Combinatorial Algorithms for Data Migration to Minimize Average Completion Time[*]

Rajiv Gandhi[1] and Julián Mestre[2]

[1] Department of Computer Science, Rutgers University-Camden, Camden, NJ 08102.
Research partially supported by Rutgers University Research Council Grant.
E-mail: `rajivg@camden.rutgers.edu`.
[2] Max-Planck-Institute für Informatik, Saarbrücken, Germany. Research done at the University of Maryland;
supported by NSF Awards CCR-0113192 and CCF-0430650, and the University of Maryland Dean's Dissertation
Fellowship.
E-mail: `jmestre@cs.umd.edu`.

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. It is modeled by a transfer graph, where vertices represent the storage devices, and edges represent data transfers required between pairs of devices. Each vertex has a non-negative weight, and each edge has a processing time. A vertex completes when all the edges incident on it complete; the constraint is 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 (*Journal of Algorithms, 55:42-57, 2005*) gave an LP-rounding 3-approximation algorithm when edges have unit processing times. We give a more efficient primal-dual algorithm that achieves the same approximation guarantee. When edges have arbitrary processing times we give a primal-dual 5.83-approximation algorithm. We also study a variant of the open shop scheduling problem. This is a special case of the data migration problem in which the transfer graph is bipartite and the objective is to minimize the sum of completion times of edges. We present a simple algorithm that achieves an approximation ratio of $\sqrt{2} \approx 1.414$, thus improving the 1.796-approximation given by Gandhi *et al.* (*ACM Transaction on Algorithms, 2(1):116-129*, 2006). We show that the analysis of our algorithm is almost tight.

1 Introduction

The *data migration* problem arises in large storage systems, such as *Storage Area Networks* [13], 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 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.* [4] show that when the edges have unit processing times, the problem reduces to edge coloring of the transfer (multi)graph of the system. The best approximation algorithm known for minimum edge coloring [18] then yields an algorithm for data migration with unit edge processing times, whose makespan is $1.1\chi' + 0.8$,

---

[*] A preliminary version of the paper appeared in the Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006.

where χ' is the chromatic index of the graph. Approximation algorithms are also developed [9, 1, 13, 14] for generalizations of the makespan minimization problem in which there are storage constraints on disks and constraints on how the data can be transferred.

The data migration problem has also been studied with the objective of minimizing the sum of weighted completion time over all storage disks. Kim [15] proved that the problem is NP-hard when edges have unit processing times 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. Gandhi *et al.* [6] show that Kim's analysis is tight. Very recently, Mestre [17] improves the approximation ratio to $1 + \phi$, where ϕ is the Golden ratio; his approach builds on the algorithm presented in this paper. When edges have arbitrary processing times Kim [15] gave a 9-approximate solution. Gandhi *et al.* [6] improved the approximation factor to 5.03. They present two algorithms each achieving an approximation ratio of 5.83 and show that combining the two solutions yields an approximation ratio of 5.03.

A problem related to the data migration problem is *open shop scheduling*. 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_j operations. For $1 \leq i \leq m_j$, operation $o_{j,i}$ has processing time $p_{j,i}$ and must be processed on $M_{\phi(j,i)}$. Each machine can process a single operation at any time, and two operations that belong to the same job cannot be processed simultaneously. Each job J_j has a positive weight, w_j and the objective is to minimize the sum of weighted completion times of all jobs. This problem is a special case of the data migration problem [6]. Open shop scheduling problem has been studied in [3, 12, 20, 21].

There has also been interest in the study of data migration problem with the objective function being to minimize the average completion time over all data transfers. This corresponds to minimizing the average edge completion time in the transfer graph. For arbitrary edge processing times, several constant factor approximation algorithms [11, 15, 7] are known with the best approximation factor being 7.682 [7]. For the case of unit processing times, Bar-Noy *et al.* [2] showed that the problem is NP-hard and gave a simple 2-approximation algorithm. When restricted to bipartite graphs, the latter problem becomes a variant of open shop scheduling in which the operations have unit processing times and the objective is to minimize the sum of completion times of operations; for this problem Gandhi *et al.* [6] give a 1.796-approximate solution that uses a sum coloring algorithm due to Halldórsson *et al.* [11].

1.1 Our Contribution

First we study the data migration problem with the objective of minimizing the average completion time over all storage disks. Kim [15] gave approximation algorithms for this problem: A 3-approximation when edges have unit processing times and a 9-approximation for the general case. Kim's algorithms round the solution produced by a linear programming relaxation for the problem. This algorithm involves solving a linear program with an exponential number of constraints, though there are equivalent linear programs with a polynomial number of constraints (cf. [7]). Gandhi *et al.* [6] show that Kim's algorithm can not give an approximation guarantee better than 3. The best approximation guarantee for the general case is 5.03 [6] which is obtained by combining solutions to two algorithms each of which yields an approximation guarantee of 5.83. At a very high level all known algorithms for this problem comprise of the following two steps: (i) label the edges, (ii) consider the edges for scheduling in sorted order of their labels. The algorithms in [15, 6] label the edges based on the fractional completion times in the linear programming solution. In this work we present simple and efficient primal-dual algorithms, which constitute the first purely combinatorial algorithms for the problem. A novel aspect of our approach is that, unlike typical primal-dual algorithms, the primal solution is not constructed using (relaxed) complementary slackness. Rather, in step (i), the dual update assigns labels to the edges. These labels are used, in step (ii), to guide a scheduling procedure, which is almost identical to that in [15] for unit edge processing times and that in [6] for arbitrary edge processing times. In the analysis, the cost of the schedule is related to the cost of the dual solution via the labels, yielding a 3-approximation when edges have unit processing times, and a 5.83-approximation when edges have arbitrary processing times.

The second problem we study is the data migration problem with the objective of minimizing the sum of completion times of edges. In other words, given a graph $G = (V, E)$ we want to partition the edge set

E into matchings $M_1, M_2, \ldots$, so as to minimize $\sum_i i|M_i|$. A schedule is *minimal* if M_i is maximal with respect to $G \setminus \cup_{j<i} M_j$. Bar-Noy *et al.* [2] show that any minimal schedule is 2-approximate. In this work, we introduce the notion of *strongly minimal* schedules. A schedule is strongly minimal if for all $b \geq 1$, the b-matching $\cup_{j \leq b} M_j$ is maximal in G, where b-matching is a subset of edges in G such that each vertex has at most b of its incident edges in the matching. We show that any strongly minimal schedule is $\sqrt{2}$-approximate, and that such schedule always exist for bipartite graphs and can be computed in polynomial time. Data migration in bipartite graphs is equivalent to a variant of open shop scheduling in which we want to minimize the sum of operation completion times. Marx [16] has shown that the problem is APX-hard. Gandhi *et al.* [6] show that using the sum-coloring algorithm of Halldórsson *et al.* [11] one can obtain a 1.796 approximation guarantee. Thus, we improve this ratio to $\sqrt{2} \approx 1.414$, though our guarantee does not extend to the objective of minimizing the sum of weighted edge completion times. We also show that the analysis is almost tight by giving an example on which the algorithm gives a 1.375-approximate solution.

2 Data Migration Problem

We are given a graph $G = (V, E)$. All our graphs are multigraphs, for simplicity, however, we drop the multi prefix when talking about graphs and sets of edges. Let $E(u)$ denote the set of edges incident on a vertex u. The vertices and edges in G are jobs to be completed. Each vertex v has weight w_v and each edge has processing time p_e. The completion time, C_e, of edge e is simply the time at which its processing is completed. The completion time, C_v, of vertex 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 $\sum_{v \in V} w_v C_v$.
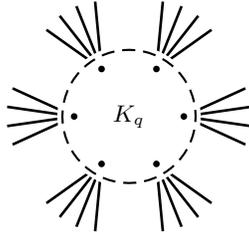

Fig. 1. An instance for the data migration problem.

To get more insight into the problem, consider the following natural and intuitive algorithm for the case when edges have unit processing times: Process the edges in any order scheduling them as early as possible without creating conflicts with the edges scheduled so far. While this algorithm gives a solution that is at most twice the cost of optimal for $\min \sum_e C_e$ [2], the following example shows that for the objective of $\min \sum_v C_v$ it may produce a solution with cost $\Omega(\sqrt[3]{n})$ times the optimum.

Consider a complete graphs on q vertices. To this, attach q copies of $K_{1,\sqrt{q}}$, so that each node in K_q is the center of one of the stars, as shown in Figure 1. The optimal solution first schedules the stars in parallel and then the edges in K_q, with a total cost of $\Theta(q^2)$. On the other hand, a minimal solution may schedule the edges in K_q before the stars, incurring an overall cost of $\Theta(q^{2.5})$. Since the graph has $\Theta(q^{1.5})$ vertices, this shows that a minimal schedule can be a factor $\Omega(\sqrt[3]{n})$ away from the optimum.

2.1 A Linear Programming Relaxation

The linear programming relaxation for the data migration problem was given by Kim [15]. Such relaxations have been proposed earlier by Wolsey [23] and Queyranne [19] for single machine scheduling problems and by Schulz [22] and Hall *et al.* [10] for parallel machines and flow shop problems. For the purpose of clarity, we state only portions of the LP relaxation relevant for obtaining the primal-dual algorithm.

For a vertex v, let C_v represent the completion time of v. Let $N(u)$ represent the set of neighbors of vertex u and $E(u)$ the set of edges incident on u. For any edge $e = (u, v)$, we use p_e to denote the processing time of e. For any set $F \subseteq E$, let $p(F) = \sum_{e \in F} p_e$.

$$\min \sum_{v \in V} w_v\, C_v$$

subject to

$$\sum_{e=(u,v) \in S(u)} p_e\, C_v \geq \frac{p(S(u))^2 + p(S(u)^2)}{2} \qquad \forall u \in V, S(u) \subseteq E(u) \quad (1)$$

$$C_v \geq p(E(v)) \qquad \forall v \in V \qquad (2)$$

The justification for constraints (1) 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 $S(u) \subseteq E(u)$. If an edge $e \in S(u)$ is the jth edge to be scheduled among the edges in $S(u)$ then, setting $C_j = C_e$ and $p_j = p_e$, we get

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

Combining this with the fact that $C_v \geq C_e$, for all $v \in V$ and $e \in E(v)$ gives us the set of constraints (1).

The dual LP contains a variable $y_{S(u)}$ for each set $S(u)$, corresponding to constraint (1), and a variable z_v for each $v \in V$, corresponding to constraint (2) . The dual LP is given below.

$$\max \sum_{\substack{u \in V \\ S(u) \subseteq E(u)}} \frac{p(S(u))^2 + p(S(u)^2)}{2}\, y_{S(u)} + \sum_{v \in V} p(E(v)) z_v$$

subject to

$$z_v + \sum_{\substack{e=(u,v) \\ S(u):e \in S(u)}} y_{S(u)}\, p_e \leq w_v \qquad \forall v \in V \qquad (3)$$

$$y_{S(u)} \geq 0 \qquad \forall u \in V, S(u) \subseteq E(u) \qquad (4)$$

$$z_v \geq 0 \qquad \forall v \in V \qquad (5)$$

3   Algorithm

The algorithm consists of two phases—*labeling* and *scheduling*. In the labeling phase, the vertices receive labels which are then used to label the edges. The scheduling phase uses the edge labels to decide the order in which the edges must be considered for processing.

3.1   Labeling Phase

The high level idea of the labeling algorithm is as follows. Initially, each vertex is unlabeled and the dual variables are set to 0. The algorithm proceeds in iterations. In each iteration we find a vertex x which maximizes the total processing times of edges going from x to unlabeled neighbors of x, call this set of edges $S(x)$. If there exists an unlabeled node h such that $p(E(h)) > p(S(x))$, then assign h a label of $p(S(x))$ and raise z_h until constraint (3) is tight, namely,

$$z_h = w_h - \sum_{\substack{e=(u,h) \\ S(u):e \in S(u)}} y_{S(u)}\, p_e.$$

Otherwise, the value of the dual variable $y_{S(x)}$ is increased until the dual constraint (3) is met with equality for some unlabeled neighbor v of x. In other words, $y_{S(x)}$ assumes the smallest value such that for some vertex $v \in N(x)$ such that $(x, v) \in S(x)$ we have

$$\sum_{\substack{e=(u,v) \\ S(u):e\in S(u)}} y_{S(u)}\, p_e \;=\; w_v.$$

All unlabeled neighbors of x for which the above equality holds are labeled $p(S(x))$. Note that the dual variables are incremented in such a way that no constraint in the dual LP is violated.

The label of vertex u is denoted by $\ell(u)$. The label $\ell(e)$ of an edge $e = (u, v)$ is the pair $(\ell(u), \ell(v))$. For any two edges $e = (u, v)$ and $f = (u', v')$, $\ell(e) \leq \ell(f)$ if:

(i) $\min\{\ell(u), \ell(v)\} < \min\{\ell(u'), \ell(v')\}$, or

(ii) $\min\{\ell(u), \ell(v)\} = \min\{\ell(u'), \ell(v')\}$ and $\max\{\ell(u), \ell(v)\} \leq \max\{\ell(u'), \ell(v')\}$.

LABEL($G = (V, E)$)

```
1     for each v ∈ V do
2         ℓ(v) ← nil                    // v is unlabeled
3     while there exists an unlabeled vertex do
4         let x be a vertex maximizing p(S(x)),
          where S(x) = {(x, v) ∈ E(x) | ℓ(v) = nil}
5         let h be an unlabeled vertex maximizing p(E(h))
6         if p(E(h)) > p(S(x)) then
7             z_h ← w_h
8             ℓ(h) ← p(S(x))     // h is now labeled
9         else
10            y_{S(x)} ← min { w_v / ∑_{e=(x,v)} p_e | v ∈ N(x) s.t. ℓ(v) = nil }
11            for each v ∈ N(x) s.t. ℓ(v) = nil do
12                w_v ← w_v − y_{S(x)} ∑_{e=(x,v)} p_e
13                if w_v = 0 then
14                    ℓ(v) ← p(S(x))     // v is now labeled
15    for each e = (u, v) ∈ E do
16        ℓ(e) ← (ℓ(u), ℓ(v))
17    return ℓ
```

Fig. 2. Labeling procedure.

The pseudo-code for the labeling phase is given in Figure 2. We now state and prove some important properties of the labels and the dual solution generated by our algorithm.

Lemma 1. *Consider an edge $e = (v, w)$. Let $F_e(w) = \{f \in E(w) \mid \ell(f) \leq \ell(e)\}$. Then, $p(F_e(w)) \leq \ell(v)$.*

Proof. Let $N_e(w) = \{y \in N(w) \mid (w, y) \in F_e(w)\}$. Note that $e \in F_e(w)$ and for each vertex $y \in N_e(w)$ we have $\ell(y) \leq \ell(v)$. Consider the iteration of the algorithm in which the first vertex in $N_e(w)$ is labeled, and let y be that vertex. At the beginning of this iteration the sum of the processing times of edges in $E(w)$ whose other endpoints are unlabeled is at least $p(F_e(w))$. Notice that x is chosen (line 4 in the pseudo-code LABEL) to be the vertex with the maximum value of $p(S(x))$, where $S(x)$ is set of edges incident to x whose other endpoints are unlabeled. Therefore, it must be the case that $p(F_e(w)) \leq p(S(x)) = \ell(y)$. $\square$

Lemma 2. *For any $h \in V$, if $z_h > 0$ then $p(E(h)) > \ell(h)$.*

5

Proof. In the pseudo-code, variable z_h is set in line 7. In order to reach line 7, the condition of if statement in line 6 must be satisfied. Thus, $p(E(h)) > \ell(h)$. □

Lemma 3. *For any $v \in V$ and $S(x)$ such that $(x, v) \in S(x)$, if $y_{S(x)} > 0$, $\max\{\ell(v), p(E(v))\} \leq p(S(x))$.*

Proof. Consider the iteration in which $y_{S(x)}$ was set. In the pseudo-code, variable $y_{S(x)}$ is set in line 10. In order to reach line 10, it must be that in this iteration all unlabeled vertices have degree at most $p(S(x))$. In particular, since v was unlabeled at the beginning of the iteration, $p(E(v)) \leq p(S(x))$. If vertex v is labeled in this iteration then $\ell(v) = p(S(x))$. Otherwise v is labeled in a later iteration. Since the value of the labels assigned by the algorithm only decreases with time, we have $\ell(v) \leq p(S(x))$. □

3.2 Scheduling edges with unit processing times

The scheduling phase is almost the same as the list scheduling algorithm [8] used by Kim [15]. The only difference is in the entity that is used to decide the order in which the edges are processed. We use the edge labels generated by algorithm LABEL to decide the ordering whereas Kim [15] uses the completion times of edges as returned by the linear programming solution.

The edges are sorted in increasing order of their labels. The edges are then processed in this order. When processing $(u, v) \in E$, the edge is scheduled at the earliest time such that no edge incident upon u or v is already scheduled at that time. The pseudo-code is given in Figure 3.

SCHEDULE$(G = (V, E), \ell)$

1   sort edges in lexicographic order of their labels
2   **for** each $e = (u, v) \in E$, processed in sorted order **do**
3   schedule e as early as possible
            without creating conflicts with edges scheduled so far.

Fig. 3. Scheduling with unit processing times

In order to gain some intuition, we trace the execution of LABEL and SCHEDULE on the unweighted instance from Figure 1. The transfer graph consists of q stars $(K_{1,\sqrt{q}})$ whose centers are fully connected, i.e., they induce a K_q. The procedure LABEL starts by choosing a star center a and sets the labels of its neighbors to $q + \sqrt{q} - 1$; then a gets a label of $\sqrt{q} + 1$ and the remaining nodes get a label of $\sqrt{q}$. As we saw at the beginning of Section 2 the optimal solution first schedules the star edges and then schedules the K_q edges. When SCHEDULE sorts the edge set, only the star edges incident on a may come before some of the K_q edges; thus the algorithm produces a schedule that is close to optimal.

Let a be the first node chosen by LABEL and let A be the set of its neighbors. In *any* schedule the collective finishing time of nodes in A must be at least $|A|^2/2 = (q + \sqrt{q} - 1)^2/2$. The finishing time of A in *our* schedule will be charged to this quantity. To avoid overcharging, the nodes in A are labeled with $|A| = q + \sqrt{q} - 1$. Let $b \neq a$ be a star center and B be its $\sqrt{q}$ star neighbors. When LABEL choses b the set of unlabeled neighbors of b is B. Their collective finishing time in any solution is at least $|B|^2/2 = q/2$. Since this quantity should pay for the finishing time of B, these nodes are given a label of $|B| = \sqrt{q}$. Notice that the budget to pay for A is $O(q)$ larger than the budget to pay for B. Hence, in our schedule we would like the nodes in B to finish earlier than the nodes in A. Indeed, the procedure SCHEDULE is set up so that the finishing time of a vertex is not much greater than its given label.

3.3 Analysis when edges have unit processing times

Let $\widetilde{C}_v$ be the completion time of vertex v in our algorithm. Recall that $E(v)$ denotes the set of edges incident on a vertex v, and $N(v)$ the set of neighbors of v.

Lemma 4. *For each $v \in V$, $\widetilde{C}_v \leq \ell(v) + |E(v)| - 1$.*

Proof. Let $e_v = (w, v)$ be the last edge to finish among the edges in $E(v)$. Recall from Lemma 1 that $F_{e_v}(w) = \{f \in E(w) \,|\, \ell(f) \leq \ell(e_v)\}$. Observe that because of the order in which the edges are scheduled, we have

$$\widetilde{C}_v \leq |F_{e_v}(w)| + |E(v)| - 1.$$

From Lemma 1 we know that $p(F_{e_v}(w)) = |F_{e_v}(w)| \leq \ell(v)$. Substituting this in the above expression completes the proof. □

Theorem 1. *The data migration problem with edges having unit processing times has a 3-approximate primal-dual algorithm.*

Proof. Let $G = (V, E)$ be an instance of the data migration problem. The cost of the schedule found by our algorithm is given by

$$\sum_{v \in V} w_v \widetilde{C}_v \leq \sum_{v \in V} w_v \left(\ell(v) + |E(v)| \right) \qquad [\text{ using Lemma 4 }]$$

$$= \sum_{v \in V} w_v \, \ell(v) + \sum_{v \in V} w_v \, |E(v)|$$

Clearly $\sum_{v \in V} w_v |E(v)| \leq OPT(G)$. Thus, if we can show that $\sum_{v \in V} w_v \ell(v) \leq 2\, OPT(G)$ we are done. To that end, we relate $\sum_{v \in V} w_v \ell(v)$ to the cost of the dual feasible solution obtained by the algorithm, which we denote by $DFS(G)$.

$$\sum_{v \in V} w_v \, \ell(v) = \sum_{v \in V} \left(z_v + \sum_{\substack{e=(u,v) \\ S(u):e \in S(u)}} y_{S(u)} \right) \ell(v) \qquad [\text{ every vertex is tight }]$$

$$= \sum_{v \in V} z_v \, \ell(v) \; + \sum_{v \in V} \sum_{\substack{e=(u,v) \\ S(u):e \in S(u)}} y_{S(u)} \, \ell(v)$$

$$\leq \sum_{v \in V} z_v \, |E(v)| \; + \sum_{v \in V} \sum_{\substack{e=(u,v) \\ S(u):e \in S(u)}} y_{S(u)} \, \ell(v) \qquad [\text{ using Lemma 2 }]$$

$$\leq \sum_{v \in V} z_v \, |E(v)| \; + \sum_{v \in V} \sum_{\substack{e=(u,v) \\ S(u):e \in S(u)}} y_{S(u)} \, |S(u)| \qquad [\text{ using Lemma 3 }]$$

$$= \sum_{v \in V} z_v \, |E(v)| \; + \sum_{\substack{u \in V \\ S(u) \subseteq E(u)}} \sum_{e \in S(u)} y_{S(u)} \, |S(u)|$$

$$= \sum_{v \in V} z_v \, |E(v)| \; + \sum_{\substack{u \in V \\ S(u) \subseteq E(u)}} y_{S(u)} \, |S(u)|^2$$

$$\leq 2\, DFS(G)$$

Since $DFS(G)$ is a lower bound on $OPT(G)$, it follows that $\sum_{v \in V} w_v \ell(v) \leq 2\, OPT(G)$. □

3.4 Scheduling edges with arbitrary processing times

The scheduling phase is essentially the same as in [6]. The only difference is in the order in which the edges are scheduled – we decide the order using the labels on the edges, whereas in [6], the ordering is based

on the completion times of the edges in the optimal linear programming solution. For completeness, we restate the scheduling algorithm here as described in [6]. In the scheduling phase, each edge $e = (u, v)$ waits for W_e time units before it can be scheduled, where

$$W_e = \beta \max\{p(F_e(u)), p(F_e(v))\}.$$

In the above expression, $F_e(u) = \{f \in E(u) \,|\, \ell(f) \leq \ell(e)\}$ and $F_e(v) = \{f \in E(v) \,|\, \ell(f) \leq \ell(e)\}$. When processing $(u, v) \in E$, the edge is scheduled at the earliest time such that no edge incident upon u or v is already scheduled at that time. 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 labels on edges as priorities. The precise rules are given in the pseudo-code below. 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 pseudo code for the algorithm, as it appears in Figure 4, 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.

SCHEDULE$(G = (V, E), \ell, W)$
1   sort edges $(u, v) \in E$ in lexicographic order of their labels.
2      $t \leftarrow 0$
3      $Finished \leftarrow Active(t) \leftarrow \emptyset$
4   **for** each $e \in E$ **do**
5           $wait(e, t) \leftarrow 0$
6   **while** $(Finished \neq E)$ **do**
7           $t \leftarrow t + 1$
8           $Active(t) \leftarrow \{e \mid e \in Active(t - 1) \text{ and } e \notin Active(t - p_e)\}$
9   **for** each edge $e \in Active(t - 1) \setminus Active(t)$ **do**
10              $\widetilde{C}_e \leftarrow t - 1$
11              $Finished \leftarrow Finished \cup \{e\}$ // e is *finished*
12   **for** each edge $e = (u, v) \in E \setminus (Active(t) \cup Finished)$
13   processed in non-decreasing order of their labels **do**
14   **if** $(Active(t) \cap (E(u) \cup E(v)) = \emptyset)$ and $(wait(e, t - 1) = W_e)$ **then**
15                  $Active(t) \leftarrow Active(t) \cup \{e\}$ // e is *active*
16   **for** each edge $e = (u, v) \in E \setminus (Active(t) \cup Finished)$ **do**
17   **if** $(Active(t) \cap (E(u) \cup E(v)) = \emptyset)$ **then**
18                  $wait(e, t) \leftarrow wait(e, t - 1) + 1$ // e is *waiting*
19   **else** $wait(e, t) \leftarrow wait(e, t - 1)$ // e is *delayed*
20   **return** $\widetilde{C}$

Fig. 4. Scheduling with arbitrary processing times.

3.5 Analysis when edges have arbitrary processing times

Consider a vertex x and an edge $e = (x, y)$. Let $B_e(x) = \{f \in E(x) \,|\, \ell(f) > \ell(e), \widetilde{C}_f < \widetilde{C}_e\}$, i.e., edges in $E(x)$ that have a greater label than that of e, but finish before e in our algorithm. Recall that $F_e(x) = \{f \in E(x) \,|\, \ell(f) \leq \ell(e)\}$. Note that $e \in F_e(x)$. Let $\overline{F}_e(x) = F_e(x) \setminus \{e\}$.

We analyze the completion time of an arbitrary but fixed vertex $v \in V$. Without loss of generality, let $e_v = (v, w)$ be the edge that finishes last among the edges in $E(v)$. We analyze our algorithm for the case

8

where all edges in $\overline{F}_{e_v}(v) \cup \overline{F}_{e_v}(w)$ finish before e_v in our algorithm. If this is not true then our results can only improve. Let $\widetilde{C}_v$ be the completion time of vertex v in our algorithm.

Lemma 5. *For each $v \in V$, $\widetilde{C}_v \leq \beta \max\{p(F_{e_v}(v)), \ell(v)\} + p(E(v)) + p(F_{e_v}(w)) + p(B_{e_v}(w))$.*

Proof. Observe that when e_v is in delayed mode it must be that some edge in $\overline{F}_{e_v}(v) \cup B_{e_v}(v) \cup \overline{F}_{e_v}(w) \cup B_{e_v}(w)$ must be active. Hence, we have

$$\widetilde{C}_v = \widetilde{C}_{e_v}$$
$$\leq W_{e_v} + p(F_{e_v}(v)) + p(B_{e_v}(v)) + p(F_{e_v}(w)) + p(B_{e_v}(w))$$
$$= \beta \max\{p(F_{e_v}(v)), p(F_{e_v}(w))\} + p(E(v)) + p(F_{e_v}(w)) + p(B_{e_v}(w))$$
$$\leq \beta \max\{p(F_{e_v}(v)), \ell(v)\} + p(E(v)) + p(F_{e_v}(w)) + p(B_{e_v}(w))$$

The last expression follows using Lemma 1. $\square$

Lemma 6. *For any vertex $v \in V$, $p(F_{e_v}(w)) + p(B_{e_v}(w)) \leq \max\{p(F_{e_v}(v)), \ell(v)\} + \frac{1}{\beta} p(E(v))$.*

Proof. Let $f = (w, z) \in B_{e_v}(w)$ be an edge with the largest label. When edge f is waiting, it must be that e_v is waiting or some edge in $E(v)$ is being processed. Thus we have

$$\beta \left(p(F_{e_v}(w)) + p(B_{e_v}(w)) \right) \leq W_f$$
$$\leq W_{e_v} + p(E(v))$$
$$= \beta \max\{p(F_{e_v}(v)), p(F_{e_v}(w))\} + p(E(v))$$
$$\leq \beta \max\{p(F_{e_v}(v)), \ell(v))\} + p(E(v))$$

The last inequality follows from Lemma 1. $\square$

Lemma 7. *For any vertex $v \in V$, $\widetilde{C}_v \leq (1 + \beta) \max\{p(E(v)), \ell(v)\} + (1 + \frac{1}{\beta}) p(E(v))$*

Proof. The claim follows from Lemmas 5 and 6, and the fact that $p(F_{e_v}(v)) \leq p(E(v))$. $\square$

Theorem 2. *The data migration problem with edges having arbitrary processing times has a 5.83-approximate primal-dual algorithm.*

Proof. Let $G = (V, E)$ be an instance of the data migration problem. The cost of the schedule found by our algorithm is given by

$$\sum_{v \in V} w_v \widetilde{C}_v \leq \sum_{v \in V} w_v \left((1 + \beta) \max\{p(E(v)), \ell(v)\} + \left(1 + \frac{1}{\beta} \right) p(E(v)) \right) \qquad [\text{ using Lemma 7 }]$$

$$= (1 + \beta) \sum_{v \in V} w_v \max\{p(E(v)), \ell(v)\} + \left(1 + \frac{1}{\beta} \right) \sum_{v \in V} w_v \, p(E(v))$$

Clearly, $\sum_{v \in V} w_v \, p(E(v)) \leq OPT(G)$. Now, suppose

$$\sum_{v \in V} w_v \max\{p(E(v)), \ell(v)\} \leq 2 \, OPT(G). \tag{6}$$

It follows that

$$\sum_{v \in V} w_v \widetilde{C}_v \leq \left(3 + 2\beta + \frac{1}{\beta} \right) OPT(G).$$

9

The right hand side in the above expression is minimized for $\beta = \frac{1}{\sqrt{2}}$, which gives a ratio of $3 + 2\sqrt{2} \approx 5.83$. It all boils down to showing (6). This is done by relating the left hand side of (6) to the cost of the dual feasible solution obtained by the algorithm, which we denote by $DFS(G)$.

$$\sum_{v \in V} w_v \, \max\{p(E(v)), \ell(v)\}$$

$$= \sum_{v \in V} \left(z_v + \sum_{\substack{e=(u,v) \\ S(u):e \in S(u)}} y_{S(u)} p_e \right) \max\{p(E(v)), \ell(v)\} \qquad [\text{ all } v \in V \text{ are tight }]$$

$$\leq \sum_{v \in V} z_v \, p(E(v)) \; + \sum_{v \in V} \sum_{\substack{e=(u,v) \\ S(u):e \in S(u)}} y_{S(u)} \, p_e \max\{p(E(v)), \ell(v)\} \qquad [\text{ using Lemma 2 }]$$

$$\leq \sum_{v \in V} z_v \, p(E(v)) \; + \sum_{v \in V} \sum_{\substack{e=(u,v) \\ S(u):e \in S(u)}} y_{S(u)} \, p_e \, p(S(u)) \qquad [\text{ using Lemma 3 }]$$

$$= \sum_{v \in V} z_v \, p(E(v)) \; + \sum_{\substack{u \in V \\ S(u) \subseteq E(u)}} \sum_{e \in S(u)} y_{S(u)} \, p_e \, p(S(u))$$

$$= \sum_{v \in V} z_v \, p(E(v)) \; + \sum_{\substack{u \in V \\ S(u) \subseteq E(u)}} y_{S(u)} \, p(S(u))^2$$

$$\leq 2 \, DFS(G)$$

Since $DFS(G)$ is a lower bound on $OPT(G)$, it follows that $\sum_{v \in V} w_v \max\{p(E(v)), \ell(v)\} \leq 2 \, OPT(G)$. □

4 Minimizing Sum of Edge Completion Times

The problem of scheduling the edges (with unit processing times) of a graph to minimize the sum of their completion times can be cast as an edge coloring problem: Given $G = (V, E)$ we want to partition the edge set E into matchings $M_1, \ldots, M_k$ as to minimize $\sum_i i \, |M_i|$. Indeed, this problem is also known as minimum sum edge coloring.

Bar-Noy *et al.* [2] show that *any* minimal schedule is 2-approximate. In a minimal schedule every matching M_i is maximal with respect to $G \setminus \cup_{j<i} M_j$. The main result of this section is to identify a stronger minimality requirement that results in a better approximation guarantee.

Definition 1. *A schedule $M_1, \ldots, M_k$ of G is said to be* strongly minimal *if, for all $1 \leq b \leq k$, the b-matching $\cup_{i \leq b} M_i$ is maximal with respect to G.*

Theorem 3. *Any strongly minimal schedule is $\sqrt{2}$-approximate.*

Proof. The high level idea of the proof is to *assign* every edge to at least one of its endpoints. Each vertex is responsible for paying for the cost of the edges assigned to it. In order to pay for this cost each vertex charges a lower bound on the completion time of the edges assigned to it.

Let $(u, v) \in M_i$, we say endpoint u is *full* if u is matched in all M_j, $j < i$. We consider the endpoints of edges in M_1 to be full. Notice that every edge $(u, v) \in M_i$ must have at least one full endpoint, otherwise $\cup_{j<i} M_j + (u, v)$ would be a valid $(i-1)$-matching, which contradicts the fact that the schedule is strongly minimal. If both endpoints of (u, v) are full then the edge is *half-assigned* to u and v. Otherwise the edge is *fully-assigned* to the one full endpoint.

Every vertex u is responsible for the cost of edges assigned to it. If an edge is half-assigned to u, then u pays for half of its completion time; if the edge is fully-assigned then u pays in full. Let s_1 and s_2 be

10

the number of half-assigned and fully-assigned edges to u respectively. Notice that all edges assigned to u must belong to M_j for some $j \leq s_1 + s_2$. Think of u as paying $\frac{1}{2}$ of the completion time of *all* edges assigned to it, plus an additional $\frac{1}{2}$ for the fully-assigned edges, which in the worst case will be scheduled the latest,

$$u \text{ must pay} \leq \frac{1}{2} \sum_{i=1}^{s_1+s_2} i + \frac{1}{2} \sum_{i=s_1+1}^{s_1+s_2} i.$$

Vertex u will pay this amount by charging the completion time (in the optimal solution) of the edges assigned to it. Fully-assigned edges are charged a soon-to-be-determined ρ factor, and half-assigned edges are charged $\frac{\rho}{2}$. This constitutes u's budget. How fast can the optimal solution possibly schedule these edges?

$$u\text{'s budget} \geq \frac{\rho}{2} \sum_{i=1}^{s_1+s_2} i + \frac{\rho}{2} \sum_{i=1}^{s_2} i.$$

Notice that every edge is charged at most to an extent of ρ: fully-assigned edges are charged ρ once, from a single endpoint, and half-assigned edges are charged $\frac{\rho}{2}$ twice, once from each endpoint. Thus, strongly minimal schedules are ρ-approximate. The discrepancy between the upper and lower bound on the completion times of edges assigned to u is due to fully-assigned edges which are scheduled the latest in the upper bound, and the earliest in the lower bound. We need to determine the smallest ρ such that u's budget is enough to cover u's payment, namely

$$\frac{(s_1 + s_2)(s_1 + s_2 + 1)}{4} + \frac{(2s_1 + s_2 + 1)s_2}{4} \leq \rho \frac{(s_1 + s_2)(s_1 + s_2 + 1)}{4} + \rho \frac{s_2(s_2 + 1)}{4}.$$

Or equivalently,

$$(s_1 + s_2)^2 + (2s_1 + s_2)s_2 \leq \rho(s_1 + s_2)^2 + \rho s_2^2 + (\rho - 1)(s_1 + 2s_2).$$

Let $\alpha = \frac{s_2}{s_1 + s_2}$. Since $\rho > 1$, the above follows, provided

$$\frac{1 + 2\alpha - \alpha^2}{1 + \alpha^2} \leq \rho.$$

The left hand side is maximized for $\alpha = \sqrt{2} - 1$, which yields $\sqrt{2} \leq \rho$ $\qquad \square$

While strongly minimal schedules are not guaranteed to exist for general graphs, we now show that in bipartite graphs they always exist and can be computed in polynomial time. The bipartite is an interesting and nontrivial case: It is a variant of the open shop scheduling problem in which we want to minimize the sum of completion time of operations [6]. The problem is APX-hard [16] and the best known approximation guarantee for it is 1.796 [6].

Theorem 4. *The procedure* FIND STRONGLY MINIMAL *is a $\sqrt{2}$-approximation for minimizing the sum of completion times of open shop with unit processing times. scheduling.*

Proof. In each iteration, the procedure FIND STRONGLY MINIMAL computes a matching incident to the maximum degree vertices of G and removes the matching from G. This continues until all edges have been removed. The matchings found are then scheduled in reverse order. Because the degree of G decreases by one with each iteration, the algorithm finishes after Δ iterations, here Δ is the degree of the original graph.

Let us argue that the schedule found is strongly minimal. Let $e \in M_i$ and $b < i$, we want to show that e cannot be added to $\cup_{j \leq b} M_j$ without violating the b-matching property. Let G' be the remaining graph when M_i was computed. One of the endpoint of e must have degree i in G', let u be that endpoint. After

11

```
FIND STRONGLY MINIMAL(G)

1      for i ← Δ down to 1 do
2           M_i ← a matching incident to all vertices of G with degree i
3           G ← G \ M_i
4      return M_1, M_2, . . . , M_Δ
```

Fig. 5. Computing a strongly minimal schedule.

removing M_i the degree of u becomes $i-1$, and thus u must be matched in M_{i-1}. In general u will be matched in all $M_{j<i}$. Therefore, the degree of u in $\cup_{j \leq b} M_j$ is b, which in turn means the b-matching is maximal with respect to e.

In bipartite graphs a matching incident to all the maximum degree vertices always exists and can be computed in polynomial time (cf. [5]). Together with Theorem 3, this finishes the proof. $\qquad \square$

4.1 An almost tight example

While at first sight the analysis of the approximation factor of strongly minimal schedules may seem too pessimistic, it turns out it is almost tight. Consider the following bipartite graph with vertices $u_1, \ldots, u_n$ on one side and vertices $v_1, \ldots, v_n$ on the other side of the bipartition. There is an edge $(u_i, v_j) \in E$ if and only if $i \leq j$.

It is not difficult to show that the optimal schedule uses matchings

$$M_k = \{(u_i, v_{i+k-1}) \,|\, i \leq n - k + 1\}$$

and has cost $\sum_{i=1}^n i\,(n-i+1) = \frac{1}{6}n^3 + 3n^2 + 2n$.

Now suppose we run FIND STRONGLY MINIMAL. Initially the maximum degree vertices are u_1 and v_n, and the algorithm finds the matching M_n consisting of $(u_1, v_{\frac{n}{2}})$ and $(u_{\frac{n}{2}+1}, v_n)$. After removing M_n the maximum degree vertices are u_1, u_2, v_{n-1}, and v_n. In general the algorithm may find, for $\frac{n}{2} < k \leq n$,

$$M_k = \{(u_i, v_{i+k-\frac{n}{2}-1}) \,|\, i \leq n - k + 1\} \cup \{(u_{j-k+\frac{n}{2}+1}, v_j) \,|\, j \geq k\}.$$

After these matchings are removed from the graph we are left with a complete bipartite graph on $u_1, \ldots u_{\frac{n}{2}}$ and $v_{\frac{n}{2}+1}, \ldots v_n$, thus $|M_k| = \frac{n}{2}$ for all $1 \leq k \leq \frac{n}{2}$. Therefore, the cost of this strongly minimal schedule is $\frac{11}{48}n^3 + \frac{5}{8}n^2 + \frac{1}{3}n$.

The ratio of the cost of the optimal and strongly minimal solutions approaches 1.375 as $n \to \infty$. Compare this to the approximation guarantee of $\sqrt{2} \approx 1.414$ obtained in Theorem 3.

4.2 LP formulation gap

Let us now study the inherent limitations of the lower bounding technique used to prove Theorem 3. The lower bound used there can be generalized as follows: For any subset of edges $S(u)$ incident on a vertex u, we know that *any* feasible schedule must spend at least $\frac{|S(u)|(|S(u)|+1)}{2}$ time on these edges. We can charge the cost incurred by this set of edges, a factor $y_{S(u)} \geq 0$. If for every edge e the total charge $\left(\sum_{S(u)\,:\,e\in S(u)} y_{S(u)}\right)$ on e is at most 1, then $\sum_{S(u)} \frac{|S(u)|(|S(u)|+1)}{2} y_{S(u)}$ offers a lower bound on the cost an optimal schedule. The best such lower bound corresponds to the optimal solution of the following dual linear program.
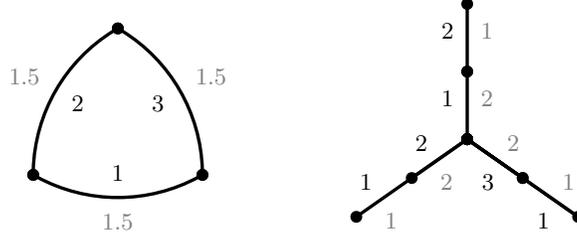
Fig. 6. LP formulation gap examples for general and bipartite instances for $\min \sum_e C_e$. Integral finishing times appear in black; fractional finishing times appear in gray.

$$\max \sum_{\substack{u \in V \\ S(u) \subseteq E(U)}} \frac{|S(u)|(|S(u)|+1)}{2} y_{S(u)}$$

subject to

$$\sum_{S(u):e \in S(u)} y_{S(u)} \leq 1 \qquad \forall e \in E \qquad (7)$$

$$y_{S(u)} \geq 0 \qquad \forall u \in V, S(u) \subseteq E(u)$$

In hindsight, the proof of Theorem 3 can be viewed as a case of dual-fitting in which constraint (7) is violated a $\sqrt{2}$ factor. To determine how good a lower bound the dual offers, we derive the primal LP and study its *LP fomulation gap*, that is, the ratio of the cost of the best optimal schedule to the cost of the best fractional schedule. (Here we do not talk about integrality gap because the LP formulations are not exact even if integrality is inforced.)

Theorem 5. *The LP formulation gap of the LP below is at least $\frac{4}{3}$ in general graphs and at least $\frac{10}{9}$ in bipartite graphs.*

$$\min \sum_{e \in E} C_e$$

subject to

$$\sum_{e \in S(u)} C_e \geq \frac{|S(u)|(|S(u)|+1)}{2} \qquad \forall u \in V, S(u) \subseteq E(u) \qquad (8)$$

$$C_e \geq 0 \qquad \forall e \in E$$

Proof. For general graphs, consider a triangle. The optimal solution schedules one edge at the time, and incurs a cost of 6. The LP can schedule all edges at $C_e = 1.5$, with a cost of 4.5. Thus, the LP formulation gap for this graph is $\frac{4}{3}$.

For the bipartite case (our example is in fact a tree) consider a spider with three legs of length two. The graph is shown in Figure 6 along with the edge completion times of an optimal schedule (in black and to the left) and of the optimal LP solution (in gray and to the right). Optimum schedules three edges in M_1, two in M_2 and one in M_3, with a total cost of 10. On the other hand, the LP solution manages to schedule all edges in two rounds, with a total cost of 9. Thus the LP formulation gap for bipartite graphs is at least $\frac{10}{9}$. $\square$

4.3 Limitations of strongly minimal schedules

We conclude this section with a note on the limitations of strongly minimal schedules. One common generalization of our scheduling problem is to minimize the weighted sum of completion times. In this setting the proof of Theorem 3 does not go through as we make crucial use of the fact that the edges have uniform weight.

It would be natural to hope that the following slight modification of FIND STRONGLY MINIMAL would produce good schedules: Instead of finding any matching incident to the maximum degree vertices, find one with minimum weight. Unfortunately, the following bipartite example shows that strongly minimal schedules are just not suited for the weighted case. Take a path of length four and replace each edge with a copy of $K_{t,t}$. The edges in the first and the last $K_{t,t}$ have weight 1, and the ones in the middle have weight 0. The optimal solution schedules the first and the last $K_{t,t}$ in the first t rounds and the remaining edges are scheduled in the next $2t$ rounds, with a total cost of $t^2(t+1)$. On the other hand, any strongly minimal solution can schedule at most t edges with weight 1 per round, thus incurring a total cost of $t^2(2t+1)$. The ratio of the cost of the two solutions approaches 2 as $t \to \infty$.

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*. Proc. of the Workshop on Algorithm Engineering, pages 145-158, 2001.
2. 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.
3. 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, 646-657, 1996.
4. 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.
5. H. Gabow and O. Kariv. *Algorithms for edge coloring bipartite graphs and multigraphs*. SIAM Journal of Computing, 11(1), February 1982.
6. R. Gandhi, M. M. Halldórsson, G. Kortsarz, and H. Shachnai. *Improved Results for Data Migration and Openshop Scheduling*. ACM Transactions on Algorithms, 2(1):116-129, 2006.
7. R. Gandhi, M. M. Halldórsson, G. Kortsarz, and H. Shachnai. *Improved Bounds for Scheduling Conflicting Jobs with Minsum Criteria*. Proc. of the Second Workshop on Approximation and Online Algorithms, 68-82, 2004.
8. R. Graham. *Bounds for certain multiprocessing anomalies*. Bell System Technical Journal, 45:1563-1581, 1966.
9. 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, 620-629, 2001.
10. 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.
11. 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.
12. 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, 353-366, 1998.
13. S. Khuller, Y. Kim, and Y. C. Wan. *Algorithms for Data Migration with Cloning.* SIAM Journal on Computing, 33(2):448461, 2004.
14. S. Khuller and A. Malekian and Y. Kim.. *Improved Algorithms for Data Migration.* In Proc. of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, 164–175, 2006.
15. Y. Kim. *Data Migration to Minimize the Average Completion Time.* Journal of Algorithms,55:42-57, 2005.
16. D. Marx. *Complexity results for minimum sum edge coloring.* Manuscript, 2004.
17. J. Mestre. *Adaptive Local Ratio.* Proc. of the 19th ACM-SIAM Symposium on Discrete Algorithms, 2007.

18. T. Nishizeki and K. Kashiwagi. *On the 1.1 edge-coloring of multigraphs.* SIAM Journal on Discrete Mathematics, 3(3):391-410, 1990.
19. M. Queyranne. *Structure of a Simple Scheduling Polyhedron.* Mathematical Programming, 58:263-285, 1993.
20. 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.
21. M. Queyranne and M. Sviridenko. *Approximation Algorithms for Shop Scheduling Problems with Minsum Objective.* Journal of Scheduling, 5:287-305, 2002.
22. A. S. Schulz. *Scheduling to Minimize Total Weighted Completion Time: Performance Guarantees of LP-based Heuristics and Lower Bounds.* In Proc. of the 5th International Conference on Integer Programming and Combinatorial Optimization, LNCS 1084, 301-315, 1996.
23. 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.