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] Department of Computer Science, University of Maryland, College Park, MD 20742.
Research 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 the edges represent the data transfers required between pairs of devices. Each vertex has a non-negative weight, and each edge has unit 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. We give a more efficient primal-dual algorithm that achieves the same approximation guarantee, which can be extended to yield a 5.83-approximation for arbitrary processing times. 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 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 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. In this work, we assume that the edges have unit length, that is, all data transfers take the same amount of 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 equal (unit) lengths, the problem reduces to edge coloring of the transfer (multi)graph of the system. The best approximation algorithm known for minimum edge coloring [17] then yields an algorithm for data migration with unit edge length, whose makespan is $1.1\chi' + 0.8$, 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 edge lengths are the same and showed that Graham's list scheduling algorithm [7], when guided by an optimal solution to a linear programming relaxation, gives an approximation ratio of 3. When edges have arbitrary lengths there are several constant factor approximation algorithms [5, 15, 20] with the best approximation guarantee being 5.03 [5].

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 [5]. Open shop scheduling problem has been studied in [3, 12, 19, 20].

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 lengths, several constant factor approximation algorithms [11, 15, 6] are known with the best approximation factor being 7.682 [6]. For the case of unit edges lengths, 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.* [5] give a 1.796-approximate solution that uses an algorithm for sum coloring of interval graphs due to Halldórsson *et al.* [11].

Our Contribution: First we study the data migration problem with unit length edges and the objective of minimizing the average completion time over all storage disks. Kim [15] gave a 3-approximation algorithm that rounds 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. [6]). Gandhi *et al.* [5] show that Kim's algorithm can not give an approximation guarantee better than 3. In this work, we present an efficient primal-dual algorithm that gives a 3-approximate solution; our scheme can be extended to yield a 5.83-approximation for 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 $\sum_i i|M_i|$. Bar-Noy *et al.* [2] show that if M_i is maximal with respect to $G \setminus \cup_{j<i} M_j$ then we get a 2-approximate solution. We show that if, for all $b \geq 1$, the b-matching $\cup_{j \leq b} M_j$ is maximal in G then we get a $\sqrt{2}$-approximate schedule. We show that such schedules always exist in 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.* [5] show that using the sum-coloring algorithm of Halldórsson *et al.* [11] one can obtain a 1.796 approximation guarantee. 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 $G = (V, E)$. 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. We assume that edges have unit length. The completion time of an edge is simply the time at which its processing is completed. 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 $\sum_{v \in V} w_v C_v$.

We first analyze the performance of the following natural and intuitive algorithm: 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],

we show in Appendix A that for the objective of $\min \sum_v C_v$ it may produce a solution with cost $\Omega(\sqrt[3]{n})$ times 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 [22] and Queyranne [18] for single machine scheduling problems and by Schulz [21] 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 neighbors of vertex u.

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

subject to

$$\sum_{v \in S_u} C_v \geq \frac{|S_u| \, (|S_u| + 1)}{2} \qquad \forall u \in V, S_u \subseteq N(u) \qquad (1)$$

$$C_v \geq 0 \qquad \forall v \in V$$

The dual LP contains a variable y_{S_u} (for each set S_u) corresponding to each constraint represented by (1). The dual LP is given below.

$$\max \sum_{\substack{u \in V \\ S_u \subseteq N(u)}} \frac{|S_u|(|S_u| + 1)}{2} \, y_{S_u}$$

subject to

$$\sum_{\substack{u \in V \\ S_u : v \in S_u}} y_{S_u} \leq w_v \qquad \forall v \in V \qquad (2)$$

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

2.2 Algorithm

The high level idea of the algorithm is as follows. There are two phases—*labeling* and *scheduling*.

In the labeling phase, each vertex is initially unlabeled. This phase proceeds in iterations; iteration i labels some neighbors of x_i. Vertex x_i is chosen to be the one with maximum number of unlabeled neighbors. Let S_{x_i} be the unlabeled neighbors of x_i. The value of the dual variable $y_{S_{x_i}}$ is incremented until the dual constraint (2) is met with equality for some vertex $v \in S_{x_i}$. In other words, $y_{S_{x_i}}$

assumes the smallest value such that for some vertex $v \in S_{x_i}$ we have

$$\sum_{j<i\,:\,v\in S_{x_j}} y_{S_{x_j}} + y_{S_{x_i}} = w_v.$$

Let $T_{x_i} \subseteq S_{x_i}$ be the vertices for which the above equality holds. All vertices T_{x_i} are labeled $|S_{x_i}|$. The label of vertex u is denoted by $\ell(u)$.

In the scheduling phase, the edges in E are ordered so that edge (u, v) precedes (u', v') if:

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

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

The edges in E are then processed in 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 below.

```
PRIMAL-DUAL(G = (V, E))
1   // labeling phase
2   for each v ∈ V do
3       ℓ(v) ← nil                    // v is unlabeled
4   i ← 0
5   while (there exists an unlabeled vertex) do
6           i ← i + 1
7           x_i ← vertex with the maximum number of unlabeled neighbors.
8           S_{x_i} ← unlabeled neighbors of x_i.
9           y_{S_{x_i}} ← min_{v∈S_{x_i}} {w_v}
10          T_{x_i} ← {v ∈ S_{x_i} | w_v = y_{S_{x_i}}}
11          for each v ∈ T_{x_i} do
12              ℓ(v) ← |S_{x_i}|      // v is now labeled
13          for each v ∈ S_{x_i} do
14              w_v ← w_v − y_{S_{x_i}}
15  // scheduling phase
16  sort edges (u, v) ∈ E in lex. order of ⟨min{ℓ(u), ℓ(v)}, max{ℓ(u), ℓ(v)}⟩
17  for each edge e = (u, v) ∈ E processed in order do
18      schedule e if no edge in E(u) ∪ E(v) is already scheduled.
```

2.3 Analysis

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

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

Proof. Let (w, v) be the last edge to finish among the edges in $E(v)$. Also, let $F(w, v) = \{y \in N(w) \,|\, \ell(y) \leq \ell(v)\}$. Observe that because of the order in which

the edges are scheduled, $\widetilde{C}_v \leq |F(w,v)| + |E(v)| - 1$. Let i be the iteration of the algorithm in which the first vertex in $F(w,v)$ is labeled, and let y be that vertex. At the beginning of the ith iteration vertex w has at least $|F(w,v)|$ unlabeled neighbors. Because x_i is chosen (line 7 of the pseudocode) to be the vertex with the maximum number of unlabeled neighbors it must be the case that $|F(w,v)| \leq |S_{x_i}| = \ell(y)$, which by definition is at most $\ell(v)$. □

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. Let $DFS(G)$ denote the cost of the dual feasible solution for instance G obtained by our algorithm. Let $OPT(G)$ denote the cost of an optimal solution for instance G. Clearly, $DFS(G) \leq OPT(G)$. Also, $OPT(G) \geq \sum_{v \in V} w_v |E(v)|$. Let $iter(v)$ be the iteration in which v gets labeled. The cost of our algorithm is given by

$$\begin{aligned}
\sum_v w_v \widetilde{C}_v &\leq \sum_v w_v \left(\ell(v) + |E(v)|\right) \qquad \text{(using Lemma 1)} \\
&= \sum_v w_v \, \ell(v) + \sum_v w_v |E(v)| \\
&= \sum_v \left(\sum_{i \,:\, v \in S_{x_i}} y_{S_{x_i}} \right) \ell(v) + OPT(G) \\
&= \sum_v \left(\sum_{i \,:\, v \in S_{x_i}} y_{S_{x_i}} \right) |S_{x_{iter(v)}}| + OPT(G) \\
&\leq \sum_v \sum_{i \,:\, v \in S_{x_i}} y_{S_{x_i}} |S_{x_i}| + OPT(G) \\
&= \sum_i \sum_{v \in S_{x_i}} y_{S_{x_i}} |S_{x_i}| + OPT(G) \\
&= \sum_i y_{S_{x_i}} |S_{x_i}|^2 + OPT(G) \\
&= \sum_{\substack{u \in V \\ S_u \subseteq N(u)}} y_{S_u} |S_u|^2 + OPT(G) \\
&\leq 2 \cdot DFS(G) + OPT(G) \\
&\leq 3 \cdot OPT(G)
\end{aligned}$$

□

We finish this section by mentioning that the above labeling procedure coupled with the scheduling technique used by Gandhi *et al.* [5] yields a constant factor approximation for the case of arbitrary processing times.

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

Due to space limitation the proof of the above theorem is deferred to the journal version of this paper.

3 Minimizing Sum of Edge Completion Times

The problem of scheduling the edges 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 w.r.t. 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 $M_{j<i}$. 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 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 $j \leq s_1 + s_2$. We 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 will be charged a soon-to-be-determined ρ factor, and half-assigned edges will be charged $\frac{\rho}{2}$. This will be

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 greedy 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 $\sqrt{2} \leq \rho$ □

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 [5]. This problem is APX-hard [16]. The best known approximation guarantee for the problem is 1.796 [5].

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

```
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
```

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 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. [8]). Together with Theorem 3, this finishes the proof of Theorem 4.

3.1 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 approximation guarantee of $\sqrt{2} \approx 1.414$ obtained in Theorem 3.

3.2 Integrality 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 incident on a vertex, we know that *any* feasible schedule must spend at least $\frac{|S|(|S|+1)}{2}$ time on these edges. We can charge the cost of this set of edges a factor $y_S \geq 0$. If for every edge e the total charge $\left(\sum_{S\,:\,e\in S} y_S\right)$ on e is at most 1, then $\sum_S \frac{|S|(|S|+1)}{2}\, y_S$ offers a lower bound on the cost an optimal schedule. The best such lower bound corresponds to the optimal solution of the following dual LP:

$$\max \sum_S \frac{|S|(|S|+1)}{2}\, y_S$$

subject to

$$\sum_{S:e\in S} y_S \leq 1 \qquad \forall e \in E \tag{3}$$

$$y_S \geq 0 \qquad \forall S \subseteq E(u), u \in V$$

In hindsight, the proof of Theorem 3 can be viewed as a case of dual-fitting in which constraint (3) is violated a $\sqrt{2}$ factor. To determine how good a lower bound the dual offers, we derive the primal LP and study its integrality gap.

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

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

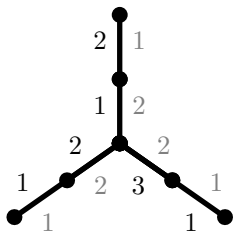
subject to

$$\sum_{e\in S} C_e \geq \frac{|S|(|S|+1)}{2} \qquad \forall S \subseteq E(u), u \in V \tag{4}$$

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

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 integrality 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 on the right 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 integrality gap for bipartite graphs is at least $\frac{10}{9}$. □



3.3 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, we 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 $t(t+1)$. On the other hand, a strongly minimal solution can schedule at most t edges with weight 1 per round, thus incurring a total cost of $t(2t+1)$. The ratio of the cost of the two solutions approaches 2 as $t \to \infty$.

Acknowledgements: We thank Yoo-Ah Kim 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*. 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. 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.
6. 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.
7. R. Graham. *Bounds for certain multiprocessing anomalies*. Bell System Technical Journal, 45:1563-1581, 1966.
8. H. Gabow and O. Kariv. *Algorithms for edge coloring bipartite graphs and multigraphs*. SIAM Journal of Computing, 11(1), February 1982.
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.* In Proc. of the 22nd ACM Symposium on Principles of Database Systems, 27-36, 2003.
14. S. Khuller and A. Malekian. *Improved Algorithms for Data Migration.* To appear in APPROX 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. T. Nishizeki and K. Kashiwagi. *On the 1.1 edge-coloring of multigraphs.* SIAM Journal on Discrete Mathematics, 3(3):391-410, 1990.
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.* In Proc. of the 5th International Conference on Integer Programming and Combinatorial Optimization, LNCS 1084, 301-315, 1996.
22. 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.

A Minimal Schedules and min $\sum_v C_v$

Since *any* minimal schedule is 2-approximate with respect to the objectives $\min \sum_e C_e$ [2] and $\min \max_e C_e$, it may tempting to think that they also perform well for $\min \sum_v C_v$. Unfortunately that is not the case. Take a complete bipartite graph $K_{q,q}$ and attach to it $2q$ copies of $K_{1,\sqrt{q}}$ so that each node in $K_{q,q}$ is the center of one of the stars. The optimal solution first schedules the stars in parallel and then the edges in $K_{q,q}$, with a total cost of $\Theta(q^2)$. A minimal solution may schedule $K_{q,q}$ before the stars, incurring a cost of $\Theta(q^{2.5})$. Since the graph has $\Theta(q^{1.5})$ vertices, a minimal schedule can be a factor $\Omega(\sqrt[3]{n})$ away from the optimum.