Improved Results for Data Migration and Open Shop Scheduling*

Rajiv Gandhi† Magnús M. Halldórsson‡ Guy Kortsarz† Hadas Shachnai§

Abstract

The *data migration* problem is to compute an efficient plan for moving data stored on devices in a network from one configuration to another. We consider this problem with the objective of minimizing the sum of completion times of all storage devices. It is modeled by a transfer graph, where vertices represent the storage devices, and the edges indicate the data transfers required between pairs of devices. Each vertex has a non-negative weight, and each edge has a release time and a processing time. A vertex completes when all the edges incident on it complete, under the constraint that two edges incident on the same vertex cannot be processed simultaneously. The objective is to minimize the sum of weighted completion times of all vertices. Kim (*Proc. ACM-SIAM Symposium on Discrete Algorithms, 97-98, 2003*) gave a 9-approximation algorithm for the problem when edges have arbitrary processing times and are released at time 0. We improve Kim's result by giving a 5.06-approximation algorithm. We also address the *open shop scheduling* problem, $O|r_j|\sum w_jC_j$, and show that it is a special case of the data migration problem. Queyranne and Sviridenko (*Journal of Scheduling, 5:287-305, 2002*) gave a 5.83-approximation algorithm for the non-preemptive version of the open shop problem. They state as an obvious open question whether there exists an algorithm for open shop scheduling that gives a performance guarantee better than 5.83. Our 5.06 algorithm for data migration proves the existence of such an algorithm. Crucial to our improved result is a property of the linear programming relaxation for the problem. Similar linear programs have been used for various other scheduling problems. Our technique may be useful in obtaining improved results for these problems as well.

Key Words and Phrases: Data migration, open shop, scheduling, linear programming, LP rounding, approximation algorithms.

*A preliminary version of this paper appears in the proceedings of the 31^{st} *International Colloquium on Automata, Languages and Programming (ICALP'04).*

†Department of Computer Science, Rutgers University, Camden, NJ 08102. E-mail: {rajivg,guyk}@camden.rutgers.edu.

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

§Department of Computer Science, The Technion, Haifa 32000, Israel. E-mail: hadas@cs.technion.ac.il. Part of this work was done while the author was on leave at Bell Laboratories, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974.

1 Introduction

The *data migration* problem arises in large storage systems, such as *Storage Area Networks* [14], where a dedicated network of disks is used to store multimedia data. As the data access pattern changes over time, the load across the disks needs to be rebalanced so as to continue providing efficient service. This is done by computing a new data layout and then "migrating" data to convert the initial data layout to the target data layout. While migration is being performed, the storage system is running suboptimally, therefore it is important to compute a data migration schedule that converts the initial layout to the target layout quickly.

This problem can be modeled as a *transfer graph* [15], in which the vertices represent the storage disks and an edge between two vertices u and v corresponds to a data object that must be transferred from u to v, or vice-versa. Each edge has a processing time (or length) that represents the transfer time of a data object between the disks corresponding to the end points of the edge. An important constraint is that any disk can be involved in at most one transfer at any time.

Several variations of the data migration problem have been studied. These variations arise either due to different objective functions or due to additional constraints. One common objective function is to minimize the *makespan* of the migration schedule, i.e., the time by which all migrations complete. Coffman *et al.* [7] introduced this problem. They showed that when edges may have arbitrary lengths, a class of greedy algorithms yields a 2-approximation to the minimum makespan. In the special case where the edges have equal (unit) lengths, the problem reduces to edge coloring of the transfer (multi)graph of the system for which an asymptotic approximation scheme is now known [22].

Hall *et al.* [10] studied the data migration problem with unit edge lengths and *capacity constraints*; that is, the migration schedule must respect the storage constraints of the disks. The paper gives a simple 3/2-approximation algorithm for the problem. The papers [10, 1] also present approximation algorithms for the makespan minimization problem with the following constraints: (*i*) data can only be moved, i.e, no new copies of a data object can be created, (*ii*) additional nodes can assist in data transfers, and (*iii*) each disk has a unit of spare storage. Khuller *et al.* [14] solved a more general problem, where each data object can also be copied. They gave a constant factor approximation algorithm for the problem.

Another objective function is to minimize the *average completion time* over all data migrations. This corresponds to minimizing the average edge completion time in the transfer graph. For the case of unit edges lengths, Bar-Noy *et al.* [3] showed that the problem is NP-hard and gave a simple 2-approximation algorithm. For arbitrary edge lengths, Halldórsson *et al.* [12] gave a 12-approximation algorithm for the problem. This was improved to 10 by Kim [15].

In this paper, we study the data migration problem with the objective of minimizing

the average completion time over all storage disks. Indeed, this objective favors the individual storage devices, which are often geographically distributed over a large network. It is therefore natural to try and minimize the average amount of time that each of these (independent) devices is involved in the migration process. For the case where vertices have arbitrary weights, and the edges have unit length, Kim [15] proved that the problem is NP-hard and showed that Graham's list scheduling algorithm [8], when guided by an optimal solution to a linear programming relaxation, gives an approximation ratio of 3. She also gave a 9-approximation algorithm for the case where edges have arbitrary lengths. We show that the analysis of the 3-approximation algorithm is tight, and for the case where edges have release times and arbitrary lengths, we give a 5.06-approximation algorithm.

A problem related to the data migration problem is non-preemptive *open shop scheduling*, denoted by $O|r_j|\sum w_jC_j$ in the standard three-field notation [16]. In this problem, we have a set of jobs, $\mathcal{J}$, and a set of machines $M_1, \ldots, M_m$. Each job $J_j \in \mathcal{J}$ consists of a set of m operations: $o_{j,i}$ has the processing time $p_{j,i}$ and must be processed on M_i, $1 \leq i \leq m$. Each machine can process a single operation at any time, and two operations that belong to the same job cannot be processed simultaneously. Also, each job J_j has a positive weight, w_j, and a release time, r_j, which means that no operation of J_j can start before r_j. The objective is to minimize the sum of weighted completion times of all jobs. This problem is MAX-SNP hard [13]. Chakrabarti *et al.* [5] gave a $(5.78 + \epsilon)$-approximation algorithm for the case where the number of machines, m, is some fixed constant. They also gave a $(2.89 + \epsilon)$-approximation algorithm for the preemptive version of the problem and fixed number of machines. For arbitrary number of machines, Queyranne and Sviridenko [20] presented algorithms that yield approximation factors of 5.83 and 3 for the non-preemptive and preemptive versions of the problems, respectively. The approximation factor for the preemptive version was subsequently improved to $(2 + \epsilon)$ by the same authors [19].

Our Contribution Since the open shop scheduling problem is a special case of the data migration problem all of our positive results for data migration apply to open shop scheduling. Note that the MAX-SNP hardness of the data migration problem follows from the MAX-SNP hardness of open shop scheduling [13]. Our main result is a 5.06-approximation algorithm for the data migration problem with arbitrary edge lengths. Our algorithm is based on rounding a solution of a linear programming (LP) relaxation of the problem. The general idea of our algorithm is that edges have to wait before they are actually processed (i.e., data transfer begins). This idea has been used for other scheduling problems as well [6, 12, 23]. Even though the high-level idea is similar to the one used in [12], there are subtle differences that are crucial to the improved results that we present here. Our method combines solutions obtained by using two different wait functions. It is interesting to note that each solution (when all edges are released at time 0) is a 5.83-approximate

solution, which is the approximation ratio obtained by Queyranne and Sviridenko [20]. In their algorithm artificial precedence constraints are defined and after that a schedule is obtained by a greedy algorithm whereas in our work artificial wait function for the edges (edges have to wait for some time before being processed) is defined after which a schedule is obtained using a greedy approach. To obtain an approximation ratio better than 5.83, we crucially use a property of the LP relaxation that we prove in Lemma 3.1. Although the LP relaxation has been used earlier [25, 18, 21, 11, 15, 20], we are not aware of any previous work that uses such a property of the LP. Our technique may be useful for deriving improved results for other shop scheduling problems.

For the case where edges have unit lengths, we show, by giving a tight example, that the list scheduling analysis of Kim [15] is tight. This illustrates the limitations of the LP relaxation. Finally, we study the open shop problem under *operations completion time* criteria (cf. [20]); that is, we sum the completion times of all operations for every job. For the special case of unit length operations with arbitrary non-negative weights, we show that an algorithm of [12] yields a 1.796 approximation algorithm for the problem.

2 Relation of Data Migration and Open Shop Scheduling

In this section, we formally state the data migration and open shop scheduling problems and show that the latter is a special case of the former.

Data Migration Problem We are given a graph $G = (V, E)$. The vertices represent storage devices, and the edges correspond to data transmissions among the devices. We denote by $E(u)$ the set of edges incident on a vertex u. Each vertex v has weight w_v and processing time 0. Each edge e has a length, or processing time, p_e. Moreover, each edge e can be processed only after its release time r_e. All release times and processing times are non-negative integers. The completion time of an edge is simply the time at which its processing is completed. Each vertex v can complete only after all the edges in $E(v)$ are completed. Since each vertex v has the processing time 0, the completion time, C_v, of v is the latest completion time of any edge in $E(v)$. The crucial constraint is that two edges incident on the same vertex cannot be processed at the same time. The objective is to minimize $\sum_{v \in V} w_v C_v$.

Open Shop Scheduling Problem We are given a set of jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$, to be scheduled on a set of machines $\mathcal{M} = \{M_1, \ldots, M_m\}$. Each job J_j has a non-negative weight w_j; also, J_j consists of a set of m operations $o_{j,1}, \ldots, o_{j,m}$, with the corresponding processing times $p_{j,i}$, $1 \leq i \leq m$; the operation $o_{j,i}$ must be processed on the machine M_i. Each machine can process at most one operation at any time, and no two operations

belonging to the same job can be processed simultaneously. The completion time C_j of each job J_j is the latest completion time of any of its operations. The objective is to minimize $\sum_{J_j \in \mathcal{J}} w_j C_j$.

The open shop scheduling problem is a special case of the data migration problem, as shown by the following reduction. Given an instance of the open shop scheduling problem, construct a bipartite graph $B = (J, M, F)$ as follows. Each vertex $j_\ell \in J$ represents a job $J_\ell \in \mathcal{J}$, and each vertex $m_i \in M$ represents a machine $M_i \in \mathcal{M}$. The edge $(j_\ell, m_i) \in F$ with processing time $p_{\ell,i}$ corresponds to the operation $o_{\ell,i}$, $1 \leq i \leq m$. Assign $w_{m_i} = 0$ to each vertex $m_i \in M$, and $w_{j_\ell} = w_\ell$ (i.e., the weight of the job J_ℓ) to each vertex $j_\ell \in J$. It is now easy to verify that any data migration schedule for B is a valid solution for the corresponding open shop problem.

In the remainder of the paper, we consider only the data migration problem, with the understanding that all of our results apply to open shop scheduling.

3 A Linear Programming Relaxation

The linear programming relaxation for the data migration problem (without release times) was given by Kim [15]. Such relaxations have been proposed earlier by Wolsey [25] and Queyranne [18] for single machine scheduling problems and by Schulz [21] and Hall *et al.* [11] for parallel machines and flow shop problems. For the sake of completeness, we state below the LP relaxation for the data migration problem.

For an edge e (vertex v) let C_e (C_v) be the variable that represents the completion time of e (resp., v) in the LP relaxation. For any set of edges $S \subseteq E$, let $p(S) = \sum_{e \in S} p_e$ and $p(S^2) = \sum_{e \in S} p_e^2$.

$$
\begin{aligned}
(LP) \qquad & \text{minimize} \quad \textstyle\sum_{v \in V} w_v C_v && && (1)\\
\text{subject to:} \quad & C_v \geq r_e + p_e, && \forall v \in V, e \in E(v) && (2)\\
& C_v \geq p(E(v)), && \forall v \in V && (3)\\
& C_v \geq C_e, && \forall v \in V, e \in E(v) && (4)\\
& \sum_{e \in S(v)} p_e C_e \geq \frac{p(S(v))^2 + p(S(v)^2)}{2}, && \forall v \in V, S(v) \subseteq E(v) && (5)\\
& C_e \geq 0, && \forall e \in E && (6)\\
& C_v \geq 0, && \forall v \in V \qquad . && (7)
\end{aligned}
$$

The set of constraints represented by (2), (3), and (4) are due to the different lower bounds on the completion times of a vertex. The justification for constraints (5) is as follows. By the problem definition, no two edges incident on the same vertex can be scheduled at

the same time. Consider any ordering of the edges in $S(v) \subseteq E(v)$. If an edge $e \in S(v)$ is the j-th edge to be scheduled among the edges in $S(v)$ then, setting $C_j = C_e$ and $p_j = p_e$, we get

$$\sum_{j=1}^{|S(v)|} p_j C_j \geq \sum_{j=1}^{|S(v)|} p_j \sum_{k=1}^{j} p_k = \sum_{j=1}^{|S(v)|} \sum_{k=1}^{j} p_j p_k = \frac{p(S(v))^2 + p(S(v)^2)}{2}.$$

Although there are exponentially many constraints, the above LP can be solved in polynomial time via the ellipsoid algorithm [18].

3.1 A Property of the LP

In this section, we state and prove a property of the LP that plays a crucial role in the analysis of our algorithm. Let $X(v, t_1, t_2) \subseteq E(v)$ denote the set of edges that complete in the time interval $(t_1, t_2]$ in the LP solution (namely, their fractional value belongs to this interval). Hall *et al.* [11] showed that $p(X(v, 0, t)) \leq 2t$. In Lemma 3.1 we prove a stronger property of a solution given by the above LP. Intuitively, our property states that if too many edges complete early, then the completion times of other edges must be delayed. For example, as a consequence of our property, for any $t > 0$ if $p(X(v, 0, t/2)) = t$ then $p(X(v, t/2, t)) = 0$, which means that no edges in $E(v) \setminus X(v, 0, t/2)$ complete before t in the LP solution. We now formally state and prove the lemma.

Lemma 3.1 *Consider a vertex v and times $t_1 > 0$ and $t_2 \geq t_1$. If $p(X(v, 0, t_1)) = \lambda_1$ and $p(X(v, t_1, t_2)) = \lambda_2$, then λ_1 and λ_2 are related by*

$$\lambda_2 \leq t_2 - \lambda_1 + \sqrt{t_2^2 - 2\lambda_1(t_2 - t_1)}$$

Proof: Using the constraint (5) of the LP relaxation for vertex v, we get

$$\begin{aligned}
\frac{p(X(v,0,t_2))^2}{2} &\leq \sum_{e \in X(v,0,t_1)} p_e C_e + \sum_{e \in X(v,t_1,t_2)} p_e C_e \\
&\leq p(X(v,0,t_1))t_1 + p(X(v,t_1,t_2))t_2 \\
\therefore (\lambda_1 + \lambda_2)^2 &\leq 2\lambda_1 t_1 + 2\lambda_2 t_2 \\
\therefore (\lambda_1 + \lambda_2 - t_2)^2 &\leq t_2^2 - 2\lambda_1(t_2 - t_1) \\
\therefore \lambda_2 &\leq t_2 - \lambda_1 + \sqrt{t_2^2 - 2\lambda_1(t_2 - t_1)}
\end{aligned}$$

□

The following result of [11] follows from Lemma 3.1 by substituting $t_1 = 0$, $\lambda_1 = 0$, and $t_2 = t$.

Corollary 3.2 *For any vertex v and time $t \geq 0$, $p(X(v, 0, t)) \leq 2t$.*

4 Algorithm

Note that if an edge has processing time 0, it can be processed as soon as it is released, without consuming any time steps. Hence, without loss of generality, we assume that the processing time of each edge is a positive integer.

The algorithm is parameterized by a *wait function* $W : E \to \mathcal{R}^+$. The idea is that each edge e must *wait* for W_e ($W_e \geq r_e$) time steps before it can actually start processing. The algorithm processes the edges (that have waited enough time) in non-decreasing order of their completion times in the LP solution. When e is being processed, we say that e is *active*. Once it becomes active, it remains active for p_e time steps, after which it is *finished*. A not-yet-active edge can be *waiting* only if none of its neighboring edges are active; otherwise, it is said to be *delayed*. Thus, at any time, an edge is in one of four modes: *delayed*, *waiting*, *active*, or *finished*. When adding new active edges, among those that have done their waiting duty, the algorithm uses the LP completion time as priority.

The precise rules are given in the pseudocode in Fig. 1. Let $wait(e,t)$ denote the number of time steps that e has waited until the end of time step t. Let $Active(t)$ be the set of active edges during time step t. Let $\widetilde{C_e}$ ($\widetilde{C_u}$) be the completion time of edge e (vertex u) in our algorithm. The algorithm in Fig. 1, implemented as is, would run in pseudo-polynomial time; however, it is easy to implement the algorithm in strongly polynomial time, by increasing t in each iteration by the smallest remaining processing time of an active edge.

One property of our processing rules, that distinguishes it from the wait functions used in [12] for the sum of edge completion times, is that multiple edges touching the same vertex can wait at the same time.

We run the algorithm for two different wait functions W and choose the better of the two solutions. For any vertex v (edge e), let C_v^* (C_e^*) be its completion time in the optimal LP solution. In the first wait function, for each edge e we choose $W_e = \lfloor \beta_1 C_e^* \rfloor$, $\beta_1 \geq 1$ and in the second one, we choose $W_e = \lfloor \beta_2 \max\{r_e, p(S_e(u)), p(S_e(v))\} \rfloor$, where $S_e(u) = \{f | f \in E(u), C_f^* \leq C_e^*\}$ and $\beta_2 \geq 1$. Note that the choice of wait functions ensures that the edges become active only after they are released. When all release times are 0, we can choose β_1 and β_2 such that $\beta_1 > 0$ and $\beta_2 > 0$.

5 Analysis

Consider a vertex x and an edge $e = (x, y)$, and recall that C_x^* and C_e^* are their completion times in the optimal LP solution. Let $B_e(x) = \{f | f \in E(x), C_f^* > C_e^*, \widetilde{C_f} < \widetilde{C_e}\}$, i.e., edges in $E(x)$ that finish after e in the LP solution, but finish before e in our algorithm. Recall that $S_e(x) = \{f | f \in E(x), C_f^* \leq C_e^*\}$. Note that $e \in S_e(x)$. Let $\overline{S_e(x)} = S_e(x) \setminus \{e\}$. By constraint (3), we have $p(S_e(x)) + p(B_e(x)) \leq C_x^*$.

SCHEDULE$(G = (V, E), W)$

Solve the LP relaxation for the given instance optimally.
$t \leftarrow 0$
$Finished \leftarrow Active(t) \leftarrow \emptyset$
for each $e \in E$ **do**
    $wait(e, t) \leftarrow 0$
while $(Finished \neq E)$ **do**
    $t \leftarrow t + 1$
    $Active(t) \leftarrow \{e \mid e \in Active(t-1)$ and $e \notin Active(t - p_e)\}$
    for each edge $e \in Active(t-1) \setminus Active(t)$ **do**
        $\widetilde{C_e} \leftarrow t - 1$
        $Finished \leftarrow Finished \cup \{e\}$ // e is *finished*
    for each edge $e = (u, v) \in E \setminus (Active(t) \cup Finished)$
        in non-decreasing order of LP completion time **do**
        if $(Active(t) \cap (E(u) \cup E(v)) = \emptyset)$ and $(wait(e, t-1) = W_e)$ **then**
            $Active(t) \leftarrow Active(t) \cup \{e\}$ // e is *active*
    for each edge $e = (u, v) \in E \setminus (Active(t) \cup Finished)$ **do**
        if $(Active(t) \cap (E(u) \cup E(v)) = \emptyset)$ **then**
            $wait(e, t) \leftarrow wait(e, t-1) + 1$ // e is *waiting*
        else $wait(e, t) \leftarrow wait(e, t-1)$ // e is *delayed*
return $\widetilde{C}$

Figure 1: Algorithm for Data Migration

We analyze our algorithm separately for the two wait functions defined in Section 4. In each case, we analyze the completion time of an arbitrary but fixed vertex $u \in V$. Without loss of generality, let $e_u = (u, v)$ be the edge that finishes last among the edges in $E(u)$. By constraint (4), we have $C^*_{e_u} \leq C^*_u$, since the LP solution we are considering is optimal, we can assume that $C^*_{e_u} = C^*_u$. We analyze our algorithm for the case where all edges in $\overline{S_{e_u}(u)} \cup \overline{S_{e_u}(v)}$ finish before e_u in our algorithm. If this is not true then our results can only improve. Let

$$p(S_{e_u}(v)) = \lambda_{e_u} C^*_{e_u} = \lambda_{e_u} C^*_u, \qquad 0 \leq \lambda_{e_u} \leq 2 \tag{8}$$

The upper bound on λ_{e_u} follows from Corollary 3.2.

For ease of reference, in Table 1 we summarize the key notation used in the rest of this section. Lemma 5.1 gives an upper bound on the completion time of any vertex in our algorithm for any wait function. Lemma 5.2 and Lemma 5.3 use Lemma 5.1 to give upper bounds for the two wait functions.

Symbol	Description
C_e^* (C_v^*)	completion time of edge e (vertex v) in the LP solution.
$\widetilde{C_e}$ ($\widetilde{C_u}$)	completion time of edge e (vertex v) in our algorithm.
$p(Z)$	$\sum_{e \in Z} p_e$.
r_e	release time of edge e.
$E(u)$	set of edges incident on vertex u.
$S_e(u)$	$\{f \mid f \in E(u) \text{ and } C_f^* \leq C_e^*\}$
$\overline{S_e(u)}$	$S_e(u) \setminus \{e\}$.
$B_e(u)$	$\{f \mid f \in E(u) \text{ and } C_f^* > C_e^* \text{ and } \widetilde{C_f} < \widetilde{C_e}\}$, i.e., edges in $E(u)$ that finish after e in the LP solution, but finish before e in our algorithm.
W_e	waiting time of edge e.
e_u	edge that finishes last in our algorithm among all edges in $E(u)$.

Table 1: Summary of notation

Lemma 5.1 $\widetilde{C_u} \leq W_{e_u} + C_u^* + p(S_{e_u}(v)) + p(B_{e_u}(v))$

Proof: Observe that when e_u is in delayed mode it must be that some edge in $\overline{S_{e_u}(u)} \cup B_{e_u}(u) \cup \overline{S_{e_u}(v)} \cup B_{e_u}(v)$ must be active. Hence, we have

$$\begin{aligned} \widetilde{C_{e_u}} &\leq W_{e_u} + p(S_{e_u}(u)) + p(B_{e_u}(u)) + p(S_{e_u}(v)) + p(B_{e_u}(v)) \\ \therefore \widetilde{C_u} &\leq W_{e_u} + C_u^* + p(S_{e_u}(v)) + p(B_{e_u}(v)) \end{aligned}$$

□

Define $f(\beta_1, \lambda) = \beta_1 + 1 + \frac{\beta_1+1}{\beta_1} + \sqrt{\left(\frac{\beta_1+1}{\beta_1}\right)^2 - \frac{2\lambda}{\beta_1}}$.

Lemma 5.2 *If* $W_{e_u} = \lfloor \beta_1 C_{e_u}^* \rfloor$ *then* $\widetilde{C_u} \leq f(\beta_1, \lambda_{e_u}) C_u^*$

Proof: Let $e_b \in B_{e_u}(v)$ be the edge with the largest completion time in the LP solution among all the edges in $B_{e_u}(v)$. Note that when e_b is in waiting mode it must be that either e_u is waiting or an edge in $\overline{S_{e_u}(u)} \bigcup B_{e_u}(u)$ is active (i.e., e_u is delayed). Thus, we get

$$W_{e_b} \leq W_{e_u} + p(\overline{S_{e_u}(u)}) + p(B_{e_u}(u)).$$

Hence, we have that

$$\lfloor \beta_1 C_{e_b}^* \rfloor \leq \lfloor \beta_1 C_{e_u}^* \rfloor + p(S_{e_u}(u)) - p_{e_u} + p(B_{e_u}(u)).$$

Since $p_{e_u} \geq 1$, it follows that $\beta_1 C^*_{e_b} - 1 \leq \beta_1 C^*_u + C^*_u - 1$, and $C^*_{e_b} \leq \frac{\beta_1+1}{\beta_1} C^*_u$.

Note that in the LP solution, each edge in $S_{e_u}(v)$ finishes at or before C^*_u and each edge in $B_{e_u}(v)$ finishes after C^*_u and at or before $C^*_{e_b}$. Thus, by substituting $t_1 = C^*_u, t_2 = \frac{\beta_1+1}{\beta_1} C^*_u$, $\lambda_1 = p(S_{e_u}(v)) = \lambda_{e_u} C^*_u$, and $\lambda_2 = p(B_{e_u}(v))$ in Lemma 3.1 we get

$$\begin{aligned} p(B_{e_u}(v)) &\leq \left(\frac{\beta_1+1}{\beta_1} - \lambda_{e_u} + \sqrt{\left(\frac{\beta_1+1}{\beta_1}\right)^2 - 2\lambda_{e_u}\left(\frac{\beta_1+1}{\beta_1} - 1\right)} \right) C^*_u \\ &= \left(\frac{\beta_1+1}{\beta_1} - \lambda_{e_u} + \sqrt{\left(\frac{\beta_1+1}{\beta_1}\right)^2 - \frac{2\lambda_{e_u}}{\beta_1}} \right) C^*_u \end{aligned}$$

Then, using (8), we have that

$$p(S_{e_u}(v)) + p(B_{e_u}(v)) \leq \left(\frac{\beta_1+1}{\beta_1} + \sqrt{\left(\frac{\beta_1+1}{\beta_1}\right)^2 - \frac{2\lambda_{e_u}}{\beta_1}} \right) C^*_u$$

The lemma now follows from Lemma 5.1 and the fact that $C^*_{e_u} = C^*_u$. □

Define $h(\beta_2, \lambda) = (\beta_2 + 1)\max\{1, \lambda\} + \frac{\beta_2+1}{\beta_2}$.

Lemma 5.3 *If* $W_{e_u} = \lfloor \beta_2 \max\{r_{e_u}, p(S_{e_u}(u)), p(S_{e_u}(v))\} \rfloor$ *then* $\widetilde{C_u} \leq h(\beta_2, \lambda_{e_u}) C^*_u$

Proof: By constraints (2) and (4) $r_{e_u} \leq C^*_{e_u} = C^*_u$, and from constraint (3), $p(S_{e_u}(u)) \leq C^*_u$. Also, recall that $p(S_{e_u}(v)) = \lambda_{e_u} C^*_u, \quad 0 \leq \lambda_{e_u} \leq 2$. Hence,

$$W_{e_u} \leq \lfloor \beta_2 \max\{C^*_u, \lambda_{e_u} C^*_u\} \rfloor. \tag{9}$$

In the following we upper bound $p(S_{e_u}(v)) + p(B_{e_u}(v))$. Let $z \in \overline{S_{e_u}(v)} \cup B_{e_u}(v)$ be the edge with the *largest waiting time*, i.e., $W_z = \max_{f \in \overline{S_{e_u}(v)} \cup B_{e_u}(v)} \{W_f\}$. When z is in waiting mode it must be that either e_u is waiting or an edge in $\overline{S_{e_u}(u)} \bigcup B_{e_u}(u)$ is active. Thus, using (9), we get

$$\begin{aligned} W_z &\leq W_{e_u} + p(\overline{S_{e_u}(u)}) + p(B_{e_u}(u)) \\ &\leq \lfloor \beta_2 \max\{C^*_u, \lambda_{e_u} C^*_u\} \rfloor + p(S_{e_u}(u)) - p_{e_u} + p(B_{e_u}(u)) \\ &\leq \beta_2 \max\{C^*_u, \lambda_{e_u} C^*_u\} + C^*_u - 1. \end{aligned} \tag{10}$$

Let l be the edge with the *largest completion time* in the LP solution among the edges in $\overline{S_{e_u}(v)} \cup B_{e_u}(v)$, i.e., $C^*_l = \max_{f \in \overline{S_{e_u}(v)} \cup B_{e_u}(v)} \{C^*_f\}$. Since $W_l \leq W_z$, we have

$$\lfloor \beta_2 (p(S_{e_u}(v)) + p(B_{e_u}(v))) \rfloor = \lfloor \beta_2 \cdot p(S_l(v)) \rfloor \leq W_l \leq W_z. \tag{11}$$

Combining (10) and (11) we get

$$\lfloor \beta_2(p(S_{e_u}(v)) + p(B_{e_u}(v))) \rfloor \quad \leq \quad \beta_2 \max\{C_u^*, \lambda_{e_u} C_u^*\} + C_u^* - 1.$$

Hence,

$$p(S_{e_u}(v)) + p(B_{e_u}(v)) \leq \left(\max\{1, \lambda_{e_u}\} + \frac{1}{\beta_2} \right) C_u^*.$$

Now, from Lemma 5.1 we have that

$$\widetilde{C_u} \leq W_{e_u} + C_u^* + (\max\{1, \lambda_{e_u}\} + \frac{1}{\beta_2}) C_u^*,$$

and from (9),

$$\widetilde{C_u} \leq \beta_2 C_u^* \cdot \max\{1, \lambda_{e_u}\} + C_u^* + C_u^* (\max\{1, \lambda_{e_u}\} + \frac{1}{\beta_2}).$$

This gives the statement of the lemma. □

Combining the two solutions To obtain an improved performance ratio, we run our algorithm with each of the two wait functions and return the better result. The cost of that solution is, by Lemmas 5.2 and 5.3, bounded by

$$ALG \leq \min \left(\sum_{v \in V} f(\beta_1, \lambda_v) C_v^*, \sum_{v \in V} h(\beta_2, \lambda_v) C_v^* \right).$$

This expression is unwieldy for performance analysis, but can fortunately be simplified. The following lemma allows us to assume without loss of generality that there are only two different λ-values: 0, or a constant $\hat{\lambda}$ in the range $[1, 2]$. To analyze the performance ratio, we then numerically optimize a function over three variables: β_1, β_2, and $\hat{\lambda}$.

We first introduce some notation. Let $OPT^* = \sum_v C_v^*$ be the cost of the optimal LP solution. Partition V into $V_0 = \{v \in V : \lambda_v \in [0, 1]\}$ and $V_1 = \{v \in V : \lambda_v \in (1, 2]\}$. Let $w = (\sum_{v \in V_0} C_v^*)/OPT^*$ be the contribution of V_0 to the LP solution. Given an instance with values λ_v, LP solution vector C_v^*, and the notation above, we define two functions: Let $F(t) = [w \cdot f(\beta_1, 0) + (1 - w) \cdot f(\beta_1, t)]$ and $H(t) = [w \cdot h(\beta_2, 0) + (1 - w) \cdot h(\beta_2, t)]$.

Lemma 5.4 *There is a $\hat{\lambda} \in [1, 2]$ such that $ALG \leq \min(H(\hat{\lambda}), F(\hat{\lambda}))OPT^*$.*

Proof: Since h is constant on $[0, 1]$ and f is decreasing, $h(\beta_2, \lambda_v) = h(\beta_2, 0)$ and $f(\beta_1, \lambda_v) \leq f(\beta_1, 0)$, for $v \in V_0$. Let $\hat{\lambda} = \left(\sum_{v \in V_1} \lambda_v C_v^* \right) / |V_1|$ be the average λ-value of nodes in V_1 weighted by their LP value. Since h is linear on $[1, 2]$, we have that $\sum_{v \in V_1} h(\beta_2, \lambda_v) C_v^* =$

$\sum_{v \in V_1} h(\beta_2, \hat{\lambda}) C_v^*$, and since f is convex, $\sum_{v \in V_1} f(\beta_1, \lambda_v) C_v^* \leq \sum_{v \in V_1} f(\beta_1, \hat{\lambda}) C_v^*$. It follows that

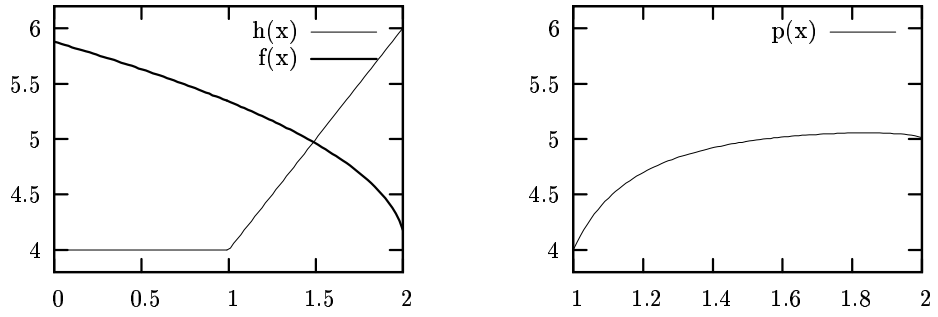
$$ALG \leq \sum_{v \in V} f(\beta_1, \lambda_v) C_v^* \leq \sum_{v \in V_0} f(\beta_1, 0) C_v^* + \sum_{v \in V_1} f(\beta_1, \hat{\lambda}) C_v^* = F(\hat{\lambda}) OPT^*.$$

The same type of bound holds for H, establishing the lemma. □

The worst case occurs when $h'(t) = f'(t)$; let $\hat{\lambda}$ denote that value of t. We can then determine the value of w from the other variables. Namely, defining $g(x) = g_{\beta_1,\beta_2}(x) = h(\beta_2, x) - f(\beta_1, x)$, we have $w = g(\hat{\lambda})/(g(\hat{\lambda}) - g(0))$. The performance ratio of the algorithm is then bounded by

$$\rho \leq \max_{\hat{\lambda} \in [1,2]} \left(h(\hat{\lambda}) + g(\hat{\lambda}) \frac{h(0) - h(\hat{\lambda})}{g(\hat{\lambda}) - g(0)} \right).$$

We find the best choice of parameters β_1 and β_2 numerically. When the release times can be non-zero, we need to ensure that each operation e does not begin executing before its release time. This is satisfied if the β-values are at least 1, ensuring that $W_e \geq r_e$. Setting $\beta_1 = 1.177$ and $\beta_2 = 1.0$, the worst-case is then achieved at about $\hat{\lambda} = 1.838$, giving a ratio of $\rho \leq 5.0553$.



(a) The functions $h(\lambda)$, $f(\lambda)$

(b) The performance ratio $p(x) = h(x) + g(x)(h(0) - h(x))/(g(x) - g(0))$ as a function of $\hat{\lambda}$

Figure 2: The performance functions when $\beta_1 = 1.177$ and $\beta_2 = 1.0$

Theorem 5.5 *There exists a* 5.06*-approximation algorithm for the data migration problem, as well as for the open shop scheduling problem.*

When all release times are zero, we can widen the search to all non-zero values. We then obtain a slightly improved ratio of 5.03, when choosing $\beta_1 = 1.125$ and $\beta_2 = 0.8$.

6 Unit Processing Times

It is known that the open shop scheduling problem is polynomially solvable when all operations are of unit length (c.f. [2]). On the other hand, Kim [15] showed that the more general data migration problem is NP-hard already in the case of unit edge lengths. For such instances, where all edges are released at time 0, it is shown in [15] that Graham's list scheduling algorithm [8], guided by an optimal solution to the LP relaxation[1] gives a 3-approximate solution; the algorithm is called *Ordered List Scheduling (OLS)*. The problem of obtaining a better than 3-approximate solution remained open. In Section 6.1, we show by giving a tight example that OLS cannot achieve a ratio better than 3. The tight example also illustrates the limitations of the LP solution. For the sake of completeness, we describe the OLS algorithm and its analysis here. The edges are sorted in non-decreasing order of their completion times in the LP solution. At any time, an edge $e = (u, v)$ is scheduled iff no edge in $S_e(u) \cup S_e(v)$ is scheduled at that time. (Recall that $S_e(u) = \{f | f \in E(u), C^*_f \leq C^*_e\}$.) For any vertex u, if e_u is the edge that finishes last among the edges in $E(u)$, and if $\widetilde{C_u}$ is the completion time of u in OLS, then $\widetilde{C_u} \leq p(S_{e_u}(u)) + p(S_{e_u}(v))$. Combining the fact that $p(S_{e_u}(u)) \leq C^*_u$ along with $p(S_{e_u}(v)) \leq 2C^*_{e_u} = 2C^*_u$ (Corollary 3.2), we get $\widetilde{C_u} \leq 3C^*_u$ and hence a 3-approximation ratio.

6.1 A tight example

Consider a tree rooted at vertex r. Let $S = \{s_1, s_2, \ldots, s_k\}$ be the children of r. For each vertex s_i, let $L_i = \{l^i_1, l^i_2, \ldots, l^i_h\}$ be the children of s_i. Let $L = \cup_{i=1}^k L_i$. Let $k = (n+1)/2$ and $h = n - 1$. For each vertex $u \in S$, let $w_u = \epsilon$ and for each vertex $v \in L$, let $w_v = 0$. Let $w_r = M$. For each edge e, let $C^*_e = (n+1)/2$ be its completion time in the LP solution. For each vertex $v \in L \cup \{r\}$, $C^*_v = (n+1)/2$ and for each vertex $v \in S$, $C^*_v = n$. The completion times of vertices in L do not matter as the weights of all those vertices are zero. It is easy to verify that this is an optimal LP solution. The cost of the LP solution equals

$$w_r \left(\frac{n+1}{2}\right) + \sum_{i=1}^{k} w_{s_i} n \quad = \quad M\left(\frac{n+1}{2}\right) + k\epsilon n = M\left(\frac{n+1}{2}\right) + \left(\frac{n(n+1)}{2}\right)\epsilon \ .$$

OLS could process the edges in the following order. At any time t, $1 \leq t \leq n-1$, OLS processes all edges in $\{(s_1, l^1_t), (s_2, l^2_t), \ldots, (s_k, l^k_t)\}$. At time $t = n + z$, $0 \leq z < (n+1)/2$, OLS processes edge (r, s_{z+1}). The cost of the solution in OLS is at least

$$w_r \left(n - 1 + \frac{n+1}{2}\right) + \sum_{i=1}^{k} w_{s_i} n = M\left(\frac{3n-1}{2}\right) + \left(\frac{n(n+1)}{2}\right)\epsilon \ .$$

For large n, if $M \gg \epsilon$, the ratio of the cost of the OLS solution to the cost of the LP solution approaches 3.

[1] See in Section 3).

6.2 Open shop and sum of operation completion times

Consider now the open shop problem, where each operation has unit processing time and a non-negative weight, and the objective is to minimize the weighted sum of completion times of all operations. Even though the problem of minimizing the sum of job completion times is polynomially solvable for open shop scheduling, minimizing the weighted sum of completion times of unit operations is NP-hard. This holds even for uniform weights as recently proved in [17]. We relate this problem to a result of [12] for the *sum coloring* problem. The input to sum coloring is a graph G, where each vertex corresponds to a unit length job. We need to assign a positive integer (color) to each vertex (job) so as to minimize the sum of the colors over all vertices. The constraint is that adjacent vertices receive distinct colors. In the weighted case, each vertex (job) is associated with a non-negative weight, and the goal is to minimize the weighted sum of the vertex colors.

In the *maximum k-colorable subgraph* problem, we are given an undirected graph G and a positive integer k; we need to find a maximum size subset $U \subseteq V$ such that $G[U]$, the graph induced by U, is k-colorable. In the weighted version, each vertex has a non-negative weight and we seek a maximum weight k-colorable subgraph. The following theorem is proved in [12].

Theorem 6.1 *The weighted sum coloring problem admits a* 1.796 *ratio approximation algorithm on graphs for which the maximum weight k-colorable subgraph problem is polynomially solvable.*

We can relate this theorem to the above variant of the open shop problem, by defining the bipartite graph $B = (J, M, F)$ (see in Section 2) and setting $G = L(B)$, i.e., G is the line graph of B. Recall that in $L(B)$ the vertices are the edges of B; two vertices are neighbors if the corresponding edges in B share a vertex.

In order to apply Theorem 6.1, we need to show that the maximum weight k-colorable subgraph problem is polynomial on $L(B)$. Note that this is the problem of finding a maximum weight collection of edges in B that is k-colorable (i.e., can be decomposed into k disjoint matchings in B). Observe that, on bipartite graphs, this problem is equivalent to the well-known weighted *b-matching* problem. In weighted b-matching, we seek a maximum weight set of edges that induces a subgraph of maximum degree at most k. Recall that a bipartite graph always admits a matching touching every vertex of maximum degree (c.f. [9]). It follows, that the chromatic index of a bipartite graph is equal to its maximum degree. Since weighted b-matching is solvable in polynomial time (c.f. [4]), the same holds for the weighted k-colorable subgraph problem on $L(B)$. Hence, we have shown

Theorem 6.2 *Open shop scheduling of unit jobs, under weighted sum of operation completion time criteria, admits a* 1.796 *ratio approximation.*

Acknowledgments The first author would like to thank Yoo-Ah Kim for introducing to the author the problem of data migration, and Samir Khuller, Yoo-Ah Kim, Aravind Srinivasan, Chaitanya Swamy for useful discussions.

References

[1] E. Anderson, J. Hall, J. Hartline, M. Hobbes, A. Karlin, J. Saia, R. Swaminathan, and J. Wilkes. *An Experimental Study of Data Migration Algorithms.* Workshop on Algorithm Engineering, pages 145–158, 2001.

[2] P. Brucker, *Scheduling Algorithms. 3rd edition*, Springer Verlag, 2001.

[3] A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, and T. Tamir. *On Chromatic Sums and Distributed Resource Allocation.* Information and Computation, 140:183–202, 1998.

[4] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. Combinatorial Optimization. John Wiley and Sons, 1998.

[5] S. Chakrabarti, C. A. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein. *Improved Scheduling Problems For Minsum Criteria.* Proc. of the 23rd International Colloquium on Automata, Languages, and Programming, LNCS 1099, pages 646–657, 1996.

[6] C. Chekuri, R. Motwani, B. Natarajan, and C. Stein. *Approximation Techniques for Average Completion Time Scheduling.* SIAM Journal on Computing, 31(1):146–166, 2000.

[7] E. G. Coffman, M. R. Garey, D. S. Johnson, and A. S. LaPaugh. *Scheduling File Transfers.* SIAM Journal on Computing, 14(3):744–780, 1985.

[8] R. Graham. *Bounds for certain multiprocessing anomalies.* Bell System Technical Journal, 45:1563–1581, 1966.

[9] H. Gabow and O. Kariv. *Algorithms for edge coloring bipartite graphs and multigraphs.* SIAM Journal of Computing, 11(1):117–129, 1982.

[10] J. Hall, J. Hartline, A. Karlin, J. Saia, and J. Wilkes. *On Algorithms for Efficient Data Migration.* Proc. of the 12th ACM-SIAM Symposium on Discrete Algorithms, pages 620–629, 2001.

[11] L. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. *Scheduling to Minimize Average Completion Time: Off-line and On-line Approximation Algorithms.* Mathematics of Operations Research, 22:513–544, 1997.

[12] M. M. Halldórsson, G. Kortsarz, and H. Shachnai. *Sum Coloring Interval Graphs and k-Claw Free Graphs with Applications for Scheduling Dependent Jobs.* Algorithmica, 37:187–209, 2003.

[13] H. Hoogeveen, P. Schuurman, and G. Woeginger. *Non-approximability Results For Scheduling Problems with Minsum Criteria.* Proc. of the 6th International Conference on Integer Programming and Combinatorial Optimization, LNCS 1412, pages 353–366, 1998.

[14] S. Khuller, Y. Kim, and Y. C. Wan. *Algorithms for Data Migration with Cloning.* In Proc. of the 22nd ACM Symposium on Principles of Database Systems, pages 27–36, 2003.

[15] Y. Kim. *Data Migration to Minimize the Average Completion Time.* Proc. of the 14th ACM-SIAM Symposium on Discrete Algorithms, pages 97–98, 2003.

[16] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan, and D. B. Shmoys. Sequencing and Scheduling: Algorithms and Complexity. In S. C. Graves et al, eds., *Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory*, 445–522, 1993.

[17] Dániel Marx. Complexity results for minimum sum edge coloring. Manuscript, 2004. See URL: `http://www.cs.bme.hu/~dmarx/publications.html`

[18] M. Queyranne. *Structure of a Simple Scheduling Polyhedron.* Mathematical Programming, 58:263-285, 1993.

[19] M. Queyranne and M. Sviridenko. *A $(2+\epsilon)$-Approximation Algorithm for Generalized Preemptive Open Shop Problem with Minsum Objective.* Journal of Algorithms, 45:202–212, 2002.

[20] M. Queyranne and M. Sviridenko. *Approximation Algorithms for Shop Scheduling Problems with Minsum Objective.* Journal of Scheduling, 5:287–305, 2002.

[21] A. S. Schulz. *Scheduling to Minimize Total Weighted Completion Time: Performance Guarantees of LP-based Heuristics and Lower Bounds.* Proc. of the 5th International Conference on Integer Programming and Combinatorial Optimization, LNCS 1084, pages 301–315, 1996.

[22] P. Sanders and D. Steurer. *An Asymptotic Approximation Scheme for Multigraph Edge Coloring.* Proc. of the 16th ACM-SIAM Symposium on Discrete Algorithms, 2005.

[23] M. Skutella and M. Uetz. *Stochastic Machine Scheduling with Precedence Constraints.* SIAM Journal on Computing 34(4), 788-802, 2005

[24] V. G. Timkovsky. Identical parallel machines vs. unit-time shops, preemptions vs. chains, and other offsets in scheduling complexity, Technical Report, Dept. of Comp.Sc. and Systems, McMaster Univ. Hamilton.

[25] L. Wolsey. *Mixed Integer Programming Formulations for Production Planning and Scheduling Problems.* Invited talk at the 12th International Symposium on Mathematical Programming, MIT, Cambridge, 1985.