

Graph Scheduling with Group Completion Times

Lars Rohwedder 

University of Southern Denmark, Odense, Denmark

Leander Schnaars 

Technical University of Munich, Germany

Abstract

In the Graph Scheduling problem we schedule a given multiset of edges on discrete time steps, such that at each step the set of edges forms a matching. The goal is to minimize the sum of weighted group completion times, where a group is a set of edges and it completes when the last edge has been scheduled. Two popular variants of this problem are Coflow Scheduling and Data Migration. Our main result is extending a recent iterated rounding approach from Coflow Scheduling, roughly corresponding to the bipartite case, to the general Graph Scheduling problem. This yields an essentially tight $(2 + \epsilon)$ -approximation for the asymptotic setting where OPT is assumed to be large. For this we rely on polyhedral techniques from general matching, namely odd-set inequalities, and graph theoretical results on edge colorings in multigraphs. The state-of-the-art approximation algorithm for Data Migration is a $(1 + \phi)$ -approximation that improves when OPT is small. Taking the best of this and our main result, we obtain an improvement of the approximation rate for Data Migration in any regime.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Scheduling algorithms

Keywords and phrases Graph Scheduling, Approximation Algorithms, Iterated Rounding

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.100

Related Version *Full Version:* <https://arxiv.org/abs/2606.31530>

Funding *Lars Rohwedder:* Supported by Dutch Research Council (NWO) project “The Twilight Zone of Efficiency: Optimality of Quasi-Polynomial Time Algorithms” [grant number OCEN.W.21.268].

Leander Schnaars: Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - GRK 2201/2 - Projektnummer 277991500.

1 Introduction

In Graph Scheduling with group completion times, the input consists of a multigraph $G = (V, E)$ and jobs $J_1, \dots, J_B \subseteq E$ with weights $\omega_1, \dots, \omega_B \in \mathbb{R}_{\geq 0}$. In other words, every job consists of some set of edges, where these sets do not necessarily have to be disjoint. In each discrete time step, we are allowed to schedule a matching on the graph, that is to say a set of edges not sharing any vertex. A job is completed if all of its edges have been scheduled. The goal is to minimize the weighted sum of completion times. Group completion times can model many common objectives. If for example there is a single job containing every edge, then the problem is equivalent to makespan minimization, while if every edge belongs to a unique job, this represents min-sum edge coloring. The single job case is equivalent to edge coloring and thus can be solved in polynomial time on bipartite graphs using König’s Theorem [19], while the general case is NP-hard [15]. For min-sum edge coloring, the best known approximation ratio for the general case is a 1.8661-approximation shown by Halldórsson, Kortsatz, and Sviridenko [14]. On bipartite graphs, the best known ratios are $\sqrt{2}$ by Gandhi and Mestre [12] for the unweighted case and 1.796 by Halldórsson, Kortsatz, and Shachnai [13] for the weighted case.



© Lars Rohwedder and Leander Schnaars;

licensed under Creative Commons License CC-BY 4.0

34th Annual European Symposium on Algorithms (ESA 2026).

Editors: Philip Bille, Seth Pettie, and Sabine Storandt; Article No. 100; pp. 100:1–100:22



Leibniz International Proceedings in Informatics

LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Graph Scheduling has been studied mainly in the context of Coflow Scheduling [7, 2, 1, 18, 10, 23] and Data Migration [21, 11]. In Coflow Scheduling, the underlying graph is required to be bipartite and the groups are formed by a disjoint partition of the edge set. This models a data exchange task commonly found in applications such as distributed computing, where between computation stages the different servers have to exchange intermediate results. Most common distributed computing frameworks such as MapReduce, Spark, and Hadoop [8, 34, 30] contain algorithms for such tasks. The problem has thus enjoyed wide attention from both a theoretical as well as an applied side. For several years, the best known guarantees were multiple different 4-approximations [2, 28, 1, 10], while there is a factor $(2 - \epsilon)$ -approximation hardness assuming $P \neq NP$ [24]. Recently, the factor 4 has been improved to 3.415 in [23]. Some authors have studied extensions to the online setting [18] and to a variant with matroids instead of matchings in each time step [16].

In Data Migration the graph is unrestricted, but each job consists of exactly the edges adjacent to some vertex, which implies that every edge belongs to exactly two jobs. As the name implies, this problem models a general data migration setting in a network, where every vertex represents a server and each edge is a data transmission requirement. A server finishes its data transmission task if all adjacent edges have been served. The best known approximation ratio for this setting is $(1 + \phi) \approx 2.618$ [21]. There is an extension of the problem to arbitrary demands on each edge for which a 4.96-approximation is known [11], though this changes the setting in a fundamental way and thus requires completely different approaches.

As remarked by Fukunaga [10], some of the 4-approximation algorithms for Coflow Scheduling [2, 28] also work for general graphs, hence the best so far known approximation ratio for Graph Scheduling is 4. In [23], Rohwedder and Schnaars introduce iterative rounding techniques that achieve an essentially tight $(2 + \epsilon)$ -approximation for Coflow Scheduling in an asymptotic regime where $\text{OPT} \gg \sum_{j \in [B]} \omega_j$. Our main result extends this to general Graph Scheduling:

► **Theorem 1.** *Given a Graph Scheduling instance G with cost vector $\omega \in \mathbb{R}_{\geq 0}^B$ and parameter $\epsilon > 0$, in polynomial time we can find a solution S such that:*

$$\omega(S) \leq (2 + \epsilon) \cdot \text{OPT} + \mathcal{O}\left(\frac{1}{\epsilon}\right) \cdot \sum_{j \in [B]} \omega_j$$

For instances with $\text{OPT} \gg \sum_{j \in [B]} \omega_j$, this gives a $(2 + \epsilon)$ -approximation, which is essentially tight due to the $(2 - \epsilon)$ lower bound. This form of approximation guarantee is not often seen in the literature, but it turns out that many existing algorithms actually have *negative* additive terms in their guarantees when analyzing them more precisely. In [23], an approximation framework is presented which allows obtaining approximation ratios from combinations of multiple such algorithms' allocation guarantees. Their framework can be directly lifted to the more general setting discussed in this work, but we do not employ it here and show the relevant results directly. Two examples in which negative additive terms can be derived are the greedy allocation for Coflow Scheduling in [23] and the local ratio procedure for Data Migration by [21], which we refine in Section 4. These multiplicative and positive and negative additive guarantees can then be combined to obtain improved approximation algorithms only containing a multiplicative term.

In Section 4 we use the refined analysis of local ratio in combination with Theorem 1 to obtain such an improved approximation ratio for Data Migration:

► **Theorem 2.** *There is a 2.617-approximation algorithm for Data Migration with unit sized edges.*

While the main focus is the extension of the techniques from Coflow Scheduling to general graphs and other related settings, from Theorem 1 combined with known results for Graph Scheduling, an improved approximation algorithm for the general case also follows. See Appendix C for full details.

► **Theorem 3.** *There is a polynomial time $\frac{1104}{281} < 3.93$ -approximation algorithm for Graph Scheduling.*

Other variants of Graph Scheduling have been studied, such as when instead of unit size, every edge has some transmission time $p_e \in \mathbb{R}_{\geq 0}$. However, such as in the case of Data Migration, this usually fundamentally changes the nature of the problem and thus also of the required approaches and obtained approximation guarantees. Note that this is distinct from the setting where edges have some multiplicities $p_e \in \mathbb{N}$, for which most results can be transferred at no or only a minor loss in approximation. We also refer to [20] for a broader study of scheduling problems with group completion times, where Graph Scheduling is one of the main applications.

1.1 Techniques from Coflow Scheduling

As this work is a generalization of the algorithms and techniques presented in [23] for Coflow Scheduling, we briefly review the approach and overview the most important technical aspects. The approach has two core steps: In the first step, a deadline is determined for each job. These deadlines are chosen in a way to both fulfill a cost approximation guarantee while also guaranteeing feasibility of an allocation linear program. In the second step, through an iterative rounding procedure an integral point is obtained from the feasible LP. This integral solution does not directly correspond to a feasible schedule, as edges are only assigned to so called blocks and not to individual time slots. To obtain a valid schedule, a time slot assignment is computed using Kőnig's Theorem [19] for bipartite graphs.

The algorithm presented in this work follows the same structure, but due to the general multigraph setting both the LP as well as the allocation algorithm have to be significantly modified. The first step of determining the deadlines works essentially unmodified. The second step however requires larger technical modifications, as we employ results from graph theory for which we have to structurally strengthen the LP. This added structure improves the obtained guarantees, but requires additional care when iteratively rounding.

1.2 Extension to General Graphs

Kőnig's Theorem guarantees the existence and polynomial time computability of a $\Delta(G)$ edge coloring of bipartite graphs of maximum degree $\Delta(G)$, which is an important subroutine of the algorithm. A k edge coloring of a graph directly corresponds to a decomposition into k matchings, so this provides an important tool for creating graph schedules. For general simple graphs, a similar guarantee is given by Vizing's Theorem [33], which gives computability of a $\Delta(G) + 1$ coloring in this case. However, for multigraphs, the best possible guarantees are much worse: By a theorem from Shannon [29], every multigraph can be colored with $\lceil \frac{3}{2}\Delta \rceil$ colors and there are instances for which this bound is tight. So if one were to use the approach from [23] unmodified for general graphs, in the final step in which the schedule is created from the point obtained through iterative rounding, an additional approximation penalty of roughly $\frac{3}{2}$ might be incurred. While the overall approximation ratio might not

worsen by as much due to the algorithm employing a combination of different allocation procedures, it is unlikely that large improvements over the approximation ratio of 4 could be achieved and especially an asymptotic $(2 + \epsilon)$ -approximation would not be possible.

To alleviate this, we strengthen the LP used in [23] using exponentially many odd-set inequalities from Edmonds [9] well known matching polytope. After rounding, the resulting integer solution has stronger properties, with which we can apply recent results from edge coloring theory to construct schedules using much fewer time slots than required by naive application of Shannon's algorithm. However, the added constraints create challenges in the iterative rounding, as iterative rounding is very sensitive towards the number of constraints, so more sophisticated counting arguments are required. The improved LP structure allows us to obtain algorithms which both achieve a strictly better than 4-approximation for general Graph Scheduling as well as a $(2 + \epsilon)$ -approximation for Graph Scheduling in the asymptotic case. The approximation improvements for Data Migration also require the enhanced algorithm, as a direct approach does not yield sufficiently strong guarantees.

2 LP Rounding

In this section we describe a linear program for group completion times on graphs. This LP cannot be directly constructed from the underlying problem instance but rather needs suitable deadlines $C_1, \dots, C_B$ for the jobs $J_1, \dots, J_B$ as a prerequisite. In applications such as seen in Sections 3.1 and 4, these deadlines would for example be obtained through another algorithm as a first step. The deadlines have to be chosen such that the LP is feasible and together they describe certain combinatorial structure which fractional solutions of the underlying problem instance are required to obey. Instead of assigning edges to individual time slots, we group the slots into so called blocks, each of which consists of the time frame between two consecutive deadlines. We show how to iteratively round this LP to obtain an integral point and through this an integral assignment of edges to the blocks. The thus derived point might not be feasible for the original LP, but we establish strong bounds on the maximum constraint violations incurred through this rounding procedure. In a later step of the algorithm, the edge to block assignments are then turned into time slot assignments and thus into valid schedules for the underlying graph scheduling instance.

2.1 Linear Program for General Graphs

Assume that we have B jobs and the corresponding deadlines $C_1, \dots, C_B$ for which the following LP is feasible. In the definition we assume without loss of generality that the jobs are ordered such that $C_1 \leq C_2 \leq \dots \leq C_B$ and we define $C_0 := 0$ for easier notation.

$$\begin{aligned}
\sum_{b \in [B]} x_{e,b} &= 1 & \forall e \in E \\
\sum_{e: v \in e} x_{e,b} &\leq (C_b - C_{b-1}) & \forall v \in V, \forall b \in [B] \\
\sum_{e \in E[U]} x_{e,b} &\leq (C_b - C_{b-1}) \cdot \lfloor \frac{1}{2} |U| \rfloor & \forall U \subseteq V : |U| \text{ odd}, \forall b \in [B] \\
x_{e,b} &= 0 & \forall e \in E, \forall b > b(e) \\
x_{e,b} &\geq 0
\end{aligned}$$

The variables $x_{e,b}$ represent whether edge $e \in E$ is assigned to block $b \in [B]$. Here, E and V are the edge and respectively vertex sets of the underlying graph. For an edge $e \in E$, the function $b(e)$ returns the smallest index $i \in [B]$ of the jobs to which this edge belongs. Before explaining the structure, we slightly modify the LP. For some fixed $\tau \in \mathbb{N}_{\geq 2}$, we round the deadlines $C_1, \dots, C_B$ up to the next multiple of τ . Let the resulting deadlines be $C'_1, \dots, C'_B$ and for all $b \in [B]$ define $k_b \in \mathbb{N}$ such that $C'_b - C'_{b-1} = \tau \cdot k_b$. Using these definitions, we obtain the following LP:

$$\begin{aligned}
 \sum_{b \in [B]} x_{e,b} &= 1 & \forall e \in E \\
 \sum_{e: v \in e} x_{e,b} &\leq k_b \cdot \tau & \forall v \in V, \forall b \in [B] \\
 \sum_{e \in E[U]} x_{e,b} &\leq k_b \cdot \tau \cdot \lfloor \frac{1}{2}|U| \rfloor & \forall U \subseteq V : |U| \text{ odd}, \forall b \in [B] \\
 x_{e,b} &= 0 & \forall e \in E, \forall b > b(e) \\
 x_{e,b} &\geq 0
 \end{aligned} \tag{LP}$$

As explained earlier, we group the previously unit sized time slots into blocks between consecutive deadlines. The first set of constraints ensures that each edge is assigned to some block, while the second set ensures that in each block at most the block's size many edges are adjacent to each vertex. Additionally to these constraints, we also have block versions of the so called odd-set constraints, which ensure that in each block $b \in [B]$ in every odd sized vertex set $|U| \subseteq V$, at most $k_b \cdot \tau \cdot \lfloor \frac{1}{2}|U| \rfloor$ many edges can be included. These types of inequalities are well known and were first introduced by Edmonds [9]. In the case of the standard matching polytope, for non bipartite graphs these constraints are necessary to ensure that all polytope vertices are integral. In our case, vertex solutions in general will not be integral, but the added structure will be exploited in Section 3 to ensure that certain good schedules exist and can be efficiently constructed.

Note that for (LP) without the odd-set constraints there exists an algorithm determining deadlines such that the LP is feasible and such that the weighted sum over these deadlines is 2-approximate with respect to the optimal cost of the underlying Graph Scheduling instance. This algorithm has been used in previous works for Coflow Scheduling and related problems [16, 10]. We show that the algorithm can be extended to our setting with the same guarantees, see Appendix B for details.

2.2 Iterated LP Rounding

In this section we describe how we round a fractional solution to (LP) to obtain an integer point, which can be turned into a valid schedule using a procedure which is explained in Section 3. This integer point might violate some of the original constraints of (LP), but we show that we can control the magnitude of this violation and thus obtain strong delay guarantees for the resulting schedule.

Let $k \in \mathbb{N}$ be a fixed constant. Our rounding procedure has two steps. First we solve (LP) and fix every variable which is integral in the solution. In the second step, we perform two modifications to the constraint sets. We drop every vertex v which now has strictly less than k fractional edges adjacent to it. Dropping a vertex means removing the corresponding vertex degree constraint. For the odd-set constraints, whenever we drop some vertex, we

stop considering its adjacent fractional edges in any odd-set constraint. So it is possible that some edges only contribute to degree constraints of one vertex but are not considered for the odd-set inequalities.

Let $E_b^{(f)}$ be the set of fractional edges remaining in block $b \in [B]$ and let $V_b^{(f)}$ be the set of vertices in that block with at least k fractional edges adjacent. Let $E^{(1)}$ be the set of edges which have been fixed to 1 in a previous step and $E^{(0)}$ respectively the ones fixed to 0. Let $E_b^{(d)} \subseteq E$ be the set of dropped edges in block b , meaning that they were fractional at the time one of their vertices was dropped. Using these definitions, in any step after the first one we have the following LP:

$$\begin{aligned}
 \sum_{b \in [B]} x_{e,b} &= 1 & \forall e \in \bigcup_{b \in [B]} E_b^{(f)} \\
 \sum_{e: v \in e} x_{e,b} &\leq k_b \cdot \tau & b \in [B], \forall v \in V_b^{(f)} \\
 \sum_{e \in E[U] \setminus E_b^{(d)}} x_{e,b} &\leq k_b \cdot \tau \cdot \lfloor \frac{1}{2} |U| \rfloor & \forall U \subseteq V : |U| \text{ odd}, \forall b \in [B] \\
 x_{e,b} &= 0 & \forall e \in E^{(0)} \\
 x_{e,b} &= 1 & \forall e \in E^{(1)} \\
 x_{e,b} &\geq 0
 \end{aligned} \tag{LP R}$$

Note that the odd-set inequalities can still be separated in polynomial time by considering the subgraph of only the edges in $E \setminus E_b^{(d)}$ and using the standard separation procedure [22, 26]. We repeat the fixing variables, dropping constraints, and re-solving of the resulting LP until all variables are integral. Whenever we solve the LP, we assume that we obtain a basic feasible solution, which can be done in polynomial time [31]. Analyzing this procedure requires two major ingredients. We have to bound the maximal constraint violations which the integral point will have with respect to (LP) and we have to show that the procedure is guaranteed to terminate and run in polynomial time.

2.3 Bounding Violations

Assuming that the procedure described in the previous section terminates, we obtain an integral point, which potentially violates some of the constraints of (LP). We give exact upper bounds on this violation:

► **Lemma 4.** *For the integral point obtained through the rounding procedure it holds:*

- a) *The vertex degree constraints in (LP) are exceeded by at most $k - 2$.*
- b) *For each $b \in [B]$ and $U \subseteq V_b$, the corresponding odd-set constraint in (LP) is violated by at most $2(k - 1) \cdot \lfloor \frac{1}{2} |U| \rfloor - 1$.*

Proof. Consider some fixed vertex $v \in V$ and the corresponding vertex constraint $\sum_{e: v \in e} x_e \leq k_b \cdot \tau$. This constraint gets dropped when at most $k - 1$ edges adjacent to v are still fractional. Integral and thus fixed edges never get changed again, so only these $k - 1$ edges could change compared to the solution at the moment the constraint is dropped. If they all get included and thus set to 1, the total sum on the left side would increase by at most $k - 1$. However, as these edges are fractional at the point the constraint gets dropped, the sum over the fixed edges at that point can be at most $k_b \cdot \tau - 1$, hence the maximum violation of vertex

constraints is bounded by $k - 2$. For the odd-set constraints, consider some vertex set $U \subseteq V$ of odd size. The corresponding odd-set constraint never gets dropped from the LP during the procedure, however the considered set of edges will change over time. Every time a vertex v from U is dropped, the fractional edges adjacent to v in $E[U]$ will no longer be considered in this constraint in future iterations. Consider the vertices $v_1, \dots, v_{|U|}$ in U by the order in which they are dropped, with all non-dropped vertices at the end. By definition, there are at most $k - 1$ fractional edges adjacent to a vertex at the time it is dropped. However, when counting the edges we only need to consider the ones which are connected to vertices later in the order, as the ones connected to earlier vertices will have been counted in an earlier step already. Therefore, specifically the very last vertex can never contribute to the edge count in this counting argument. Thus there are at most $(|U| - 1) \cdot (k - 1)$ fractional edges contained in $E_b^{(d)}[U]$. Through the same fractionality argument as above, this bound can further be strengthened by -1 , as the sum over the fixed edges has to be strictly smaller than the upper bound to allow fractional edges to exist. So, assuming that all fractional dropped edges later get included, the constraint bound is exceeded by at most $(|U| - 1) \cdot (k - 1) - 1 = 2(k - 1) \cdot \lfloor \frac{1}{2}|U| \rfloor - 1$. ◀

2.4 Counting Constraints

In order to establish that the procedure terminates, we show that in each step we make progress, meaning that we obtain at least one new integral variable. To prove this, we bound the number of tight linearly independent constraints and show that this is strictly less than the number of variables, which by standard arguments from LP theory suffices to establish the claim. When counting tight constraints in $(\mathbf{LP} \ \mathbf{R})$, we do not consider the constraints of the form $x_{e,b} = 0$, $x_{e,b} = 1$ or $x_{e,b} \geq 0$, as them being tight directly implies existence of an integral variable. Also the fixing constraints and their respective variables are in a one to one correspondence, so these variables do not affect the inequality argument and are thus omitted from the count.

The total number of variables in each intermediate step is $\sum_{b \in [B]} |E_b^{(f)}|$. We independently for each block establish certain bounds on the number of tight linearly independent constraints. Let $b \in [B]$ be thus fixed and as previously consider the sets $V_b^{(f)}$ and $E_b^{(f)}$. For each $v \in V_b^{(f)}$ and for each $e \in \bigcup_{b \in [B]} E_b^{(f)}$ we obtain one constraint.

► **Lemma 5.** *For all $b \in [B]$:* $|V_b^{(f)}| \leq \frac{2}{k} |E_b^{(f)}|$

Proof. By definition, every vertex in $V_b^{(f)}$ has at least k fractional edges adjacent, so for every $b \in [B]$ we have

$$k \cdot |V_b^{(f)}| \leq |\{(v, e) \mid v \in V_b^{(f)}, e \in E_b^{(f)} \cap \delta(v)\}|.$$

Every edge contains exactly two vertices, so we also have the bound

$$|\{(v, e) \mid v \in V_b^{(f)}, e \in E_b^{(f)} \cap \delta(v)\}| \leq 2 \cdot |E_b^{(f)}|.$$

Combining the two and rearranging yields the result. ◀

► **Lemma 6.** *It holds* $|\bigcup_{b \in [B]} E_b^{(f)}| \leq \frac{1}{2} \sum_{b \in [B]} |E_b^{(f)}|$.

Proof. The first set of constraints in **(LP R)** prescribes that $\sum_{b \in [B]} x_{e,b} = 1$, which implies that whenever some edge e has a fractional assignment in some block b , there has to exist at least one other block in which this edge is also fractionally assigned. Hence such an edge only contributes once to the left side of the inequality and contributes at least twice to the right side, yielding the claimed bound. $\blacktriangleleft$

Lemma 5 and Lemma 6 provide bounds for the first two sets of constraints, so it remains to establish a similar bound for the odd-set inequalities. A priori the number of inequalities is exponential, as there is one inequality for each subset $U \subseteq V$ of odd size. However, we claim that the number of such tight constraints which are linearly independent of each other and all other constraints in **(LP R)** is actually linear in $|V_b^{(f)}|$. Whenever we write linearly independent we refer to this notion of independence.

► **Lemma 7.** *In **(LP R)**, the number of tight linearly independent odd-set constraints $|\mathcal{O}_b|$ in block $b \in [B]$ is upper bounded by $|V_b^{(f)}| - 1$.*

Proof. We establish the result in two steps. We first show certain structure which linearly independent constraints must obey and then establish the bound using an inductive argument.

It is well known that the vertex sets belonging to the tight odd-set constraints form a laminar family $\mathcal{L}$, which can be shown using standard uncrossing arguments [27]. We claim that there cannot be an extension of a laminar set using only dropped vertices, meaning that if there is a set $L \in \mathcal{L}$ and set $L_1 \in \mathcal{L}$ which is subset of L such that $L \setminus L_1$ only consists of dropped vertices, then L is linearly dependent of L_1 and possibly some tight edge constraints. As all vertices in $L \setminus L_1$ were dropped at some point, by definition no fractional edge adjacent to them will be considered in the constraint belonging to L . So the only possible difference to the constraint belonging to L_1 is in variables belonging to fixed edges adjacent to vertices in $L \setminus L_1$. However, for each of these by definition there is a equality constraint setting them to 1. Therefore, the constraint of L can be written as a linear combination of these fixed edge constraints and the constraint of L_1 , showing that it is not linearly independent.

We show using induction that the maximum cardinality laminar family representing tight linearly independent odd-sets on $n \geq 2$ non-dropped vertices (and arbitrarily many dropped vertices) has size at most $n - 1$. Any odd-set which induces a constraint that is not simply a linear combination of fixed edge constraints must contain at least 2 non-dropped vertices, as otherwise it cannot contain any fractional edges. For $n = 2$, this leaves a single set with 2 non-dropped vertices and some odd number of dropped vertices as the only option, as any other set would either be disjoint and thus contain only dropped vertices or it would have to be an extension using only dropped vertices, which by the previous argument is not possible.

Now consider some laminar family $\mathcal{L}$ on $n \geq 3$ non-dropped vertices. If $\mathcal{L}$ contains more than one set $L_1, L_2, \dots, L_i$, each not contained in any larger set, then each has to contain at least two non-dropped vertices and thus also strictly fewer than n non-dropped vertices. Thus by the induction hypothesis, if $\ell_1, \dots, \ell_i$ are the respective number of non-dropped vertices in the sets, each can consist of at most $\ell_1 - 1, \dots, \ell_i - 1$ sets. Thus the total number of sets is at most $n - i < n - 1$. If no two or more such sets exist, then either $\mathcal{L}$ contains no set, in which case the claim is trivially true, or it contains exactly one such maximal set L . If L contains exactly one maximal set L_1 , then it has to contain at least one non-dropped vertex not in L_1 to be linearly independent, so applying the induction hypothesis to L_1 directly yields the result. If L contains $i \geq 2$ maximal sets, then again each has to contain non-dropped vertices, so we can apply the induction hypothesis to each and get an upper bound of $n - i$ contained sets in L . Together with L itself this yields the bound $n - 1$.

Note that a single chain of $n - 1$ nested sets, where the innermost set contains 2 non-dropped and an odd number of dropped vertices and every other set extends its predecessor by a single non-dropped and an odd number of dropped vertices has the claimed size, so this bound is tight. $\blacktriangleleft$

2.5 Integral Solution Guarantees

Using the just established results, we derive the following lemma, which gives precise bounds on the achieved constraint violations for the integral point obtained through the iterated rounding procedure.

► **Lemma 8.** *Given integral deadlines $C_1, \dots, C_B$ for which $(\mathbf{LP})$ is feasible, in polynomial time we can find an integral point such that:*

- a) *The vertex degree constraints in $(\mathbf{LP})$ are exceeded by at most 6.*
- b) *For each $b \in [B]$ and $U \subseteq V$, the corresponding odd-set inequality in $(\mathbf{LP})$ is exceeded by at most $14\lfloor \frac{1}{2}|U| \rfloor - 1$.*
- c) *All other constraints in $(\mathbf{LP})$ are fulfilled.*

Proof. We have to show two things, namely that the iterated rounding procedure terminates in polynomial time and that all of the claimed constraint (non-) violations hold. For the runtime, we show that in each iteration the number of linearly independent constraints is strictly lower than the number of variables, which by standard LP theory implies that in a basic solution at least one variable must be integral. Combining the bounds from Lemma 5, Lemma 6, and Lemma 7 we obtain for the constraints **Cons** and variables **Vars** in $(\mathbf{LP} \ \mathbf{R})$:

$$\begin{aligned}
 |\mathbf{Cons}| &= \left| \bigcup_{b \in [B]} E_b^{(f)} \right| + \sum_{b \in [B]} |V_b^{(f)}| + \sum_{b \in [B]} |\mathcal{O}_b| \\
 &\leq \frac{1}{2} \sum_{b \in [B]} |E_b^{(f)}| + \frac{2}{k} \sum_{b \in [B]} |E_b^{(f)}| + \frac{2}{k} \sum_{b \in [B]} |E_b^{(f)}| - B \\
 &= \left(\frac{1}{2} + \frac{4}{k} \right) |\mathbf{Vars}| - B
 \end{aligned}$$

So for $k \geq 8$ we have the desired strict inequality, implying that there is at least one new fixed integral variable in every iteration. From this it directly follows that the procedure runs in polynomial time, as only a polynomial number of iterations of the rounding procedure are needed and as a basic solution to $(\mathbf{LP} \ \mathbf{R})$ can be determined in polynomial time. It is easy to see that c) must hold, as these other constraints are never modified and thus remain valid during each step of the rounding procedure. For the choice $k = 8$, the claims in a) and b) directly follow from the violation bounds shown in Lemma 4. $\blacktriangleleft$

3 Integral Time Slot Assignments

Given an integral assignment of edges to blocks, we want to determine a corresponding assignment of these edges to individual time slots. For bipartite graphs, König's Theorem provides a simple tool to obtain such assignments, but as discussed in Section 1.2, for general multigraphs no similarly strong guarantees hold. However, using further graph properties apart from the maximum degree, useful guarantees can still be established. Further information and historic context for the concepts used and for general edge coloring can be found for example in [25, 5, 32].

Due to space constraints we refer for most of the details to Appendix A in the appendix and only briefly recap the main ideas here. Using the odd-set constraints in **(LP)** and the guarantees from Lemma 8, we show that the so called density of each graph belonging to each block $b \in [B]$ in an integral solution obtained by the rounding procedure is at most $k_b \cdot \tau + 14$. Using recent results from edge-coloring theory for multigraphs [17], this enables us to schedule all edges in each block in at most $k_b \cdot \tau + 14$ time slots. From this result, a procedure with the following guarantees follows, where $\text{GBF}^\tau(C_j)$ refers to the finishing time of job j in the returned schedule.

► **Lemma 9.** *For given deadlines $C_1, \dots, C_B$ for which **LP** is feasible and a parameter $\tau \in \mathbb{N}_{\geq 2}$, there is an algorithm GBF^τ returning a valid schedule such that the following holds for all $j \in [B]$:*

$$\text{GBF}^\tau(C_j) \leq \frac{\tau + 14}{\tau} C_j + \tau + 14$$

3.1 Scheduling Algorithm

The guarantees from Lemma 8 provide bounds on the delay incurred through the iterated rounding procedure, however the lemma implicitly relies on the deadlines used in the definition of **(LP)**. By combining this lemma with an approximation algorithm which generates deadlines, we obtain a more explicit parameterized scheduling algorithm.

► **Theorem 1.** *Given a Graph Scheduling instance G with cost vector $\omega \in \mathbb{R}_{\geq 0}^B$ and parameter $\epsilon > 0$, in polynomial time we can find a solution S such that:*

$$\omega(S) \leq (2 + \epsilon) \cdot \text{OPT} + \mathcal{O}\left(\frac{1}{\epsilon}\right) \cdot \sum_{j \in [B]} \omega_j$$

Proof. Let $G = (V, E)$ be a multigraph representing a Graph Scheduling instance, together with jobs $J_1, \dots, J_B \subseteq E$ and a weight vector $\omega \in \mathbb{R}_{\geq 0}^B$. From Lemma 16 shown in the Appendix in Appendix B, we obtain that there is an algorithm returning deadlines for this instance for which **(LP)** is feasible such that the following upper bound holds with respect to the optimal cost **OPT**:

$$\sum_{j \in [B]} \omega_j C_j \leq 2 \cdot \text{OPT} - \sum_{j \in [B]} \omega_j$$

Combining these deadlines with the procedure from Lemma 9 yields the following guarantees for any $\tau \geq 2$. Here again $\text{GBF}^\tau(C_j)$ is the finishing time of job J_j in the schedule returned by the procedure.

$$\begin{aligned} \sum_{j \in [B]} \omega_j \text{GBF}^\tau(C_j) &\leq \sum_{j \in [B]} \omega_j \left(\frac{\tau + 14}{\tau} C_j + \tau + 14 \right) \\ &= \frac{\tau + 14}{\tau} \sum_{j \in [B]} \omega_j C_j + (\tau + 14) \sum_{j \in [B]} \omega_j \\ &\leq \frac{\tau + 14}{\tau} \left(2 \cdot \text{OPT} - \sum_{j \in [B]} \omega_j \right) + (\tau + 14) \sum_{j \in [B]} \omega_j \\ &= 2 \cdot \frac{\tau + 14}{\tau} \cdot \text{OPT} + \left(\tau + 14 - \frac{\tau + 14}{\tau} \right) \sum_{j \in [B]} \omega_j \end{aligned}$$

As τ can be chosen arbitrarily large, for any $\epsilon > 0$ there is a τ such that we obtain a cost bound of

$$(2 + \epsilon) \cdot \text{OPT} + \mathcal{O}\left(\frac{1}{\epsilon}\right) \cdot \sum_{j \in [B]} \omega_j. \quad \blacktriangleleft$$

Note that we are focusing on the technical ideas in this work and have not fully optimized the various constants appearing throughout. The same ideas as used in [23] should provide small improvements for the guarantees shown in Theorem 1 and further enhancements are likely possible.

4 Data Migration

Data Migration is defined as Graph Scheduling on a general multigraph $G = (V, E)$ with a special job structure. For each vertex $v \in V$ there is a job $J_v := \{e \in \delta_E(v)\}$, so J_v contains all edges adjacent to v . This is equivalent to considering each vertex as a job and defining a vertex to finish whenever all its adjacent edges have been scheduled. As the name suggests, the setting is inspired by a problem in distributed computation and storage, where the vertices for example represent some distributed servers and each edge represents some data migration request. We only consider unit sized edges here as the setting with general demands requires very different techniques.

4.1 $(1 + \phi)$ -Approximation

The so far best known approximation algorithm for Data Migration provides a $(1 + \phi) \approx 2.618$ -approximation [21]. Here $\phi \approx 1.618$ represents the golden ratio. The algorithm is based on a technique called local ratio, where guarantees are first shown on a local scale and then lifted to a general bound. Through more fine-grained analysis of the proofs in [21], we are able to show a slight additive improvement over their guarantees.

► **Lemma 10.** *For any instance of Data Migration with cost vector $\omega \in \mathbb{R}_{\geq 0}^B$, there is an algorithm returning a solution S such that:*

$$\omega(S) \leq (1 + \phi) \cdot \text{OPT} - \frac{1}{2} \sum_{j \in [B]} \omega_j$$

Here OPT refers to the optimal cost of the underlying problem instance. We briefly review the algorithm and involved techniques before establishing the additive improvement. More comprehensive background information and additional examples can be found in works such as [4, 3].

4.2 Local Ratio Concept

Assume that we are given some maximization or minimization problem with a weight vector $\omega \in \mathbb{R}_{\geq 0}^n$. We present the idea for the minimization setting, the maximization case follows analogously. Assume that we have some decomposition of ω into vectors $\omega_1, \dots, \omega_\ell \in \mathbb{R}_{\geq 0}^n$, meaning that $\omega = \sum_{j \in [\ell]} \omega_j$. Let S be some feasible solution to our minimization problem and let OPT be an optimum solution. If for some $\alpha \geq 1$, we can show that for all $j \in [\ell]$, the inequality $\omega_j(S) \leq \alpha \cdot \omega_j(\text{OPT})$ is fulfilled, then it follows that $\omega(S) \leq \alpha \cdot \omega(\text{OPT})$, which shows that S is α -approximate. This basic observation is a direct consequence of the linearity of the objective function. This result is also known as the Local Ratio Theorem. The key in Local Ratio algorithms is in good construction of the weight decomposition. One common approach is an iterative one, where the weight decomposition is constructed step by step, with other algorithmic decisions taking place in-between. It is often much easier to show approximation guarantees for the individual weight functions, as they are based on local structure or by

construction fulfill certain strong properties. For example, in the following section on Data Migration, every individual weight function will be supported on the neighborhood of some vertex, which then enables strong local ratio bounds to be shown.

4.3 Local Ratio in Data Migration

We review the algorithm and most important analysis of the Local Ratio approach employed by Mestre [21] to obtain a $(1 + \phi)$ -approximation for Data Migration. For full details and analysis we refer to their work. The authors present the Algorithm 1 for Data Migration. The idea is to assign every vertex some label and then schedule the edges greedily by their labels. Here, for some vertex $u \in V$, $\text{UN}(u)$ is the set of unlabeled neighbors of the vertex u . The algorithm in this form is actually a class of algorithms, as it depends on a hitherto unspecified choice of weight vector in line 7.

■ **Algorithm 1** The base local ratio algorithm for Data Migration as given in [21]. The weight vector $\hat{\omega}$ in line 7 is calculated through an LP, which we do not include in this pseudocode for easier readability.

```

1 Algorithm ([21]) DMLR( $V, E, \omega$ ):
2 // Labeling Stage
3 foreach  $v \in V$  do
4    $\ell_v \leftarrow \_$ 
5 repeat
6   choose  $u \in V$  maximizing  $\Delta = |\text{UN}(u)|$ 
7   choose  $\hat{\omega}$  with support in  $\text{UN}(u)$ 
8    $\omega \leftarrow \omega - \min \left\{ \frac{\omega(u)}{\hat{\omega}(u)} \mid \hat{\omega}(u) > 0 \right\} \hat{\omega}$ 
9   foreach  $v \in \text{UN}(u) \mid \omega(v) = 0$  do
10     $\ell_v \leftarrow \Delta$ 
11 until every vertex is labeled
12 // Scheduling Stage
13 sort  $(u, v) \in E$  in lexicographic order of  $\langle \min\{\ell_u, \ell_v\}, \max\{\ell_u, \ell_v\} \rangle$ 
14  $S \leftarrow$  empty schedule
15 foreach  $e \in E$  in sorted order do
16   add  $e$  to  $S$  as early as possible
17 return  $S$ 

```

The vector $\hat{\omega}$ in line 7 is chosen as the minimizer of the following LP, where d_i is the degree sequence of the vertices in $\text{UN}(u)$:

$$\begin{aligned}
 \min \quad & \sum_{i \in [\Delta]} (d_i + \Delta - 1) \hat{\omega}_i \\
 \sum_{i \in [\Delta]} (y_i - z_i) \quad & \geq 1 \\
 y_i - z_j \quad & \leq \max\{d_i, j\} \hat{\omega}_i \quad \forall i, j \in [\Delta] \\
 y_i, z_i, \hat{\omega}_i \quad & \geq 0 \quad \forall i \in [\Delta]
 \end{aligned} \tag{LP M}$$

In their analysis, for a given degree sequence d and a weight function ω , they show upper and lower bounds on the weighted sum of completion times and define several constants based on these. The upper bound $\text{UB}(d, \omega)$ is given as follows

$$\text{UB}(d, \omega) = \sum_{i \in [\Delta]} \omega_i (d_i + \Delta - 1),$$

while the lower bound is as follows:

$$\text{LB}(d, \omega) = \min_{\sigma: [\Delta] \rightarrow [\Delta]} \sum_{i \in [\Delta]} \omega_i \max\{\sigma(i), d_i\}$$

Here $\sigma: [\Delta] \rightarrow [\Delta]$ is the set of all permutations on $[\Delta]$. They define $\rho(d) := \inf_{\omega} \frac{\text{UB}(d, \omega)}{\text{LB}(d, \omega)}$ and call this the minimum local ratio of d . The **(LP M)** aims to minimize the local ratio under the assumption $\text{LB}(d, \omega) = 1$, which is valid as the ratio is scaling invariant. The worst case local ratio is defined as $\rho := \sup_d \rho(d)$, with the worst case ratio for degree sequences of length Δ being written as $\rho_{\Delta} := \max_{d: |d|=\Delta} \rho(d)$.

4.4 Bounds on the Local Ratio

In their work, the authors of [21] show several results for the local ratio of Algorithm 1 for Data Migration, which leads to the $(1 + \phi)$ -approximation guarantee of the algorithm. In this section we review their results and subsequently provide more fine-grained guarantees by strengthening and extending their analysis. The following three results can be found in slightly modified form in [21].

► **Lemma 11** ([21]). *For any $\Delta \in \mathbb{N}_{\geq 2}$, the value of ρ_{Δ} is upper bounded by $(1 + \phi) + \frac{2}{\Delta - 1}$.*

► **Lemma 12** ([21]). *For any $\Delta \in \mathbb{N}_+$, there holds $\rho_{\Delta} < \rho_{2\Delta}$.*

Combining these two results leads to the $(1 + \phi)$ -approximation guarantee.

► **Lemma 13** ([21]). *The local ratio ρ is upper bounded by $(1 + \phi)$.*

Keeping in line with the idea behind Local Ratio, we show that this bound can be additively improved on a local scale, which then directly lifts to a general additive improvement.

► **Lemma 14.** *Let ω^{ALG} be the cost function derived in step 7 in Algorithm 1 and let S be any schedule and OPT be an optimal schedule. Then we can bound the local ratio in the following way:*

$$\omega^{\text{ALG}}(S) \leq (1 + \phi) \cdot \omega^{\text{ALG}}(\text{OPT}) - \frac{1}{2} \sum_{j \in [\Delta]} \omega_j^{\text{ALG}}$$

Proof. We start by deriving a sharper version of Lemma 12 using their proof as a base. Let d be a degree sequence of length Δ such that $\rho_{\Delta} = \rho(d)$ and let $k \in \mathbb{N}_{\geq 2}$. We define a new degree sequence d' of length $k\Delta$ which we use to derive a first bound. For this purpose, replace any element in d by k copies of it which are multiplied by k to obtain d' , i.e. the first k elements in d' have value kd_1 , the next k value kd_2 and so on. Solve the **(LP M)** for d' to obtain the values (ω', y', z') . For $i \in [\Delta]$, define $y_i := \sum_{j \in [i \cdot k - (k-1), \dots, i \cdot k]} y'_j$, $z_i := \sum_{j \in [i \cdot k - (k-1), \dots, i \cdot k]} z'_j$, and $\omega_i = k \cdot \sum_{j \in [i \cdot k - (k-1), \dots, i \cdot k]} \omega'_j$. Before we show that (ω, y, z) is feasible for **(LP M)** on d , we derive the following bound:

$$\begin{aligned}
\sum_{i \in [\Delta]} (d_i + \Delta - 1) \omega_i &= \sum_{i \in [\Delta]} (d_i + \Delta - 1) \cdot k \cdot \sum_{j \in [i \cdot k - (k-1), \dots, i \cdot k]} \omega'_j \\
&= \sum_{i \in [k\Delta]} (kd_i + k\Delta - k) \cdot \omega'_i \\
&= \sum_{i \in [\Delta']} (d'_i + \Delta' - k) \cdot \omega'_i \\
&= \sum_{i \in [\Delta']} (d'_i + \Delta' - 1) \cdot \omega'_i - (k-1) \sum_{i \in [\Delta']} \omega'_i \\
&= \sum_{i \in [\Delta']} (d'_i + \Delta' - 1) \cdot \omega'_i - \frac{k-1}{k} \sum_{i \in [\Delta]} \omega_i \\
&\leq \rho_{k\Delta} - \frac{k-1}{k} \sum_{i \in [\Delta]} \omega_i
\end{aligned}$$

From the choice of d and the bound from Lemma 13 this implies

$$\rho_{\Delta} \leq \rho_{k\Delta} - \frac{k-1}{k} \sum_{i \in [\Delta]} \omega_i \leq (1 + \phi) - \frac{k-1}{k} \sum_{i \in [\Delta]} \omega_i.$$

As k can be chosen arbitrarily large, it in fact follows that we can drop the fraction $\frac{k-1}{k}$ to obtain

$$\rho_{\Delta} \leq (1 + \phi) - \sum_{i \in [\Delta]} \omega_i.$$

The ω_i in the previous bound is the weight vector derived from the optimal solution to **(LP M)** for degree sequence d' . For our purposes we require a bound against ω^{LP} , the optimal solution for degree sequence d . We have:

$$\begin{aligned}
1 &\geq \frac{\sum_{i \in [\Delta]} (d_i + \Delta - 1) \omega_i^{\text{LP}}}{\sum_{i \in [\Delta]} (d_i + \Delta - 1) \omega_i} = \frac{\sum_{i \in [\Delta]} (\frac{d_i + \Delta - 1}{2\Delta}) \omega_i^{\text{LP}}}{\sum_{i \in [\Delta]} (\frac{d_i + \Delta - 1}{2\Delta}) \omega_i} \\
&\geq \frac{\frac{1}{2} \cdot \sum_{i \in [\Delta]} \omega_i^{\text{LP}}}{\frac{2\Delta-1}{2\Delta} \cdot \sum_{i \in [\Delta]} \omega_i} = \frac{\Delta}{2\Delta-1} \cdot \frac{\sum_{i \in [\Delta]} \omega_i^{\text{LP}}}{\sum_{i \in [\Delta]} \omega_i}
\end{aligned}$$

In the third step we use the fact that all d_i are in $[\Delta]$, so $\frac{d_i + \Delta - 1}{2\Delta} \in [\frac{1}{2}, \frac{2\Delta-1}{2\Delta}]$. Thus by rearranging we obtain $\sum_{i \in [\Delta]} \omega_i \geq \frac{\Delta}{2\Delta-1} \sum_{i \in [\Delta]} \omega_i^{\text{LP}}$.

We now show that the triple (ω, y, z) is feasible for **(LP M)**. We have

$$\sum_{i \in [k\Delta]} (y'_i - z'_i) \geq 1 \Leftrightarrow \sum_{i \in [\Delta]} (y_i - z_i) \geq 1$$

by simple combination of the terms, so the first set of constraints in **(LP M)** is fulfilled. For any $\ell \in [k]$ and $i, j \in \{\ell \cdot k - (k-1), \dots, \ell \cdot k\}$ we know that

$$y'_i - z'_j \leq \max\{d'_i, j\} \omega'_i = \max\{kd_\ell, j\} \omega'_i$$

holds. By summing these inequalities over all values in the index set, we obtain:

$$\begin{aligned}
y_i - z_j &= \sum_{\kappa \in \{0, \dots, k-1\}} (y'_{i \cdot k - \kappa} - z'_{j \cdot k - \kappa}) \leq \sum_{\kappa \in \{0, \dots, k-1\}} \max\{kd_i, j \cdot k - \kappa\} \omega'_{i \cdot k - \kappa} \\
&\leq \max\{d_i, j\} \omega_i
\end{aligned}$$

So the second set of constraints is also fulfilled, which shows that (ω, y, z) is feasible.

We proceed by bounding the actual cost of any schedule in terms of the algorithm's objective function. Let ω^{LP} be the optimal LP solution calculated in the Algorithm 1 and let ω^{ALG} be the scaled weight function used by the algorithm. Let $\alpha > 0$ be such that $\omega^{\text{LP}} \cdot \alpha = \omega^{\text{ALG}}$. We can derive the following relation for the lower bound $\text{LB}(d, \omega^{\text{LP}})$:

$$\begin{aligned} \text{LB}(d, \omega^{\text{LP}}) &= \max_{\sigma: [\Delta] \rightarrow [\Delta]} \sum_{i \in [\Delta]} \omega_i^{\text{LP}} \max\{d_i, \sigma(i)\} \\ &= \max_{\sigma: [\Delta] \rightarrow [\Delta]} \sum_{i \in [\Delta]} \frac{1}{\alpha} \cdot \omega_i^{\text{ALG}} \max\{d_i, \sigma(i)\} = \frac{1}{\alpha} \text{LB}(d, \omega^{\text{ALG}}) \end{aligned}$$

Using the previous two inequalities, the cost $\omega(S)$ can be bounded in the following way:

$$\begin{aligned} \omega^{\text{ALG}}(S) &\leq \text{UB}(d, \omega^{\text{ALG}}) = \frac{\text{UB}(d, \omega^{\text{ALG}})}{\text{LB}(d, \omega^{\text{ALG}})} \cdot \text{LB}(d, \omega^{\text{ALG}}) = \rho(d) \cdot \text{LB}(d, \omega^{\text{ALG}}) \\ &\leq \left(\rho_\infty - \frac{\Delta}{2\Delta-1} \sum \omega_i^{\text{LP}} \right) \cdot \text{LB}(d, \omega^{\text{ALG}}) \\ &\leq (1 + \phi) \text{LB}(d, \omega^{\text{ALG}}) - \left(\frac{\Delta}{2\Delta-1} \sum \frac{1}{\alpha} \omega_i^{\text{ALG}} \right) \cdot \alpha \cdot \text{LB}(d, \omega^{\text{LP}}) \\ &\stackrel{\text{LB}(d, \omega^{\text{LP}}) \geq 1}{\leq} (1 + \phi) \text{LB}(d, \omega^{\text{ALG}}) - \frac{\Delta}{2\Delta-1} \sum \omega_i^{\text{ALG}} \end{aligned}$$

In the third step we use that ω^{ALG} is a scalar multiple of ω^{LP} and thus $\frac{\text{UB}(d, \omega^{\text{ALG}})}{\text{LB}(d, \omega^{\text{ALG}})} = \frac{\text{UB}(d, \omega^{\text{LP}})}{\text{LB}(d, \omega^{\text{LP}})}$ and as the algorithm chooses the ω^{LP} which minimizes this ratio, we have $\frac{\text{UB}(d, \omega^{\text{LP}})}{\text{LB}(d, \omega^{\text{LP}})} = \inf_{\omega} \frac{\text{UB}(d, \omega)}{\text{LB}(d, \omega)} = \rho(d)$. The claimed bound follows from the observations $\text{LB}(d, \omega^{\text{ALG}}) \leq \omega^{\text{ALG}}(\text{OPT})$ and $\frac{\Delta}{2\Delta-1} > \frac{1}{2}$. $\blacktriangleleft$

From the Local Ratio Theorem and Lemma 14 it follows directly that for any weight vector ω , the Algorithm 1 returns a solution which also fulfills a bound of this general form:

► **Lemma 10.** *For any instance of Data Migration with cost vector $\omega \in \mathbb{R}_{\geq 0}^B$, there is an algorithm returning a solution S such that:*

$$\omega(S) \leq (1 + \phi) \cdot \text{OPT} - \frac{1}{2} \sum_{j \in [B]} \omega_j$$

4.5 Combined Approximation

Using our results for general Graph Scheduling combined with the improved analysis of the local ratio procedure for Data Migration, we show a slightly improved approximation guarantee. As a first step, using the results from Section 3.1, we obtain the following bound.

► **Lemma 15.** *For any instance of Data Migration with cost vector $\omega \in \mathbb{R}_{\geq 0}^B$, there is an algorithm returning a solution S such that:*

$$\omega(S) \leq \left(2 + \frac{28}{97}\right) \cdot \text{OPT} + \frac{10656}{97} \sum_{j \in [B]} \omega_j$$

Proof. Follows directly from the bound in the proof of Theorem 1 with $\tau = 97$. $\blacktriangleleft$

Combining the guarantees from Lemma 10 and Lemma 15 leads to the following purely multiplicative bound, which is an improvement over the best known $(1 + \phi)$ -approximation.

► **Theorem 2.** *There is a 2.617-approximation algorithm for Data Migration with unit sized edges.*

Proof. The procedure is straightforward: Run both the algorithms from Lemma 10 and Lemma 15 independently and return the schedule with lower cost. Let S_1 and S_2 be the respective schedules returned by the algorithm. The overall cost is then given by $\min\{\omega(S_1), \omega(S_2)\}$.

If instead of returning the schedule with lower cost, we return the first schedule with some probability $\lambda \in [0, 1]$ and the second with probability $1 - \lambda$, we obtain a random schedule S_R with cost at least as high. Bounding the costs, we obtain:

$$\begin{aligned} \min\{\omega(S_1), \omega(S_2)\} &\leq \mathbb{E}[\omega(S_R)] \\ &\leq \lambda\omega(S_1) + (1 - \lambda)\omega(S_2) \\ &= \left(\lambda(1 + \phi) + (1 - \lambda)\frac{222}{97}\right) \text{OPT} + \left(-\frac{\lambda}{2} + (1 - \lambda)\frac{10656}{97}\right) \sum_{j \in [B]} \omega_j \end{aligned}$$

For the choice $\lambda = \frac{21312}{21409}$, the additive term vanishes and the first term gives the desired multiplicative factor $2.6165 \dots < 2.617$ ◀

References

- 1 Saksham Agarwal, Shijin Rajakrishnan, Akshay Narayan, Rachit Agarwal, David Shmoys, and Amin Vahdat. Sincronia: near-optimal network design for coflows. In *Proceedings of SIGCOMM*, pages 16–29. ACM, 2018. doi:10.1145/3230543.3230569.
- 2 Saba Ahmadi, Samir Khuller, Manish Purohit, and Sheng Yang. On Scheduling Coflows. In *Proceedings of IPCO*, volume 10328, pages 13–24. Springer International Publishing, 2017. Series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-319-59250-3_2.
- 3 Reuven Bar-Yehuda, Keren Bendel, Ari Freund, and Dror Rawitz. Local ratio. *ACM Comput. Surv.*, 36(4):422–463, 2004. doi:10.1145/1041680.1041683.
- 4 Reuven Bar-Yehuda and Shimon Even. A local-ratio theorem for approximating the weighted vertex cover problem. In *Analysis and Design of Algorithms for Combinatorial Problems*, North-Holland mathematics studies, pages 27–45. Elsevier, 1985. doi:10.1016/S0304-0208(08)73101-3.
- 5 Yan Cao, Guantao Chen, Guangming Jing, Michael Stiebitz, and Bjarne Toft. Graph edge coloring: A survey. *Graphs Comb.*, 35(1):33–66, 2019. doi:10.1007/s00373-018-1986-5.
- 6 Guantao Chen, Guangming Jing, and Wenan Zang. Proof of the Goldberg–Seymour conjecture on edge-colorings of multigraphs. *J. Comb. Optim.*, 50(3), 2025. doi:10.1007/s10878-025-01348-6.
- 7 Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. Efficient coflow scheduling with Varys. In *Proceedings of SIGCOMM*, pages 443–454. ACM, 2014. doi:10.1145/2619239.2626315.
- 8 Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. doi:10.1145/1327452.1327492.
- 9 Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B*, 69:125–130, 1965.
- 10 Takuro Fukunaga. Integrality Gap of Time-Indexed Linear Programming Relaxation for Coflow Scheduling. In *Proceedings of APPROX/RANDOM*, volume 245 of *LIPIcs*, pages 36:1–36:13, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.36.

- 11 Rajiv Gandhi, Magnús M Halldórsson, Guy Kortsarz, and Hadas Shachnai. Corrigendum: Improved results for data migration and open shop scheduling. *ACM Trans. Algorithms*, 9(4):1–7, 2013. doi:10.1145/2500123.
- 12 Rajiv Gandhi and Julián Mestre. Combinatorial algorithms for data migration to minimize average completion time. *Algorithmica*, 54(1):54–71, 2009. doi:10.1007/s00453-007-9118-2.
- 13 Magnús M Halldórsson, Guy Kortsarz, and Hadas Shachnai. Sum coloring interval and k-claw free graphs with application to scheduling dependent jobs. *Algorithmica*, 37(3):187–209, 2003. doi:10.1007/s00453-003-1031-8.
- 14 Magnús M Halldórsson, Guy Kortsarz, and Maxim Sviridenko. Min sum edge coloring in multigraphs via configuration LP. In *Integer Programming and Combinatorial Optimization*, pages 359–373. Springer Berlin Heidelberg, 2008. doi:10.1007/978-3-540-68891-4_25.
- 15 Ian Holyer. The np-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981. doi:10.1137/0210055.
- 16 Sungjin Im, Benjamin Moseley, Kirk Pruhs, and Manish Purohit. Matroid Coflow Scheduling. In *Proceedings of ICALP*, volume 132 of *LIPIcs*, pages 145:1–145:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. ISSN: 1868-8969. doi:10.4230/LIPIcs.ICALP.2019.145.
- 17 Guangming Jing. On edge coloring of multigraphs, 2024. arXiv:2308.15588.
- 18 Samir Khuller, Jingling Li, Pascal Sturfels, Kevin Sun, and Prayaag Venkat. Select and permute: An improved online framework for scheduling to minimize weighted completion time. *Theoretical Computer Science*, 795:420–431, 2019. Publisher: Elsevier BV. doi:10.1016/j.tcs.2019.07.026.
- 19 Dénes König. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen*, 77:453–465, 1916. doi:10.1007/BF01456961.
- 20 Alexander Lindermayr, Zhenwei Liu, and Nicole Megow. Unifying scheduling algorithms for group completion time, 2025. doi:10.48550/arXiv.2501.17682.
- 21 Julián Mestre. Adaptive local ratio. *SIAM J. Comput.*, 39(7):3038–3057, 2010. doi:10.1137/080731712.
- 22 Manfred W Padberg and Mendu R Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7(1):67–80, 1982. doi:10.1287/MOOR.7.1.67.
- 23 Lars Rohwedder and Leander Schnaars. 3.415-Approximation for Coflow Scheduling via Iterated Rounding. In *Proceedings of ICALP*, volume 334 of *LIPIcs*, pages 128:1–128:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.128.
- 24 Sushant Sachdeva and Rishi Saket. Optimal inapproximability for scheduling problems via structural hardness for hypergraph vertex cover. In *2013 IEEE Conference on Computational Complexity*, pages 219–229. IEEE, 2013. doi:10.1109/cc.2013.30.
- 25 Diego Scheide. *Edge Colourings of Multigraphs*. PhD thesis, Technische Universität Ilmenau, 2009.
- 26 Alexander Schrijver. *Combinatorial Optimization. Algorithms and Combinatorics*. Springer, Berlin, Germany, 2003 edition, 2002.
- 27 Alexander Schrijver and Paul D Seymour. *A proof of total dual integrality of matching polyhedra*. Stichting Mathematisch Centrum. Zuivere Wiskunde, 1977.
- 28 Mehrnoosh Shafiee and Javad Ghaderi. An improved bound for minimizing the total weighted completion time of coflows in datacenters. *IEEE ACM Trans. Netw.*, 26(4):1674–1687, 2018. doi:10.1109/TNET.2018.2845852.
- 29 Claude Shannon. A theorem on coloring the lines of a network. *J. Math. Phys.*, 28(1-4):148–152, 1949.
- 30 Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *Proceedings of IEEE MSST*, pages 1–10, 2010. doi:10.1109/MSST.2010.5496972.

- 31 Éva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986. doi:10.1287/opre.34.2.250.
- 32 Bjarne Toft and Robin Wilson. A brief history of edge-colorings - with personal reminiscences. *Discrete Mathematics Letters*, 6:38–46, 2021. doi:10.47443/dml.2021.s105.
- 33 Vadim Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz*, 3:25–30, 1964.
- 34 Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, page 10, USA, 2010. USENIX Association. URL: <https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets>.

A Graph Theory and Edge Colorings

In this section we briefly review relevant graph edge coloring theory, which is used to turn edge to block assignments into valid schedules. We use $\chi'(G)$ to refer to the chromatic index of a graph G , so the minimum number of colors required to properly edge color G .

A.1 Density of a Graph

In our case density refers to edge colorings of the graph and should not be confused with the more common notion of density which relates the number of vertices and number of edges in a graph.

Consider a graph $G = (V, E)$ and a valid edge coloring $c : E \rightarrow [k]$. It is clear by definition that every one of the k color classes $E_1, \dots, E_k$ has to form a matching in G , so it immediately follows that $|E_i| \leq \lfloor \frac{1}{2}|V| \rfloor$. The same relation has to hold for every subgraph $H \subseteq G$ with at least 2 vertices, meaning that $|E_i[H]| \leq \lfloor \frac{1}{2}|V[H]| \rfloor$. Because the color classes in a valid coloring form a partition of the edge set, it follows that $|E[H]| \leq k \cdot \lfloor \frac{1}{2}|V[H]| \rfloor$. As this has to hold for any subgraph with at least 2 vertices, by rearranging we obtain a lower bound on k , or more specifically a lower bound on the chromatic index of G . This is called the density $\mathcal{W}(G)$:

$$\mathcal{W}(G) = \max_{H \subseteq G: |V[H]| \geq 2} \left\lceil \frac{|E[H]|}{\lfloor \frac{1}{2}|V[H]| \rfloor} \right\rceil$$

As just observed, for any graph G there holds $\mathcal{W}(G) \leq \chi'(G)$. Note that sometimes Γ is used in place of $\mathcal{W}$ to denote the density. For $\mathcal{W}(G) \geq \Delta(G)$, the density of a general multigraph can be determined in polynomial time [26]. For $\mathcal{W}(G) < \Delta(G)$ this is open, but this case is not relevant to our analysis.

A.2 LP Bounds and Density

As shown in Section 2.2, we can find an integral point which is feasible for (LP) with some slightly weakened constraints. More specifically, for some appropriate choice of $k, \tau \in \mathbb{N}$, we can find a solution x such that for every $b \in [B]$ and $U \subseteq V$, the following constraint is valid:

$$\sum_{e \in E[U]} x_e \leq (k_b \cdot \tau + 2(k - 1)) \cdot \lfloor \frac{1}{2}|U| \rfloor - 1$$

Let $G_{b,x}$ be the graph obtained by only including the edges of block b which are integral in the LP solution. By rearranging above equation, we obtain that $\mathcal{W}(G_{b,x}) \leq k_b \cdot \tau + 2(k - 1)$. From the bound on the vertex constraint violation, we additionally know that $\Delta(G_{b,x}) \leq k_b \cdot \tau + k - 2$.

So this provides upper bounds for both the maximum degree and the density of the induced graph in each block. For the choice $k = 8$ from Lemma 8, this yields $\Delta(G_{b,x}) \leq k_b \cdot \tau + 6$ and $\mathcal{W}(G_{b,x}) \leq k_b \cdot \tau + 14$.

A.3 Efficiently Realizable Upper Bounds on the Chromatic Index

Determining the exact chromatic index $\chi'(G)$ is an NP-hard problem and in fact it is even hard to approximate strictly better than a factor of $\frac{4}{3}$, assuming $P \neq NP$ [15]. While for simple graphs there exists an algorithm which always returns a coloring using at most $\Delta + 1$ colors, for multigraphs the situation is more complicated, as discussed in Section 1.2.

We call a function f which maps graphs to integers an efficiently realizable upper bound if for any graph G we can find an integral edge coloring using at most $f(G)$ colors in time polynomial in $|V|$ and $|E|$. So for simple graphs, $f(G) = \Delta(G) + 1$ is an efficiently realizable upper bound and for general graphs, $f(G) = \frac{3}{2}\Delta(G)$ is another example. This concept is closely related to the well known Goldberg-Seymour conjecture, which states that for a general multigraph G it holds that $\chi'(G) \leq \max\{\Delta(G) + 1, \mathcal{W}(G)\}$. In recent work, the authors of [6] gave a non-constructive proof of the Goldberg-Seymour conjecture, with one of the authors later also providing an algorithm realizing this bound [17]. Thus, given a general multigraph G , $f(G) := \max\{\Delta(G) + 1, \mathcal{W}(G)\}$ is an efficiently realizable upper bound. As $\Delta(G)$ and $\mathcal{W}(G)$ are both lower bounds on the chromatic index, this result implies that one can in polynomial time find a coloring using at most one additional color compared to the optimum. From the NP-hardness of determining the chromatic index it follows that this result is best possible. Note that the algorithm in [17] is currently not peer-reviewed yet. However, there are long established slightly weaker constructive guarantees, such as contained in [25], which could be applied instead and only worsen the resulting guarantees by a bit. They show that $f(G) = \max\{\lfloor \frac{15}{14}\Delta(G) + \frac{12}{14} \rfloor, \mathcal{W}(G)\}$ and $f(G) = \max\{\Delta(G), \mathcal{W}(G)\} + \sqrt{\frac{1}{2} \max\{\Delta(G), \mathcal{W}(G)\}}$ are efficiently realizable upper bounds. For all relevant values of $\Delta(G)$ and $\mathcal{W}(G)$, the minimum of the two terms is only worse than the Goldberg-Seymour bound by at most a small additive constant. In Lemma 9, the increase of each block's size through the procedure is upper bounded by 14, so a small additional additive loss through a potential worse edge coloring only slightly increases this and thus does not affect approximation guarantees much. In fact, for the vertex degree bounds and density bounds derived in Lemma 8, for the choices of τ in Theorem 2 and Theorem 3 the weaker guarantees yield the same results, as the density term still dominates. The asymptotic guarantee of $(2 + \epsilon)$ is specifically also not affected at all and only requires a bit larger choice of τ in the corresponding application of GBF^τ to recover the result.

A.4 Delay Bound

Using the results of Appendix A.2 and Appendix A.3, we describe how to find an edge to time slot assignment given the block assignment and show upper bounds on the delay. We first consider each block separately, before lifting these partial results to overall bounds.

Block Size

For some fixed $b \in [B]$ and an integral solution x as provided by the rounding procedure from Section 2.2, consider the subgraph $G_{b,x}$ consisting of the edges assigned to block b . From Lemma 8, we know that $\Delta(G_{b,x}) \leq k_b \cdot \tau + 6$ and $\mathcal{W}(G_{b,x}) \leq k_b \cdot \tau + 14$. From Appendix A.3 it follows that in polynomial time we can find a coloring of $G_{b,x}$ using at most

$$\max\{\Delta(G_{b,x}) + 1, \mathcal{W}(G_{b,x})\}$$

colors. To obtain worst case guarantees it suffices to focus on the second term, as it will strictly dominate the first term under worst case assumptions.

Combined Guarantees

Combining all the previous results leads to the following delay guarantee, where $\text{GBF}(C_j)$ is the finishing time of the j -th job in the schedule returned by the algorithm. The analysis is similar to the one seen in [23].

► **Lemma 9.** *For given deadlines $C_1, \dots, C_B$ for which **LP** is feasible and a parameter $\tau \in \mathbb{N}_{\geq 2}$, there is an algorithm GBF^τ returning a valid schedule such that the following holds for all $j \in [B]$:*

$$\text{GBF}^\tau(C_j) \leq \frac{\tau + 14}{\tau} C_j + \tau + 14$$

Proof. The algorithm works by first rounding up all deadlines $C_1, \dots, C_B$ to the next multiple of τ to obtain the deadlines $\bar{C}_1, \dots, \bar{C}_B$. As before, for $b \in [B]$, let the block sizes be given by $k_b \cdot \tau$, for $k_b \in \mathbb{N}_{\geq 1}$ and we assume without loss of generality that no two rounded deadlines coincide. Using this construction, from Lemma 8 we obtain that we can find an integral edge to block assignment where each block $G_{b,x}$ has degree at most $k_b \cdot \tau + 6$ and density at most $k_b \cdot \tau + 14$. Hence, using the results from Appendix A.3, we can also find a coloring, so an edge to time slot assignment, using at most $k_b \cdot \tau + 14$ time slots. For some fixed $j \in [B]$, define $a \in [0, \tau)$ and $k \in \mathbb{N}$ such that $C_j = k \cdot \tau + a$. Assume $a > 0$ for now, then $\bar{C}_j = (k + 1) \cdot \tau$. The size of each previous block and the block belonging to C_j increases to at most $\tau + 14$, so the final time slot of the block after the rounding and the slot assignment is latest at time point $j \cdot (\tau + 14)$. As each block has size at least τ , we have $j \leq k + 1$ and thus $j \cdot (\tau + 14) \leq (k + 1) \cdot (\tau + 14)$. So we obtain:

$$\text{GBF}^\tau(C_j) \leq (k + 1) \cdot (\tau + 14) = \frac{(\tau + 14)}{\tau} \cdot \tau \cdot k + \tau + 14 \leq \frac{\tau + 14}{\tau} C_j + \tau + 14$$

For $a = 0$ the derivation is analogously, just with a stronger starting upper bound on the finishing time. ◀

B 2-Approximate Deadlines

In order to employ (**LP**) and the related rounding results algorithmically, deadlines with good cost guarantees are required. By a generalization from the so called Concurrent Open Shop Scheduling, it is known that the deadlines cannot be approximated strictly better than a factor of 2, assuming $P \neq NP$ [24]. Algorithms to determine such deadlines have been studied in various contexts and by combining two partial results, we obtain the following bound which shows that tight approximations are possible and that we can even gain a slight additive improvement over the guarantee.

► **Lemma 16.** *For a given instance of Graph Scheduling, there is a polynomial time algorithm determining deadlines $C_1, \dots, C_B$ for which (LP) is feasible and for which the following cost bound holds with respect to the optimal cost OPT of the underlying instance:*

$$\sum_{j \in [B]} \omega_j C_j \leq 2 \cdot \text{OPT} - \sum_{j \in [B]} \omega_j$$

Proof. The proof follows by combining two known results from the literature. In [20] they show that for any downward closed polytope, using a randomized rounding scheme, one can find $(2 + \epsilon)$ -approximate deadlines such that the corresponding LP is feasible. The additive ϵ can be omitted in case all entries are polynomially bounded. The (LP) fits into their setting, as the degree and odd-set constraints are downward closed, hence we can obtain such 2-approximate deadlines. The polynomial time solvability of the involved LP follows from the well known separation procedure for odd-set inequalities. The same rounding idea has also been used in previous algorithms for Coflow Scheduling [16, 10, 23]. In [23] the analysis of the rounding procedure in the context of Coflow Scheduling has been improved, yielding the additive $-\sum_{j \in [B]} \omega_j$ improvement. This proof directly transfers to our general version, as no properties exclusive to the more restricted case are used. ◀

C Improved Graph Scheduling Approximation

Using the allocation guarantee from Lemma 9 together with guarantees from the literature, Theorem 3 can be shown, which gives a slight improvement on the previously best known approximation ratio of 4 for Graph Scheduling. Note that as mentioned before, this ratio is not optimized and should be straightforward to improve using similar ideas as contained in [23].

► **Theorem 3.** *There is a polynomial time $\frac{1104}{281} < 3.93$ -approximation algorithm for Graph Scheduling.*

Proof. We pick two algorithms which we run independently and return the schedule with lower cost. From the proof of Theorem 1, we know that we can find a solution S_1 with cost:

$$\omega(S_1) \leq 2 \cdot \frac{\tau + 14}{\tau} \cdot \text{OPT} + \left(\tau + 14 - \frac{\tau + 14}{\tau}\right) \sum_{j \in [B]} \omega_j$$

From the algorithms for Coflow Scheduling contained in [2, 28], combined with refined analysis from [23], it follows that there is an algorithm returning a solution S_2 with cost:

$$\omega(S_2) \leq 4 \cdot \text{OPT} - 3 \sum_{j \in [B]} \omega_j$$

Similar to the proof of Theorem 2, we compare the minimum cost of the two schedules to the cost of a randomized algorithm RALG which runs the first algorithm with probability λ_1 and the second with probability λ_2 . For $\tau := 34$ and $\lambda_1 := \frac{17}{281}$ and $\lambda_2 := \frac{264}{281}$, this gives the desired bound:

100:22 Graph Scheduling with Group Completion Times

$$\begin{aligned}\min\{\omega(S_1) , \omega(S_2)\} &\leq \mathbb{E}[\omega(S_{\text{RALG}})] \\ &= \lambda_1 \omega(S_1) + \lambda_2 \omega(S_2) \\ &= \frac{17}{281} \left(\frac{48}{17} \cdot \text{OPT} + \left(48 - \frac{24}{17} \right) \sum_{j \in [B]} \omega_j \right) \\ &\quad + \frac{264}{281} \left(4 \cdot \text{OPT} - 3 \sum_{j \in [B]} \omega_j \right) \\ &= \frac{1104}{281} \cdot \text{OPT}\end{aligned}$$

