v < f_v \leq f_w$. We note that if w and v are identical intervals, then v can be considered both an internal and an external neighbor of w. See Fig. 1.

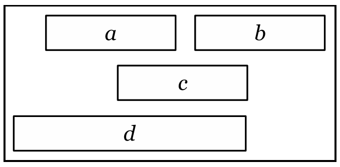

Fig. 1. Vertex a is a 1-neighbor of b, a left-neighbor of c and an internal neighbor of d

3 Hardness of Recognition

3.1 1-Strip Graphs

Remember that a 1-strip graph is defined such that each interval in G_2 is of length 1 and has integral start- and endpoints. The polynomial recognizability of 1-strip graphs follows from the following theorem, mentioned in [HRST99]. The formal proof is quite simple, and the proof is left to the reader. For now, we simply refer to Fig. 2. Since line graphs of bipartite graphs can be recognized in linear time [Leh74], the result follows instantly.

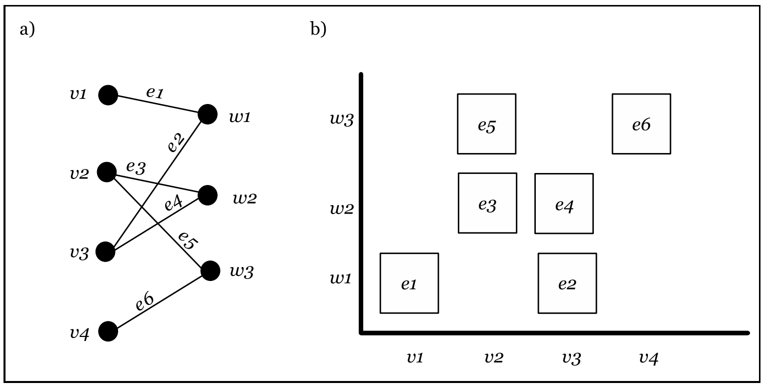

Fig. 2. The 1-strip graph (b) is the line graph of the bipartite graph (a)

Theorem 1. *A graph G is a 1-strip graph if and only if G is the line graph of a bipartite graph.*

3.2 Strip Graphs

West and Shmoys [WS84] showed that the problem of determining whether a triangle-free cubic graph is Hamiltonian is $\mathcal{NP}$-complete. They used this to show that recognizing 2-interval graphs is $\mathcal{NP}$-complete. West also used this to show, with Gyárfás, that recognition of 2-union graphs is $\mathcal{NP}$-complete [GW95]. We now use this problem to show that recognizing strip graphs is $\mathcal{NP}$-complete. What makes our reduction different from the reductions mentioned above is that we have to pay special attention to the lengths of the intervals in the graphs. Neither the reduction from [WS84] nor the reduction from [GW95] applies directly to graphs where one of the intervals is assumed to be of length 1. Additionally, those reductions don't give a boundary between when recognition is in $\mathcal{P}$ and when it is in $\mathcal{NP}$. By additionally considering the length of the intervals of G_2, we can give a restriction that allows recognition in polynomial time, and show that no similar restriction on length is recognizable in polynomial time unless $\mathcal{P} = \mathcal{NP}$.

The following theorem shows that determining whether a triangle-free cubic graph is Hamiltonian reduces to strip graph recognition. Therefore, we can conclude that recognizing strip graphs is $\mathcal{NP}$-complete.

Theorem 2. *A triangle-free cubic graph G is Hamiltonian if and only if there exists an edge e such that $G \setminus \{e\}$ is a strip graph.*

Proof. Assume a graph G is cubic, triangle-free and Hamiltonian. G is a graph on n vertices. Let C be the graph's Hamiltonian cycle, choose an arbitrary starting vertex v_1 and an arbitrary direction for C, and label the remaining vertices in order with the labels $v_2, v_3, \ldots, v_n$. We know that the remaining edges must form a perfect matching, which we denote by M. For any vertex v_i, we denote by $M(v_i)$ the vertex matched to v_i in M. We now remove the edge connecting v_1 to v_n and show that the remaining graph, denoted by G', must be a strip graph.

We begin by forming G'_2. With the removal of (v_n, v_1), C becomes a Hamiltonian path in G', which can be represented with intervals of length 2 by representing each v_i with the interval $[i-1, i+1)$. This defines G'_2. The remaining edges, as we saw above, form a perfect matching, so M can be represented with length 1 intervals in G'_1 by representing each v_i and $M(v_i)$ with the interval $[i-1, i)$. To avoid placing two intervals for each vertex, this operation only happens when v_i is of a lower index than $M(v_i)$. This defines G'_1, and the union of G'_1 and G'_2 clearly forms G'.

Figure 3(b) shows how this transformation works for a triangle-free cubic Hamiltonian graph on 6 vertices after removing the edge (v_6, v_1). The first set of intervals defines G'_2 as a Hamiltonian path, and the second set of intervals defines G'_1 as a perfect matching. Figure 3(c) then shows how this strip graph is represented as a rectangle graph.

For the other direction, assume the removal of some edge (v, w) creates a strip graph, which we again call G'. Note that since G_2 is an interval graph, it must have at least two simplicial vertices — that is, vertices that have a neighborhood covered by at most one clique in G_2. Two of these vertices must be represented by the earliest-ending and latest-starting intervals in G_2. We also know that any vertex in G has a neighborhood covered by one clique in G_1. Therefore, the neighborhoods of the simplicial vertices of G_2 are covered by 2 cliques in G. Since G is triangle-free, these vertices have degree at most 2. Therefore, since v and w are the only two vertices of degree 2, they must be the simplicial (rightmost and leftmost) vertices of G_2.

Let v's neighbor in G'_2 be the vertex z. We know that z's other neighbor in G'_2 cannot be internal unless that neighbor is of degree at most two, so we that neighbor has to be a right neighbor, itself of degree 3. The degree-3 vertices must form a path, which cannot end until it hits a vertex of degree 2, which must then be w. Since w is the vertex with the rightmost endpoint in G'_2, every vertex in G' must appear between v and w. If there exists some vertex p of degree 3 that does not appear on the path between v and w, then it must be an internal neighbor of some vertex on the path (because G'_2 is triangle-free), which means p must be of degree 2, which is a contradiction. We conclude that every degree-3 vertex in

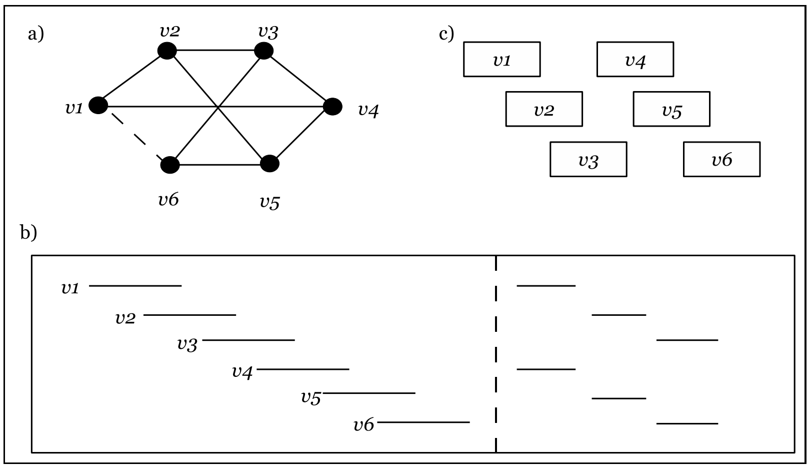

Fig. 3. (a) A cubic triangle-free Hamiltonian graph on 6 vertices. (b) Removing one edge allows us to represent the remaining graph as a strip graph. (c) The rectangle graph representation.

the graph must appear on the path between v and w. That is to say, the path is Hamiltonian.

We note that v_1 was chosen arbitrarily, and therefore any edge of the Hamiltonian cycle can be removed to create G'. For any given vertex v, we know that two of the three edges connected to v are present in any Hamiltonian cycle. Therefore, we have to select two of the edges in v. If the removal of either one creates a strip graph, then G is Hamiltonian. A polynomial-time algorithm to check these two cases in parallel gives a polynomial-time reduction from the Hamiltonian cycle problem on triangle-free cubic graphs to the strip graph recognition problem. We conclude by noting that the proof of the above reduction implies that recognition of k-strip graphs is $\mathcal{NP}$-complete for all $k \geq 2$.

Corollary 3. *Recognition of strip graphs is $\mathcal{NP}$-complete. Additionally, recognition of k-strip graphs is $\mathcal{NP}$-complete for any $k \geq 2$.*

4 Scheduling Applications of Strip Graphs

To see how strip graphs apply to scheduling problems, we first contrast them with t-interval graphs. Consider a set of jobs, each consisting of multiple intervals. These jobs can each be represented by a row in a strip graph. In the t-interval representation, each row would be contracted to a single vertex. This is useful for applications where jobs are split up over multiple intervals, and selecting a job means selecting all of its intervals. In the strip graph representation, on the other hand, each interval is a vertex, which is useful in situations that require access to individual intervals of each job. The classic example of this is the Job

Interval Selection Problem, or JISP, where each job only requires one interval of processing time, and the multiple intervals represent *possible* run-times.

The input to this problem is a set of jobs, called a *request.* Each job is comprised of multiple intervals, each representing a possible run-time for the job. The objective is to run as many jobs as possible such that no two conflict — that is, select at most one interval from each row such that we select the maximum number of intervals possible such that no two overlap. This selection of intervals is referred to as a *schedule* and is preciely equivalent to the Maximum Independent Set (that is, the largest set of mutually non-adjacent vertices, abbreviated MIS) of the corresponding strip graph.

In this section, we will give an exact algorithm for JISP / MIS of strip graphs that runs in exponential time. We will then consider how this algorithm applies to JISP in a multiple-machine environment.

4.1 Maximum Independent Set

The Maximum Independent Set problem is known to be $\mathcal{NP}$-complete for 2-strip graphs [CS96], implying the $\mathcal{NP}$-completeness of the problem on strip graphs. In this section, we will show that finding the maximum independent set is fixed-parameter tractable in the number of overlapping "liveness windows." In the following, assume G is a strip graph on n vertices.

We define $\mathfrak{R}$ to be the partition of G_1 into its connected components. That is, each member of $\mathfrak{R}$ is a row of G. Define $C = \max\{f_v | v \in G\}$. Then the rectangle representation of G is mapped onto an $|\mathfrak{R}| \times C$ grid. For a given subset $R \subset \mathfrak{R}$, we let G^R represent the subgraph of G induced by the rows in R, and for each integer $y \leq C$, we let G_y represent the subgraph of G induced by all vertices ending at or before point y in the interval representation of G_2. That is, for any $R \in \mathfrak{R}$ and $y \leq C$, $G^R_y = G(\{v \in V(G) | v \in R, f_v \leq y\})$.

We store the intermediate results in a matrix M, where $M[R, y]$ is the size of the MIS of the graph G^R_y, for $y \leq C$ an integer, and $R \subset \mathfrak{R}$ a set of rows. At position y, there are two possibilities:

– We can add a vertex v with endpoint $y = f_v$. If this case gives the MIS, then $M[R, f_v]$ is given by $M[R \setminus \{\rho_v\}, s_v] + 1$.
– No vertex with endpoint y is a member of an MIS of G^R_y. If this is the case, then $M[R, y] = M[R, y - 1]$.

We point out that the only interesting values in the above formulation are the start- and endpoints of vertices in G. Since any interval graph can be represented such that all intervals have distinct start- and endpoints, we can assume that this is the case. Therefore, we store all start- and endpoints in an increasing sequence, $y_1, y_2, \ldots, y_{2n}$. Note that since we're only considering start- and endpoints, we can essentially drop the integrality constraints on G_2.

Using this assumption, and the two possibilities above, the dynamic programming relation is given with:

$$M[R, y_i] = \max\{M[R, y_{i-1}], M[R \setminus \{\rho_v\}, s_v] + 1\}$$

where y_i is the endpoint of vertex v (if such a v exists).

To determine the minimum worst-case complexity, we need to determine how many rows need to be checked at each point y_i. For each row r, r is defined to be *live* at all points between the endpoint of its first interval and the endpoint of its last interval. Let $L_t \subset \mathfrak{R}$ be the set of live rows at point t and define $Q = \max_{t \leq C} |L_t|$. Similarly, a row is defined to be *dead* at all points after the endpoint of its last interval. Let $D_t \subset \mathfrak{R}$ be the set of all dead rows at point t. At each start- or endpoint y on the axis, there are two steps to the calculation.

We first compute $M[R \cup D_y, y]$ for all $R \subset L_y$. To understand why we can limit the calculations to this set of rows, consider a row r that is live on the interval $[a, b]$. If $y < a$, then no independent set of G_y includes row r because that row is empty, so r is never included. If, on the other hand, $y > b$, then any independent set of G_y has to consider row r; there is no need to defer the choice to a point further down the axis, because r has no intervals past b. This step is completed in time at most $\mathcal{O}(2^Q)$.

For the second step, consider a vertex v starting at point y. When we reach the endpoint of that vertex, we're going to need to be able to check $M[R' \cup D_{f_v}, y]$ for all $R' \subset L_{f_v}$. To do this, we merely iterate through each subset $R' \subset L_{f_v}$, and set

$$M[R' \cup D_{f_v}, y] = M[(R' \cup D_{f_v}) \cap (L_y \cup D_y), y]$$

this step is completed in time at most $2^{|L_{f_v}|}$. Since each point we consider is the startpoint of at most one vertex, the computation time for step 2 at any point y is at most $\mathcal{O}(2^Q)$.

Since we only consider at most $2 \cdot n$ start-/endpoints of intervals, M can be generated with a total time complexity of $\mathcal{O}(n \cdot 2^Q)$. To generate the MIS from the table M, we merely need to work backwards from C.

Assume the MIS of G is found to be of size k. We then find a vertex v such that $M[L_{f_v} \cup D_{f_v}, f_v] = k$ and $M[(L_{f_v} \cup D_{f_v}) \setminus \{\rho_v\}, s_v] = k - 1$. If we work in decreasing endpoint order, we only need to look at each vertex once, and choose the first one that fits. We add v to the set S and then repeat the procedure for the strip graph $G_{s_v}^{(L_{s_v} \cup D_{s_v}) \setminus \{\rho_v\}}$, which we know has an MIS of size $k - 1$. Using recursion, we get an independent set S of maximum size k in time $\mathcal{O}(n)$. The MIS problem is now clearly fixed-parameter tractable in Q.

Theorem 4. *For a strip graph G, define Q to be the maximum number of live rows overlapping at any point. The above algorithm then finds a maximum independent set of G in time $\mathcal{O}(n \cdot 2^Q)$.*

We conclude with two observations. First, we note that it is easy to generalize this algorithm to the Maximum Weighted Independent Set (MWIS) problem. In that case, each vertex v has a weight w_v, and the objective is to find an independent set such that the sum of the weights of all the vertices in the IS is maximized. The recursive formulation is then given by

$$M[R, y_i] = \max\{M[R, y_{i-1}], M[R \setminus \{\rho_v\}, s_v] + w_v\}$$

where y_i is the endpoint of vertex v (if it exists). The rest of the algorithm then works as described above.

Secondly, we note that a row consisting of only one interval is live at exactly one point. By assuming distinct endpoints for each interval, two rows that each consist of only one interval are never live at the same time. Therefore, our algorithm for MWIS reduces precisely to the classic linear-time DP algorithm for weighted independent set in interval graphs.

5 Multiple Machines

JISP is generalized by assuming that we have m machines running in parallel. The problem then becomes one of finding a maximum-value m-colorable subset of the input graph, which we call an m-schedule. This section will focus on reducing the m machine case to the 1-machine case and analyzing what effect this has on the above algorithm.

Assume we are given as input a strip graph G and asked to calculate the value of JISP for m machines. We define a new graph G^{*m} and show that the value of JISP for 1 machine on G^{*m} is equivalent to the value of JISP for m machines on G. G^{*m} is defined by making m identical copies of G such that the only edges between copies are between all intervals that belong to the same job. This creates a new strip graph, formed by making m identical copies of G's rectangle represenation along the horizontal axis. This transformation is shown in Fig. 4.

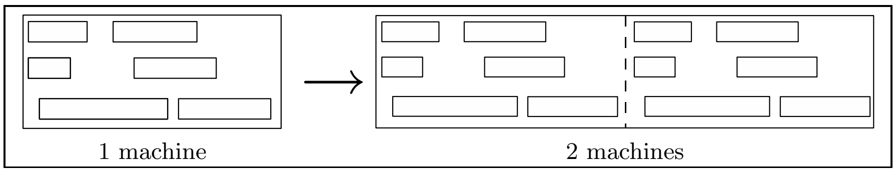

Fig. 4. A request simulated on 2 machines

Theorem 5. *G has a maximum m-schedule of size k if and only if G^{*m} has a maximum independent set of size k.*

Proof. Assume S is a maximum m-schedule and let $k = |S|$. We enumerate the color classes of S with $S_1, S_2, \dots, S_m$. Each S_i is an independent set. G^{*m} is a concatenation of m identical requests (which we can refer to as "subrequests"), with edges between intervals belonging to the same job. Therefore, we can choose the vertices of S_1 for the first subrequest of G^{*m}, the vertices of S_2 for the second subrequest, and so on. Call this set of vertices S'. Clearly S' is an independent set of G^{*m}.

Assume now that a larger independent set of G^{*m} exists, which we call T'. Since G^{*m} is formed by taking m copies of G, we can enumerate the copies (in left-to-right order) by $G_1^{*m}, G_2^{*m}, \dots, G_m^{*m}$. For each $T_i, i \in \{1, \dots, m\}$, let T_i be the vertices of G corresponding to the vertices of $T' \cap G_i^{*m}$. Clearly, each T_i is an independent set and the union of all the T_i's forms an m-schedule. But this contradicts the assumption that S is a maximum m-schedule.

The following corollary is then obtained by observing that for any G^{*m}, $m \geq 2$, every job is live at the point between the first and second machines. In the terminology of Sect. 4, the running time of the algorithm on m machines is $\mathcal{O}(mn \cdot 2^{|\mathfrak{R}|})$.

Corollary 6. *JISP on m machines, $m \geq 2$, is fixed-parameter tractable in the number of jobs.*

Bibliography

[BYHN$^+$06] Reuven Bar-Yehuda, Magnús M. Halldórsson, Joseph (Seffi) Naor, Hadas Shachnai, and Irina Shapira. Scheduling split intervals. *SIAM Journal of Computing*, 36:1–15, 2006.

[COR01] Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. In *FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, page 348, Washington, DC, USA, 2001. IEEE Computer Society.

[CS96] Yves Crama and Frits C. R. Spieksma. Scheduling jobs of equal length: complexity, facets and computational results. *Math. Program.*, 72(3):207–227, 1996.

[GW95] András Gyárfás and David B. West. Multitrack interval graphs. *Congressus Numerantium*, 109:109–116, 1995.

[HRST99] Magnús M. Halldórsson, Sridhar Rajagopalan, Hadas Shachnai, and Andrew Tomkins. Scheduling multiple resources. *unpublished manuscript*, 1999.

[Leh74] Philippe G. H. Lehot. An optimal algorithm to detect a line graph and output its root graph. *J. ACM*, 21(4):569–575, 1974.

[NH82] Kazuo Nakajima and S. Louis Hakimi. Complexity results for scheduling tasks with discrete starting times. *Journal of Algorithms*, 3:344–361, 1982.

[WS84] Douglas B. West and David B. Shmoys. Recognizing graphs with fixed interval number is NP-complete. *Discrete Appl. Math.*, 8:295–305, 1984.