On the Equivalence of Disjunctive and Cumulative Models for Identical Parallel Machine Scheduling

Pierre Schaus*

November 4, 2026

Abstract

In Constraint Programming (CP), scheduling n non-preemptive unit-demand tasks on m identical parallel machines can be formulated either via m disjunctive `noOverlap` constraints coupled with machine-assignment variables, or via a single aggregated `cumulative` constraint of capacity m. While replacing individual machine sequences with a cumulative capacity constraint is in general a relaxation for complex resource allocation problems, it is a classical result across graph theory, operations research, and constraint-based scheduling that the two formulations are strictly equivalent on the start-time variables. In this technical note, we formalize this projection equivalence, present two self-contained proofs—one via interval graph perfection and Helly's property, and one via the $O(n \log n)$ greedy Left-Edge sweep-line algorithm—and delineate the boundary conditions under which the equivalence breaks down.

1 Introduction and Model Formulations

Consider a set of n non-preemptive tasks $T = \{1, \dots, n\}$ to be scheduled on m identical parallel machines $M = \{1, \dots, m\}$. Each task $i \in T$ has a fixed processing time $p_i > 0$, a unit resource requirement $h_i = 1$, and a decision variable $s_i \in \mathbb{R}$ representing its start time (possibly subject to release dates, deadlines, or precedence constraints). Once s_i is fixed, task i executes without interruption over the half-open time interval

$$I_i(s_i) = [s_i, s_i + p_i).$$

In Constraint Programming (CP), there are two canonical ways to enforce the resource limitation of the m identical machines.

Definition 1 (Disjunctive Decomposition Model $\mathcal{M}_{\text{noOverlap}}$)**.** In the explicit machine-assignment formulation, each task $i \in T$ is associated with a machine variable $a_i \in M$ (or equivalently m optional interval variables governed by an `alternative` constraint), and each machine $k \in M$ enforces a unary disjunctive `noOverlap` constraint:

$$\mathcal{M}_{\text{noOverlap}} \;=\; \left\{(\mathbf{s}, \mathbf{a}) \in \mathbb{R}^n \times M^n \;\middle|\; \texttt{noOverlap}(\{I_i(s_i) \mid i \in T,\, a_i = k\}) \;\; \forall k \in M\right\}, \tag{1}$$

where $\texttt{noOverlap}(\mathcal{I})$ holds if and only if all intervals in $\mathcal{I}$ are pairwise disjoint: $I_i(s_i) \cap I_j(s_j) = \emptyset$ for all distinct i, j with $a_i = a_j = k$.

Definition 2 (Aggregated Cumulative Model $\mathcal{M}_{\text{cumul}}$)**.** In the aggregated formulation, the assignment variables $\mathbf{a}$ are omitted and the m unary `noOverlap` constraints are replaced by a single `cumulative` constraint [2] of capacity m with unit task heights $h_i = 1$:

$$\mathcal{M}_{\text{cumul}} \;=\; \left\{\mathbf{s} \in \mathbb{R}^n \;\middle|\; \sum_{i \in T : t \in [s_i, s_i + p_i)} 1 \le m \;\; \forall t \in \mathbb{R}\right\}. \tag{2}$$

*This technical note was written with the assistance of Gemini, which also produced the proofs and literature connections. The author's contribution lies in formulating the research question and guiding the model through prompting toward a clear scientific presentation.

Because the m machines in $\mathcal{M}_{\text{noOverlap}}$ are indistinguishable, any valid assignment $\mathbf{a}$ induces up to $m!$ symmetric solutions obtained by permuting machine indices. Model $\mathcal{M}_{\text{cumul}}$ eliminates these symmetries by projecting out $\mathbf{a}$.

2 Equivalence and Machine Reconstruction

Let $\text{proj}_{\mathbf{s}}(\mathcal{M}_{\text{noOverlap}}) = \{\mathbf{s} \in \mathbb{R}^n \mid \exists \mathbf{a} \in M^n \text{ such that } (\mathbf{s}, \mathbf{a}) \in \mathcal{M}_{\text{noOverlap}}\}$ denote the projection of $\mathcal{M}_{\text{noOverlap}}$ onto the start-time variables $\mathbf{s} = (s_1, \ldots, s_n)$.

Theorem 1 (Exactness of the Cumulative Reformulation)**.** *For non-preemptive unit-demand tasks ($h_i = 1$) on m identical parallel machines, the projection of the disjunctive model $\mathcal{M}_{noOverlap}$ onto the start-time variables coincides exactly with the cumulative model $\mathcal{M}_{cumul}$:*

$$\text{proj}_{\mathbf{s}}(\mathcal{M}_{noOverlap}) \;=\; \mathcal{M}_{cumul}. \tag{3}$$

Moreover, given any $\mathbf{s} \in \mathcal{M}_{cumul}$, a conflict-free machine assignment $\mathbf{a} \in M^n$ such that $(\mathbf{s}, \mathbf{a}) \in \mathcal{M}_{noOverlap}$ can be computed in $O(n \log n)$ time without modifying $\mathbf{s}$.

The inclusion $\text{proj}_{\mathbf{s}}(\mathcal{M}_{\text{noOverlap}}) \subseteq \mathcal{M}_{\text{cumul}}$ is immediate: if $(\mathbf{s}, \mathbf{a}) \in \mathcal{M}_{\text{noOverlap}}$, then at any time $t \in \mathbb{R}$ at most one task is active on each machine $k \in M$, so at most m tasks are active in total.

To establish the reverse inclusion $\mathcal{M}_{\text{cumul}} \subseteq \text{proj}_{\mathbf{s}}(\mathcal{M}_{\text{noOverlap}})$ from a graph-theoretic perspective, consider for any fixed $\mathbf{s} \in \mathcal{M}_{\text{cumul}}$ the *interval intersection graph* $G(\mathbf{s}) = (T, E)$ with edge set

$$E \;=\; \{\{i, j\} \subseteq T \mid i \neq j \text{ and } [s_i, s_i + p_i) \cap [s_j, s_j + p_j) \neq \emptyset\}.$$

An assignment $\mathbf{a} \in M^n$ satisfies $(\mathbf{s}, \mathbf{a}) \in \mathcal{M}_{\text{noOverlap}}$ if and only if $a_i \neq a_j$ for all $\{i, j\} \in E$, meaning that $\mathbf{a}$ is a *proper m-coloring* of $G(\mathbf{s})$. Thus, $\mathbf{s} \in \text{proj}_{\mathbf{s}}(\mathcal{M}_{\text{noOverlap}})$ if and only if the chromatic number satisfies $\chi(G(\mathbf{s})) \leq m$.

Lemma 1 (Helly's Property for 1D Intervals)**.** *A finite family of intervals $\{[s_i, s_i + p_i)\}_{i \in C}$ is pairwise intersecting (i.e., $C \subseteq T$ forms a clique in $G(\mathbf{s})$) if and only if all intervals in C share a common point $t^\star \in \mathbb{R}$:*

$$\bigcap_{i \in C} [s_i, s_i + p_i) \neq \emptyset.$$

Proof. Let $t^\star = \max_{i \in C} s_i$ be the latest start time among tasks in C, attained by some task $i^\star \in C$. For every $j \in C$, since $[s_j, s_j + p_j)$ and $[s_{i^\star}, s_{i^\star} + p_{i^\star})$ intersect and $s_j \leq s_{i^\star} = t^\star$, we must have $s_j + p_j > t^\star$. Hence $t^\star \in [s_j, s_j + p_j)$ for all $j \in C$. □

Graph-Theoretic Proof of Theorem 1. By construction, $G(\mathbf{s})$ is an interval graph. By Gallai's theorem [10, 3] (and equivalently Dilworth's theorem on interval orders [5]), every interval graph is chordal and hence *perfect*. In any perfect graph, the chromatic number equals the clique number:

$$\chi(G(\mathbf{s})) \;=\; \omega(G(\mathbf{s})).$$

Let $C \subseteq T$ be a maximum clique of size $\omega(G(\mathbf{s}))$. By Lemma 1, there exists a time point $t^\star \in \mathbb{R}$ such that $t^\star \in [s_i, s_i + p_i)$ for all $i \in C$. Since $\mathbf{s} \in \mathcal{M}_{\text{cumul}}$, evaluating (2) at $t = t^\star$ gives

$$\omega(G(\mathbf{s})) \;=\; |C| \;\leq \sum_{i \in T : t^\star \in [s_i, s_i + p_i)} 1 \;\leq\; m.$$

Consequently, $\chi(G(\mathbf{s})) = \omega(G(\mathbf{s})) \leq m$, so $G(\mathbf{s})$ admits a proper m-coloring $\mathbf{a} \in M^n$. □

From an algorithmic perspective, a conflict-free machine assignment $\mathbf{a} \in M^n$ can be constructed directly via the greedy *Left-Edge* (First-Fit) sweep-line algorithm [11, 9], stated in Algorithm 1.

Algorithm 1: Greedy Left-Edge (First-Fit) Machine Reconstruction

Input: Task intervals $\{[s_i, s_i + p_i)\}_{i \in T}$ satisfying $\mathcal{M}_{\text{cumul}}$ with capacity m.
Output: Conflict-free machine assignments $\mathbf{a} = (a_1, \ldots, a_n) \in \{1, \ldots, m\}^n$.

1: Initialize machine completion times $c_k \leftarrow -\infty$ for all $k \in \{1, \ldots, m\}$.

2: Sort T in non-decreasing order of start times: $s_{\pi(1)} \leq s_{\pi(2)} \leq \cdots \leq s_{\pi(n)}$.

3: **for** $r = 1$ **to** n **do**

4: Let $i = \pi(r)$.

5: Select any idle machine $k^\star \in \{1, \ldots, m\}$ such that $c_{k^\star} \leq s_i$.

6: Assign $a_i \leftarrow k^\star$ and update $c_{k^\star} \leftarrow s_i + p_i$.

7: **return** $\mathbf{a}$.

Constructive Proof of Theorem 1. We prove that Step 5 of Algorithm 1 never fails. Suppose for contradiction that at iteration r, when considering task $i = \pi(r)$ at start time s_i, no machine is idle; that is, $c_k > s_i$ for all $k \in \{1, \ldots, m\}$. Let $j_k \in T$ denote the task most recently assigned to machine k prior to iteration r. Because tasks are processed in non-decreasing order of start times, every such task j_k satisfies

$$s_{j_k} \leq s_i \quad \text{and} \quad s_{j_k} + p_{j_k} = c_k > s_i.$$

Hence, $s_i \in [s_{j_k}, s_{j_k} + p_{j_k})$ for each $k \in \{1, \ldots, m\}$. Since $p_i > 0$, we also have $s_i \in [s_i, s_i + p_i)$. Thus, the m distinct tasks $\{j_1, \ldots, j_m\}$ together with task i are all active at time $t = s_i$, yielding at least $m + 1$ overlapping tasks at $t = s_i$, which contradicts $\mathbf{s} \in \mathcal{M}_{\text{cumul}}$.

Sorting takes $O(n \log n)$ time, and maintaining the idle machines via a priority queue (min-heap keyed by c_k) takes $O(\log m)$ time per task, giving an overall complexity of $O(n \log n)$. □

Example 1. Consider $n = 6$ unit-demand tasks on $m = 3$ identical machines with fixed intervals $T_1 = [0, 3)$, $T_2 = [1, 5)$, $T_3 = [2, 4)$, $T_4 = [3, 7)$, $T_5 = [4, 8)$, and $T_6 = [5, 9)$. Figure 1(a) shows the task intervals and their cumulative usage profile, which peaks at $3 \leq m$ over $[2, 7)$. Running Algorithm 1 assigns $T_1 \to M_1$, $T_2 \to M_2$, $T_3 \to M_3$; at $t = 3$, T_1 finishes so $T_4 \to M_1$; at $t = 4$, T_3 finishes so $T_5 \to M_3$; and at $t = 5$, T_2 finishes so $T_6 \to M_2$, yielding the conflict-free schedule in Figure 1(b). Figure 2 depicts the corresponding interval graph $G(\mathbf{s})$, its maximum clique $\{T_2, T_3, T_4\}$ at $t = 3.5$ ($\omega(G(\mathbf{s})) = 3$), and the resulting 3-coloring.

3 Extensions and Boundary Conditions

It is instructive to examine how far Theorem 1 generalizes and where the equivalence between $\mathcal{M}_{\text{cumul}}$ and $\mathcal{M}_{\text{noOverlap}}$ breaks down. First, suppose task $i \in T$ requires $h_i \in \{1, \ldots, m\}$ identical machines simultaneously throughout $[s_i, s_i + p_i)$, and may be assigned to *any* subset $S_i \subseteq M$ of $|S_i| = h_i$ machines without migration (the parallel-task scheduling model $P \mid size_i \mid \cdot$ [6]).

Proposition 1 (Exactness for Arbitrary Non-Contiguous Machine Subsets)**.** *If each task $i \in T$ requires any subset $S_i \subseteq M$ of $|S_i| = h_i$ machines without migration during $[s_i, s_i + p_i)$, then* $\text{proj}_{\mathbf{s}}(\mathcal{M}_{noOverlap}) = \mathcal{M}_{cumul}$ *still holds.*

Proof. Process tasks in non-decreasing order of start times $s_{\pi(1)} \leq \cdots \leq s_{\pi(n)}$. When task i is considered at time s_i, all currently busy machines are occupied by tasks j that started at $s_j \leq s_i$ and finish at $s_j + p_j > s_i$. By the cumulative constraint $\sum_{j : s_i \in [s_j, s_j + p_j)} h_j \leq m$, the previously scheduled tasks active at s_i occupy at most $m - h_i$ machines. Hence at least h_i machines are idle at time s_i (and remain idle for $[s_i, s_i + p_i)$), so task i can be assigned to any h_i of them. □

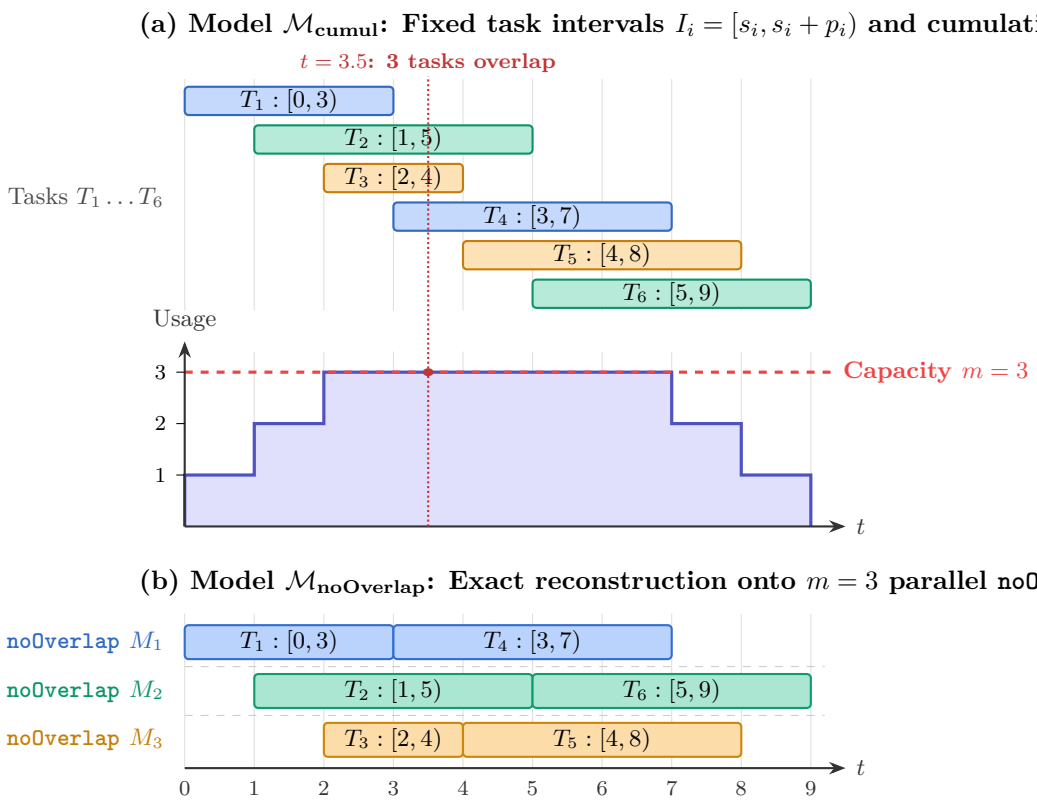


Figure 1: Visual equivalence on Example 1 ($n = 6$, $m = 3$). (a) Any solution to $\mathcal{M}_{\text{cumul}}$ whose cumulative profile stays within capacity $m = 3$ can be partitioned (b) by Algorithm 1 into $m = 3$ conflict-free `noOverlap` schedules without altering any start time s_i.

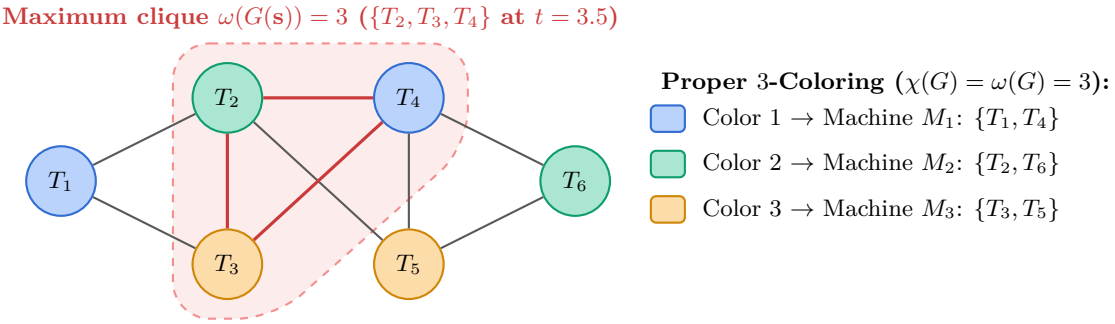


Figure 2: Interval intersection graph $G(\mathbf{s}) = (T, E)$ for Example 1. Because $G(\mathbf{s})$ is an interval (and hence perfect) graph, $\chi(G(\mathbf{s})) = \omega(G(\mathbf{s})) \leq m = 3$.

By contrast, $\mathcal{M}_{\text{cumul}}$ becomes a **strict relaxation** in three practical settings:

1. **Contiguous multi-machine allocation (2D Strip Packing):** If the machines $M = \{1, \dots, m\}$ are linearly ordered and task i requires $h_i > 1$ *adjacent* machines $\{k, k+1, \dots, k+h_i-1\}$ (e.g., contiguous memory allocation, berth allocation, or 2D orthogonal strip packing), Algorithm 1 guarantees h_i idle machines at time s_i, but they may be fragmented (non-adjacent), as shown in Figure 3.

2. **Unsplittable tasks on multi-capacity machines:** If the resource consists of k parallel machines each of capacity $C > 1$ (total capacity $m = kC$) and each task of demand $h_i \in [1, C]$ must execute entirely on a *single* machine, $\mathcal{M}_{\text{cumul}}$ ignores bin-packing fragmentation. For instance, with $k = 2$ machines of capacity $C = 3$ ($m = 6$), three concurrent tasks with

$h_1 = h_2 = h_3 = 2$ satisfy $\sum h_i = 6 \leq m$, yet no two tasks fit on the same machine ($2+2 > 3$).

3. **Machine-dependent attributes and sequence-dependent setups:** If processing times $p_{i,k}$, release dates, eligibility sets $M_i \subsetneq M$ (unrelated or restricted parallel machines), unavailability windows, or sequence-dependent setup times $\tau_{i,j,k}$ depend on the assigned machine k or the immediate predecessor on machine k, the aggregated model $\mathcal{M}_{\text{cumul}}$ loses the individual machine sequences needed to express these constraints.

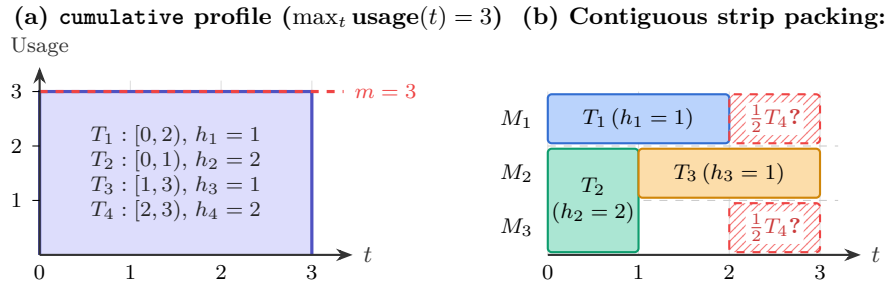


Figure 3: Vertical fragmentation under contiguous multi-machine demands ($h_i > 1$, 2D strip packing). Even when the cumulative usage is at most $m = 3$ everywhere (a), occupying the middle machine M_2 on $[1, 3)$ leaves two non-adjacent machines (M_1 and M_3) idle on $[2, 3)$, preventing a contiguous placement of T_4 ($h_4 = 2$) (b).

4 Historical Notes and Literature Connections

Theorem 1 sits at the intersection of several classical results in discrete mathematics, operations research, and constraint programming:

- **Posets and Interval Graphs (1950–1961):** Viewed as a partial order where $i \prec j \iff s_i + p_i \leq s_j$, the task intervals form an *interval order*. Dilworth's theorem [5] states that the minimum number of chains (disjunctive machines) needed to cover a poset equals the size of its largest antichain (mutually overlapping tasks). Independently, Gallai (1958, reported in Hajnal and Surányi [10]) and Berge [3] established that interval graphs are chordal and perfect ($\chi(G) = \omega(G)$).
- **VLSI Channel Routing and the Left-Edge Algorithm (1971–1979):** Algorithm 1 was introduced by Hashimoto and Stevens [11] as the *Left-Edge Algorithm* for wire-track assignment in printed circuit boards, and later analyzed by Gupta, Lee, and Leung [9], who proved its $O(n \log n)$ optimality for partitioning fixed intervals into the minimum number of conflict-free tracks.
- **Fixed Job Scheduling in Operations Research (1954–2007):** In scheduling theory, assigning fixed-interval jobs to identical parallel machines is known as the *Fixed Job Scheduling (FJS)* feasibility problem, originating in Dantzig and Fulkerson's tanker scheduling problem [4]; see Gertsbakh and Stern [7] and the comprehensive survey by Kolen et al. [12].
- **Constraint-Based Scheduling (1994–2001):** In the CP scheduling literature, following the introduction of the global `cumulative` constraint by Beldiceanu and Contejean [2], the equivalence is treated as folklore rather than stated as a standalone theorem: for instance, Baptiste, Le Pape, and Nuijten [1, Chapters 1, 3, and 5] directly model identical parallel machine problems ($P \mid r_i, \tilde{d}_i \mid \cdot$) as a special case of a single `cumulative` resource of capacity m with unit demands $h_i = 1$, relying implicitly on the interval-graph coloring property [8] to recover individual machine assignments once start times are fixed.

References

[1] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishers / Springer, 2001.

[2] N. Beldiceanu and E. Contejean. Introducing global constraints in CHIP. *Mathematical and Computer Modelling*, 20(12):97–123, 1994.

[3] C. Berge. Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 10:114–115, 1961.

[4] G. B. Dantzig and D. R. Fulkerson. Minimizing the number of tankers to meet a fixed schedule. *Naval Research Logistics Quarterly*, 1(3):217–222, 1954.

[5] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950.

[6] M. Drozdowski. Scheduling multiprocessor tasks—an overview. *European Journal of Operational Research*, 94(2):215–230, 1996.

[7] I. Gertsbakh and H. I. Stern. Minimal resources for fixed and variable job schedules. *Operations Research*, 26(1):68–85, 1978.

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

[9] U. I. Gupta, D. T. Lee, and J. Y.-T. Leung. An optimal solution for the channel-assignment problem. *IEEE Transactions on Computers*, C-28(11):807–810, 1979.

[10] A. Hajnal and J. Surányi. Über die Auflösung von Graphen in vollständige Teilgraphen. *Ann. Univ. Sci. Budapest. Eötvös Sect. Math.*, 1:113–121, 1958.

[11] A. Hashimoto and J. Stevens. Wire routing by optimizing channel assignment within large apertures. In *Proceedings of the 8th Design Automation Workshop (DAC)*, pages 155–169, 1971.

[12] A. W. J. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. C. R. Spieksma. Interval scheduling: A survey. *Naval Research Logistics*, 54(5):530–543, 2007.