SIAM J. COMPUT. © 2010 Society for Industrial and Applied Mathematics
Vol. 39, No. 7, pp. 3038–3057

ADAPTIVE LOCAL RATIO*

JULIÁN MESTRE†

Abstract. Local ratio is a well-known paradigm for designing approximation algorithms for combinatorial optimization problems. At a very high level, a local-ratio algorithm first decomposes the input weight function w into a positive linear combination of simpler weight functions or *models*. Guided by this process, a solution S is constructed such that S is α-approximate with respect to each model used in the decomposition. As a result, S is α-approximate under w as well. These models usually have a very simple structure that remains "unchanged" throughout the execution of the algorithm. In this work we show that adaptively choosing a model from a richer spectrum of functions can lead to a better local ratio. Indeed, by turning the search for a good model into an optimization problem of its own, we get improved approximations for a data migration problem.

Key words. local-ratio technique, primal-dual schema, approximation algorithms, scheduling problems

AMS subject classifications. 90C05, 90C27, 97P20, 90B35, 68W25

DOI. 10.1137/080731712

1. Introduction.

The local-ratio technique and the primal-dual schema are two well-known paradigms for designing approximation algorithms for combinatorial optimization problems. Over the years a clear connection between the two paradigms was observed as researchers found primal-dual interpretations for local-ratio algorithms [12, 3] and vice versa [8, 5]. This culminated with the work of Bar-Yehuda and Rawitz [10] showing their equivalence under a fairly general and encompassing definition of primal-dual and local-ratio algorithms. For a survey of results and a historical account of the local-ratio technique, see [7]; for a survey of the primal-dual schema, see [20, 30].

At a very high level, a local-ratio algorithm consists of two steps. First, the input weight function w is decomposed into a positive linear combination of *models* $\hat{w}_i$, that is, $w = \epsilon_1 \hat{w}_1 + \cdots + \epsilon_n \hat{w}_k$ and $\epsilon_i \geq 0$. Then, guided by this process, a solution S is constructed such that $\hat{w}_i(S) \leq \alpha\, \hat{w}_i(A)$ for any feasible solution A for all i. We refer to α as the *local ratio* of $\hat{w}_i$. By the local-ratio theorem [8], it follows that S is α-approximate with respect to w.

Typically the models used in local-ratio approximation algorithms are 0-1 functions or simple aggregates of structural features of the problem at hand. (In primal-dual parlance this corresponds to increasing some dual variables uniformly when constructing the dual solution.) Furthermore, the structure of the models remains "unchanged" throughout the execution of the algorithm. For example, consider the vertex cover problem. Bar-Yehuda and Even [8] set the weight of the endpoints of a yet-uncovered[1] edge to 1 and the remaining vertices to 0; Clarkson [13] chose a number of yet-uncovered edges forming a star and set the weight of each vertex to the number of star edges incident on it; while Gandhi, Khuller, and Srinivasan [18] and

*Received by the editors July 31, 2008; accepted for publication (in revised form) February 19, 2010; published electronically June 18, 2010. A preliminary version appeared in *Proceedings of the Nineteenth ACM-SIAM Symposium on Discrete Algorithms* (SODA), 2008, pp. 152–160. This research was supported by an Ann G. Wylie Dissertation Fellowship, NSF grant CCF-0430650, and an Alexander von Humboldt Fellowship.

http://www.siam.org/journals/sicomp/39-7/73171.html

†Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany (jmestre@mpi-inf.mpg.de).

[1]By yet-uncovered edge we mean that both of its endpoints have nonzero residual weight.

Bar-Yehuda [6] set the weight of each vertex to the number of yet-uncovered edges incident on it.[2]

This paper studies a problem for which adaptively choosing a model from a richer spectrum of weight functions leads to a better local ratio, and thus to better approximations. Indeed, by turning the search for a good model into an optimization problem of its own, we get improved approximations for a *data migration* problem. We hope that these findings encourage the study of nonuniform updates for local-ratio or primal-dual algorithms; perhaps in some cases, as in our problem, this may help realize the full potential of these techniques.

1.1. Our results. The *data migration* problem arises in large storage systems, such as *storage area networks* [24], where a dedicated network of disks is used to store multimedia data. As the data access pattern evolves over time, the load across the disks must 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 current data layout to the target data layout. While migration is being performed, the storage system is running suboptimally, so it is important to perform the migration quickly. The problem can be modeled [25] with a *transfer multigraph* (V, E), where each vertex corresponds to a disk and each edge $(u, v) \in E$ corresponds to a data item that must be transferred between u and v. A disk can be involved in at most one transfer at a time, and each data transfer takes one unit of time. We are to schedule the transfers so as to minimize the sum of completion times of the disks. The problem is NP-hard, but 3-approximations are known [25, 19].

First, we cast the primal-dual algorithm of Gandhi and Mestre [19] as a local-ratio algorithm and provide a family of instances showing their analysis is tight. To overcome these difficult instances we propose to adaptively choose a model minimizing the local ratio and formulate the problem of finding such a model as a linear program (LP). Interestingly, our algorithm is neither purely combinatorial nor LP rounding, but lies somewhere in between. Every time the weight function needs to be updated, an LP is solved to find the best model. These LPs are much smaller that the usual LP formulations, so our scheme should be faster than an LP rounding algorithm.

In the analysis we show that the models found using the LP exhibit a better local ratio than the usual 0-1 models. Somewhat surprisingly a precise characterization of the local ratio can be derived analytically. We prove that the overall scheme is a $(1 + \phi)$-approximation, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio, and give a family of instances achieving this ratio.

To derive the worst-case local ratio of our scheme we use a method similar to the factor-revealing LP approach of Jain et al. [23], which has been successfully applied in the analysis of many greedy heuristics [23, 2, 22, 11]. The basic idea of the factor-revealing LP method is to use an LP to find a worst-case instance maximizing the approximation ratio achieved by the heuristic at hand. The value of this LP is then upper bounded by producing a dual solution. We also use a mathematical program to find the worst-case instance maximizing the local ratio of our scheme. The main difference is that, since we already use an LP to guide our local-ratio algorithm, the resulting factor-revealing program is nonlinear. Even though we cannot solve this program numerically, we are still able to prove a tight bound of $1 + \phi$ on its cost.

[2]In fact, Bar-Yehuda's algorithm [6] works for a generalization of vertex cover and uses slightly different weights. However, when the algorithm is applied to regular vertex cover, the weights are as described.

Finally, we present an alternative algorithm based on the fractional local ratio technique [9], which requires solving a single, albeit larger, LP.

1.2. Related work on data migration. As noted by Coffman et al. [14], if every transfer takes one unit of time, a schedule is simply a proper edge coloring of the transfer multigraph.

Many objective functions have been studied within this framework, such as minimizing the maximum disk completion time (makespan) [14, 28], the sum of disk completion times [25, 16, 19], and the sum of transfer completion times [4, 25, 27, 31]. A common variant is to allow each transfer to have an arbitrary processing time [29, 25, 14, 16, 17]. Generalizations of the makespan minimization problem in which there are storage constraints on disks or constraints on how the data can be transferred have also been studied [21, 1, 24, 26].

This paper is concerned mainly with the objective of minimizing the weighted sum of vertex completion times. Kim [25] proved that the problem is NP-hard and showed that a simple greedy algorithm guided by an optimal LP solution is a 3-approximation. Gandhi et al. [16] showed that Kim's analysis is tight. Finally, Gandhi and Mestre [19] showed that the same greedy algorithm guided by a dual update is also a 3-approximation, but that an arbitrary greedy schedule can be a $\omega(1)$-factor away from optimum.

2. Algorithmic framework. The input to our problem consists of a transfer graph $G = (V, E)$ and a weight function $w : V \to R^+$. For ease of exposition, we assume for now that G is simple; later, in section 6, we will show how to remove this assumption. A feasible schedule $S : E \to Z^+$ is a proper edge-coloring of G; that is, if two edges $e_1 \neq e_2$ are incident on the same vertex, then $S(e_1) \neq S(e_2)$. We are to find a scheduling minimizing $w(S) = \sum_{u \in V} w(u) \max_{v \in N(u)}\{S(u, v)\}$. Throughout the paper we use $\mathrm{N}(u)$ to denote the set of neighbors for u, and d_u to denote the degree of u.

Let us cast the primal-dual algorithm of Gandhi and Mestre [19] as a local-ratio algorithm; in the process, we generalize it slightly. The pseudocode of *Adaptive Local Ratio* (ALR) is given in Algorithm 1. The algorithm has two stages: labeling and scheduling. The labeling stage assigns a label ℓ_u to every $u \in V$. These labels are then used to guide the scheduling stage.

Initially every node is unlabeled; i.e., $\ell_v = \textbf{nil}$ for all $v \in V$. Denote the set of unlabeled neighbors of u with $\mathrm{UN}(u) = \{v \in \mathrm{N}(u) \mid \ell_v = \textbf{nil}\}$. In each iteration, choose a node u with the maximum number of unlabeled neighbors $\Delta = |\mathrm{UN}(u)|$. Then choose a *model* $\hat{w} : V \to R^+$ with support in $\mathrm{UN}(u)$, find the largest $\epsilon \geq 0$ such that $\epsilon\hat{w} \leq w$, and set $w \leftarrow w - \epsilon\hat{w}$. As a result, at least one vertex in $\mathrm{UN}(u)$ has zero weight in the updated w; set the label of these vertices to Δ. This continues until all vertices are labeled and $w = 0$. How the model $\hat{w}$ is chosen will be specified shortly. Ultimately, as the name of the algorithm suggests, we will adaptively choose a model so as to minimize the algorithm's local ratio and, therefore, its approximation ratio.

Once the labels are computed, the edges $(u, v) \in E$ are sorted in increasing value of $\min\{\ell_u, \ell_v\}$, breaking ties with $\max\{\ell_u, \ell_v\}$. The edges are scheduled greedily in sorted order: Start with the empty schedule and add the edges, one by one in sorted order, to the current schedule as early as possible without creating a conflict.

The labels guide the scheduling phase and allow us to bound the finishing time of the vertices. To motivate the use of the labels, Appendix A shows that the same scheduling procedure can yield a solution with cost $\omega(1)$ times the optimum when guided by the degrees instead of the labels.

Algorithm 1. ALR (V, E, w).

```
1.  // LABELING STAGE
2.  for $u \in V$ do
3.      $\ell_u \leftarrow$ nil
4.  repeat
5.      choose $u \in V$ maximizing $\Delta = |\mathrm{UN}(u)|$
6.      choose $\hat{w}$ with support in $\mathrm{UN}(u)$
7.      $w \leftarrow w - \min\left\{\frac{w(u)}{\hat{w}(u)} \;\middle|\; \hat{w}(u) > 0\right\}\hat{w}$
8.      for $v \in \mathrm{UN}(u) \;\big|\; w(v) = 0$ do
9.          $\ell_v \leftarrow \Delta$
10. until every vertex is labeled
11. // SCHEDULING STAGE
12. sort $(u, v) \in E$ in lexicographic order of $\big\langle \min\{\ell_u, \ell_v\}, \max\{\ell_u, \ell_v\}\big\rangle$
13. $S \leftarrow$ empty schedule
14. for $e \in E$ in sorted order do
15.     add $e$ to $S$ as early as possible
16. return S
```

LEMMA 2.1 (see [19]). *In the schedule returned by* ALR *every vertex $v \in V$ finishes no later than $\ell_v + d_v - 1$.*

Proof. Let (v, y) be the edge incident on v that is scheduled the latest in S. Note that (v, y) need not be the last edge (in sorted order) among those edges incident on v and y. For example, the edge (v, y') may come after (v, y) in sorted order and yet be scheduled before (v, y); this can occur if at some early time slot both v and y' are free (allowing us to schedule (v, y')), but y is busy (preventing us from scheduling (v, y)).

Note, however, that (v, y) cannot be scheduled later than one plus the number of edges incident on v and y that come before (v, y) in sorted order. How many edges incident on u can there be before (v, y) in sorted order? Clearly, there are at most $d_v - 1$ such edges. How many edges incident on y are there before (v, y) in sorted order? We claim that there are at most $\ell_v - 1$ such edges. It follows that (v, y) must be scheduled not later than $\ell_v + d_v - 1$.

Let $x \in \mathrm{N}(y)$ be such that $\ell_v < \ell_x$. If $\ell_v < \ell_y$, then (v, y) must come before (x, y) since

$$\min\{\ell_v, \ell_y\} = \ell_v < \ell_y = \min\{\ell_x, \ell_y\}.$$

Similarly, if $\ell_v \geq \ell_y$, then (v, y) must come before (x, y) since

$$\min\{\ell_v, \ell_y\} = \min\{\ell_x, \ell_y\} = \ell_y \quad \text{and} \quad \max\{\ell_v, \ell_y\} = \ell_v < \ell_x = \max\{\ell_x, \ell_y\}.$$

Consider the set $X = \{x \in \mathrm{N}(y) \mid \ell_x \leq \ell_v\}$. It follows that the number of edges incident on y that come before (y, v) in sorted order is at most $|X| - 1$. Note that the value of the labels assigned in line 9 of ALR can only decrease with time. Consider the first iteration of the algorithm in which the node u chosen in line 5 of ALR was such that $|\mathrm{UN}(u)| = \ell_v$; at this point in time $\mathrm{UN}(y) = X$. Since u is chosen to maximize $|\mathrm{UN}(u)|$, it follows that $|X| = |\mathrm{UN}(y)| \leq |\mathrm{UN}(u)| = \ell_v$. Thus, as was claimed, there are at most $\ell_v - 1$ edges incident on y that come before (v, y) in sorted order. ∎

As it stands, the algorithm is underspecified: We have not described how the model $\hat{w}$ is chosen in line 7. It is important to realize, though, that Lemma 2.1

holds regardless of our choice of $\hat{w}$. Gandhi and Mestre [19] proposed using $\hat{w}(v) = \mathbb{I}[v \in \mathrm{UN}(u)]$ as a model, where $\mathbb{I}[\cdot]$ is a 0-1 indicator function, and showed that this is a 3-approximation. To gain some intuition, let us show that the local ratio of $\hat{w}(v) = \mathbb{I}[v \in \mathrm{UN}(u)]$ is at most 3.

LEMMA 2.2. *If line 7 of* ALR *always uses* $\hat{w}(v) = \mathbb{I}[v \in \mathrm{UN}(u)]$, *then* $\hat{w}(S) \leq \left(3 - \frac{4}{\Delta+1}\right) \hat{w}(A)$ *for any schedule* A, *where* $\Delta = |\mathrm{UN}(u)|$.

Proof. An obvious lower bound on the cost of A is $\hat{w}(A) \geq \sum_{v\in \mathrm{UN}(u)} d_v$. Furthermore, since nodes in $\mathrm{UN}(u)$ share u as a common neighbor, it follows that $\hat{w}(A) \geq \sum_{i\in[\Delta]} i = \Delta(\Delta+1)/2$. On the other hand, by Lemma 2.1 we get

$$\begin{aligned}\hat{w}(S) &\leq \sum_{v\in \mathrm{UN}(u)} (\ell_v + d_v - 1),\\ &\leq \Delta(\Delta-1) + \sum_{v\in \mathrm{UN}(u)} d_v,\\ &= \frac{\Delta(\Delta+1)}{2}\left(2 - \frac{4}{\Delta+1}\right) + \sum_{v\in \mathrm{UN}(u)} d_v,\\ &\leq \left(3 - \frac{4}{\Delta+1}\right)\hat{w}(A).\end{aligned}$$

The second inequality follows from the fact that the labels that ALR assigns can only decrease with time. The third inequality follows from the two lower bounds on $\hat{w}(A)$ outlined above. ∎

Since S is a 3-approximation with respect to every model and the input weight function w is a positive linear combination of these models, it follows that S is 3-approximate with respect to w as well. It is worth pointing out that the bound on the local ratio obtained in Lemma 2.2 is tight if we assume that the upper bound on completion times given by Lemma 2.1 is also tight. (To see this, consider what happens when the ith vertex in $\mathrm{UN}(u)$ has degree $d_i = i$.) Of course, this alone does not imply a tight bound on the overall approximation guarantee. However, as we will see in Lemma 3.7, there is a family of instances where the algorithm in [19] produces a schedule whose cost is $3 - o(1)$ times optimum.

Note that the degree sequence $d_i = i$ can be easily circumvented if we use a different model: Instead of $\hat{w}(v) = \mathbb{I}[v \in \mathrm{UN}(u)]$, which may have a local ratio $3 - \frac{4}{\Delta+1}$, use the model that gives a 1 to the nodes in $\mathrm{UN}(u)$ with maximum degree. For the latter model, this particular degree sequence has a local ratio of at most $2 - \frac{1}{\Delta}$. Indeed, in general choosing the better of these two strategies leads to a better local ratio.

LEMMA 2.3. *If line 7 of* ALR *chooses* $\hat{w}(v) = \mathbb{I}\left[v \in \mathrm{argmax}_{x\in \mathrm{UN}(u)} d_x\right]$ *when* $\max_{x\in \mathrm{UN}(u)} d_x > \lfloor \beta\Delta \rfloor$, *and* $\hat{w}(v) = \mathbb{I}[v \in \mathrm{UN}(u)]$ *when* $\max_{x\in \mathrm{UN}(u)} d_x \leq \lfloor \beta\Delta \rfloor$, *then* $\frac{\hat{w}(S)}{\hat{w}(A)} \leq 2.802$ *for* $\beta = 0.555$.

Proof. Let x be a node in $\mathrm{UN}(u)$ with maximum degree. Furthermore, suppose $d_x > \lfloor \beta\Delta \rfloor$. Then the local ratio of $\hat{w}$ can be bounded as follows:

$$\frac{\hat{w}(S)}{\hat{w}(A)} \leq \frac{\Delta + d_x - 1}{d_x} = \frac{\Delta - 1}{d_x} + 1 \leq \frac{\Delta - 1}{\lfloor \beta\Delta \rfloor + 1} + 1 < \frac{1}{\beta} + 1. \tag{2.1}$$

Now let us see what happens when $d_x \leq \lfloor \beta\Delta \rfloor$. Recall that in this case $\hat{w}(x) = 1$ for all $x \in \mathrm{UN}(u)$. Consider the order in which the edges between u and $\mathrm{UN}(u)$ are scheduled in some feasible solution A, and let $d_1, d_2, \dots, d_\Delta$ be the degree sequence of nodes in $\mathrm{UN}(u)$ listed in this order. Clearly the ith vertex in $\mathrm{UN}(u)$ cannot finish earlier than $\max\{i, d_i\}$ in A, so we get the lower bound

$$\hat{w}(A) \geq \textstyle\sum_i \max\{i, d_i\}. \tag{2.2}$$

On the other hand, by Lemma 2.1 we get the upper bound

$$\hat{w}(S) \leq \textstyle\sum_i \Delta + d_i - 1. \tag{2.3}$$

Consider a degree sequence maximizing the ratio of (2.3) and (2.2). We can assume without loss of generality that A schedules the edges so that the degrees d_i are in nondecreasing order, since this minimizes (2.2). Furthermore, for every $i \leq \lfloor \beta\Delta \rfloor$ we must have $d_i = i$, for otherwise we could increase or decrease d_i to get a larger ratio. Also, for every $i > \lfloor \beta\Delta \rfloor$ the ratio is largest when $d_i = \lfloor \beta\Delta \rfloor$—recall that we have the constraint that the maximum degree should not exceed $\lfloor \beta\Delta \rfloor$. It follows that we can restrict our attention to the degree sequence $d_i = \min\{i, \lfloor \beta\Delta \rfloor\}$, whose local ratio is

$$\frac{\hat{w}(S)}{\hat{w}(A)} \leq \frac{\Delta^2 + \frac{\Delta^2 - (\Delta - \lfloor \beta\Delta \rfloor)^2}{2}}{\frac{\Delta^2}{2}} = \frac{3\Delta^2 - (\Delta - \lfloor \beta\Delta \rfloor)^2}{\Delta^2} < 3 - (1-\beta)^2. \tag{2.4}$$

Combining (2.1) and (2.4), we get that the local ratio is

$$\max\left\{\frac{1}{\beta} + 1,\, 3 - (1-\beta)^2\right\}.$$

It is now straightforward to check that this quantity is minimized when $\beta \approx 0.555$ and that the local ratio attained for this value of β is 2.802. □

Besides a modest improvement in the approximation guarantee, Lemma 2.3 suggests a general line of attack: In each iteration find a model that minimizes the local ratio. The rest of the paper is devoted to pursuing this strategy.

3. Minimizing the local ratio.

The abstract problem we are to solve is, given a sequence $d = (d_1, d_2, \dots, d_\Delta)$ corresponding to the degrees of vertices in $\mathrm{UN}(u)$, find weights $\hat{w} = (\hat{w}_1, \hat{w}_2, \dots, \hat{w}_\Delta)$ minimizing the local ratio of $\hat{w}$.

In order to evaluate the quality of a given model $\hat{w}$ we first need an upper bound on $\hat{w}(S)$, where S is the schedule produced by ALR. For this we use Lemma 2.1 and the fact that the values of labels assigned in line 9 of ALR can only decrease with time.

DEFINITION 3.1. $\mathrm{UB}(d, \hat{w}) = \sum_{i \in [\Delta]} \hat{w}_i \,(d_i + \Delta - 1)$.

Similarly, we need a lower bound on $\hat{w}(A)$, where A can be any schedule. Note that A must schedule the edges from $\mathrm{UN}(u)$ to u at different time slots; this induces a total order on $\mathrm{UN}(u)$, which we denote by the permutation $\sigma : [\Delta] \to [\Delta]$. Note that vertex i cannot finish earlier than $\sigma(i)$ since (u, i) is the $\sigma(i)$th edge incident on u to be scheduled, or earlier than d_i since all edges incident on i must be scheduled before it finishes.

DEFINITION 3.2. $\mathrm{LB}(d, \hat{w}) = \min_{\sigma:[\Delta]\to[\Delta]} \sum_{i \in [\Delta]} \hat{w}_i \max\{d_i, \sigma(i)\}$.

It follows from the above discussion that $\hat{w}(S) \leq \mathrm{UB}(d, \hat{w})$ and $\mathrm{LB}(d, \hat{w}) \leq \hat{w}(A)$ for all A. Hence, the minimum local ratio for d can be expressed as a function of UB and LB.

DEFINITION 3.3. *Let* $\rho(d) = \inf_{\hat{w}} \frac{\mathrm{UB}(d,\hat{w})}{\mathrm{LB}(d,\hat{w})}$ *be the minimum local ratio of* d.

We now turn our attention to the problem of computing a model $\hat{w}$ with a local ratio that achieves $\rho(d)$. This can be done using the program $\min\{\mathrm{UB}(d,\hat{w}) \mid \mathrm{LB}(d,\hat{w}) \geq 1,\, \hat{w} \geq 0\}$, which can be written as an LP:

$$\text{(LP1)} \qquad \min \sum_{i\in[\Delta]} (d_i + \Delta - 1)\,\hat{w}_i$$

subject to

$$\text{(3.1)} \qquad \sum_{i\in[\Delta]} \hat{w}_i \, \max\{d_i, \sigma(i)\} \geq 1 \qquad \forall \sigma : [\Delta] \to [\Delta],$$

$$\hat{w}_i \geq 0 \qquad \forall\, i \in [\Delta].$$

Clearly, (LP1) computes a model $\hat{w}$ with local ratio $\rho(d)$. Even though (LP1) is exponentially large, it can be solved in polynomial time using the ellipsoid method—the separation oracle involves solving a minimum assignment problem where the cost of an edge (i,j) is $\hat{w}_i \max\{d_i, j\}$. The ellipsoid algorithm, however, is not practical, so below we derive a more succinct formulation. The new formulation is obtained through a number of step-by-step transformations. First, we rewrite the constraints (3.1) of (LP1) as a single (nonlinear) constraint:

$$\text{(3.2)} \qquad \min\left\{ \sum_{i,j\in[\Delta]} x_{i,j}\cdot \hat{w}_i \max\{d_i, j\} \;\middle|\; \begin{array}{rll} \sum_j x_{i,j} & \geq 1 & \forall i \in [\Delta], \\ \sum_i x_{i,j} & \leq 1 & \forall j \in [\Delta], \\ x_{i,j} & \in \{0,1\} & \forall i,j \in [\Delta] \end{array} \right\} \geq 1.$$

Clearly, constraint (3.2) is equivalent to all constraints (3.1) in (LP1) put together. Then we relax the integrality requirement on the $x_{i,j}$ variables inside the assignment problem:

$$\text{(3.3)} \qquad \min\left\{ \sum_{i,j\in[\Delta]} x_{i,j}\cdot \hat{w}_i \max\{d_i, j\} \;\middle|\; \begin{array}{rll} \sum_j x_{i,j} & \geq 1 & \forall i \in [\Delta], \\ \sum_i x_{i,j} & \leq 1 & \forall j \in [\Delta], \\ x_{i,j} & \geq 0 & \forall i,j \in [\Delta] \end{array} \right\} \geq 1.$$

Since the polytope of the assignment problem is integral [15], constraints (3.3) and (3.2) are equivalent. Now we replace the LP corresponding to the minimum cost assignment problem with its dual program

$$\text{(3.4)} \qquad \max\left\{ \sum_{i\in[\Delta]} (y_i - z_i) \;\middle|\; \begin{array}{rll} y_i - z_j & \leq \max\{d_i, j\}\,\hat{w}_i & \forall i,j \in [\Delta], \\ y_i, z_i & \geq 0 & \forall i \in [\Delta] \end{array} \right\} \geq 1.$$

By the strong duality theorem, constraints (3.3) and (3.4) are equivalent. Finally, we unpack the left-hand side of (3.4) into (LP1) to obtain the following equivalent linear programming formulation:

$$\text{(LP2)} \qquad \min \sum_{i\in[\Delta]} (d_i + \Delta - 1)\,\hat{w}_i$$

subject to

$$\text{(3.5)} \qquad \sum_{i\in[\Delta]} (y_i - z_i) \geq 1,$$

$$\text{(3.6)} \qquad y_i - z_j \leq \max(d_i, j)\,\hat{w}_i \qquad \forall\, i,j \in [\Delta],$$

$$y_i, z_i, \hat{w}_i \geq 0 \qquad \forall\, i \in [\Delta].$$

This finishes the description of ALR from Algorithm 1. Namely, in each iteration of the labeling stage, line 7 solves (LP2) to find the best model for the degree sequence of nodes in $\mathrm{UN}(u)$.

DEFINITION 3.4. *Let $\rho = \sup_d \rho(d)$ and $\rho_\Delta = \max_{d:|d|=\Delta} \rho(d)$.*

THEOREM 3.5. ALR *is a ρ-approximation for the data migration problem, and this is tight.*

The proof that the algorithm is a ρ-approximation follows straightforwardly from the local-ratio theorem [7]. Let S be the schedule produced by ALR. For every model $\hat{w}$ used by ALR, by definition, S is ρ-approximate with respect to $\hat{w}$. Since the input weight function w is a linear combination of these models, S is ρ-approximate with respect to w as well. The tightness claim follows from the next two lemmas.

LEMMA 3.6. *For any Δ, we have $\rho_\Delta < \rho_{2\Delta}$.*

Proof. Let d be such that $\rho_\Delta = \rho(d)$. Then define $d'_{2i-1} = d'_{2i} = 2d_i$ for each $i \in [\Delta]$. Let $(\hat{w}', y', z')$ be a solution to (LP2) for d' with cost $\rho(d')$. Define $y_i = y'_{2i-1} + y'_{2i}$, $z_i = z'_{2i-1} + z'_{2i}$, and $\hat{w}_i = 2(\hat{w}'_{2i-1} + \hat{w}'_{2i})$. The solution $(\hat{w}, y, z)$ is feasible for d since

$$\sum_{i\in[2\Delta]} (y'_i - z'_i) \geq 1 \implies \sum_{i\in[\Delta]} (y_i - z_i) \geq 1$$

and

$$\begin{array}{r}y'_{2i-1} - z'_{2j-1} \leq \max\{d'_{2i-1}, 2j-1\}\, \hat{w}'_{2i-1} \\ y'_{2i} - z'_{2j} \leq \max\{d'_{2i}, 2j\}\, \hat{w}'_{2i}\end{array} \implies y_i - z'_j \leq \max\{d_i, j\}\, \hat{w}_i.$$

Furthermore, the cost of $(\hat{w}, y, z)$ is less than the cost of $(\hat{w}', y', z')$:

$$\begin{aligned}\sum_{i\in[\Delta]} (d_i + \Delta - 1)\, \hat{w}_i &= \sum_{i\in\Delta} (d_i + \Delta - 1)\ 2\,(\hat{w}'_{2i-1} + \hat{w}'_{2i}) \\ &= \sum_{i\in[2\Delta]} (2d_{\lceil i/2 \rceil} + 2\Delta - 2)\, \hat{w}'_i \\ &< \sum_{i\in[2\Delta]} (d'_i + 2\Delta - 1)\hat{w}'_i.\end{aligned}$$

The lemma follows since

$$\rho_\Delta = \rho(d) \leq \mathrm{cost}(\hat{w}, y, z) < \mathrm{cost}(\hat{w}', y', z') = \rho(d') \leq \rho_{2\Delta}. \qquad \square$$

The next lemma shows that for any degree sequence d obeying certain properties, if ALR chooses a certain model $\hat{w}$ in line 7 when the vertices in $\mathrm{UN}(u)$ have degrees $d_1, \ldots, d_\Delta$, then there exists an instance where the algorithm constructs a solution whose cost is $\frac{\mathrm{UB}(d,\hat{w})}{\mathrm{LB}(d,\hat{w})}\left(1 - \frac{1}{\Delta}\right)$ times the optimum. Even though the lemma does not apply to arbitrary degree sequences, it is nevertheless general enough that we can show the worst-case performance of the algorithm to be tight.

LEMMA 3.7. *Let $d = (d_1, \ldots, d_\Delta)$ be a degree sequence such that $1 < d_i < \Delta$ for all $i \in [\Delta]$. Suppose that line 7 of* ALR *chooses model $\hat{w}$ when the vertices in* $\mathrm{UN}(u)$ *have degrees d. Then the algorithm can produce a schedule with cost $\frac{\mathrm{UB}(d,w)}{\mathrm{LB}(d,w)}\left(1 - \frac{1}{\Delta}\right)$ times the optimum.*

Proof. Consider the instance in Figure 3.1, namely, a tree with four levels. The ith node in the second level has weight $\hat{w}_i$; nodes in other levels have weight zero.

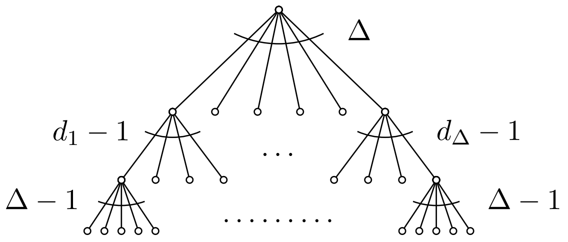


FIG. 3.1. *Tight instance for* ALR.

The root has degree Δ, the ith node in the second level has degree d_i, and nodes in the third level have degree Δ.

Consider an execution of the algorithm that chooses the root in the first iteration. As a result, all nodes in the second level get a label of Δ. In the next $\sum_i (d_i - 1)$ iterations the leaves are labeled $\Delta - 1$. Finally, the root gets a label of $\max_i d_i$, and the nodes on the third level get a label less than or equal to $\max_i d_i$. Now consider a node in the third level; note that the children are labeled $\Delta - 1$, while its parent is labeled Δ. Therefore, the edges between the third and fourth levels will be scheduled before those edges between the second and third levels. Since $d_i > 1$, the ith node in the second level has at least one child, and thus it must finish at time $d_i + \Delta - 2$ or later. On the other hand, the optimal solution has cost precisely $\mathrm{LB}(d, \hat{w})$. Therefore, the approximation ratio is at least

$$\frac{\sum_{i \in [\Delta]} \hat{w}_i \, (d_i + \Delta - 2)}{\mathrm{LB}(d, \hat{w})} = \frac{\mathrm{UB}(d, \hat{w}) - \sum_{i \in [\Delta]} \hat{w}_i}{\mathrm{LB}(d, \hat{w})} \geq \frac{\mathrm{UB}(d, w)}{\mathrm{LB}(d, w)} \left(1 - \frac{1}{\Delta}\right),$$

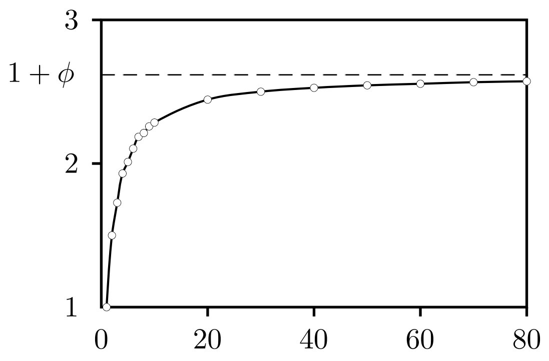
where the last inequality follows from the fact that $\mathrm{UB}(d, \hat{w}) \geq \sum_{i \in [\Delta]} \Delta \, \hat{w}_i$. ∎

As a corollary, we get that the analysis of Gandhi and Mestre [19] is essentially tight. Recall that their algorithm always uses as a model $\hat{w}_i = 1$ for all $i \in [\Delta]$ on all degree sequences. Then for the degree sequence

$$d_i = \begin{cases} 2 & \text{if } i = 1, \\ i & \text{if } 1 < i < \Delta, \\ \Delta - 1 & \text{if } i = \Delta, \end{cases}$$

we have $\mathrm{LB}(d, \hat{w}) = \frac{\Delta(\Delta+1)}{2} + 1$ and $\mathrm{UB}(d, \hat{w}) = \Delta^2 + \frac{\Delta(\Delta-1)}{2}$, which implies a local ratio of $3 - \frac{4\Delta+6}{\Delta(\Delta+1)+2}$. By Lemma 3.7 it follows that there are instances where the algorithm of Gandhi and Mestre [19] returns a $(3 - o(1))$-approximate solution.

On the other hand, if in each iteration of ALR we use (LP2) to find a model with minimum local ratio, then the approximation factor becomes ρ. To argue that the algorithm can produce solutions with arbitrarily close to ρ times the optimum, let $d_1, \ldots, d_\Delta$ be a degree sequence with $\rho(d)$ close to ρ. Recall that if $\max_i d_i \geq \Delta$, then $\rho(d) < 2$, which in this case can be achieved using the model $\hat{w}(i) = \mathbb{I}\left[d_i \in \operatorname{argmax}_{j \in [\Delta]} d_j\right]$. As we will see shortly, ρ is strictly larger than 2; thus, we can safely assume that $\max_i d_i < \Delta$. Let d' be a new degree sequence of length 2Δ defined as $d'_{2i-1} = d'_{2i} = 2d_i$. This is the sequence used in the proof of Lemma 3.6,



Δ	ρ_Δ
1	1
2	1.5
3	1.7273
4	1.9310
5	2.0115
6	2.1042
7	2.1863
8	2.2129
9	2.2589
10	2.2857

Δ	ρ_Δ
20	2.4453
30	2.5006
40	2.5275
50	2.5447
60	2.5556
70	2.5667
80	2.5728
$\vdots$	$\vdots$
∞	$1+\phi$

FIG. 3.2. *Experimental evaluation of* $\rho_\Delta = \max_{d:|d|=\Delta} \rho(d)$.

where it was shown that $\rho(d) < \rho(d')$. The degree sequence d' has the properties needed to apply Lemma 3.7, so we know that there are instances where the algorithm produces a solution with cost $\rho(d')\left(1 - \frac{1}{2\Delta}\right)$ times the optimum. By Lemma 3.6 we can assume that Δ is large enough to make this as close to $\rho(d')$, and therefore to ρ, as desired.

It remains only to bound ρ. Somewhat surprisingly, a precise characterization in terms of the golden ratio $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$ can be derived analytically.

THEOREM 3.8. *$\rho = 1 + \phi$.*

The next section is devoted to proving this theorem. Figure 3.2 shows ρ_Δ for small values of Δ obtained through exhaustive search.

4. A tight bound for ρ.

We start by showing that $\rho \le 1 + \phi$. In a sense, we need to argue that every degree sequence d has a good model. Recall that in each iteration of the algorithm a model is found by solving (LP2). At first glance this may seem like an obstacle since we are essentially treating our LP as a black box. We can, however, bound ρ using linear duality.

The idea is to replace (LP2) with its dual problem (LP3) given below. By the strong duality theorem the optimal solutions of (LP2) and (LP3) have the same cost:

$$\text{(LP3)} \qquad \max\ \alpha$$

subject to

$$\begin{aligned}
\sum_{j\in[\Delta]} x_{i,j} &\ge \alpha && \forall\, i \in [\Delta],\\
\sum_{i\in[\Delta]} x_{i,j} &\le \alpha && \forall\, j \in [\Delta],\\
\sum_{j\in[\Delta]} \max\{d_i, j\}\, x_{i,j} &\le d_i + \Delta - 1 && \forall\, i \in [\Delta],\\
x_{i,j}, \alpha &\ge 0 && \forall\, i, j \in [\Delta].
\end{aligned}$$

Recall that $\rho_\Delta = \max_{d:|d|=\Delta} \rho(d)$. Suppose we modify (LP3) by letting $d_1, \ldots, d_\Delta$ be variables in $[\Delta]$. The result is a nonlinear mathematical program for ρ_Δ:

$$(\mathrm{NLP}_\Delta) \qquad \max\ \alpha$$

subject to

$$\sum_{j\in[\Delta]} x_{i,j} \ge \alpha \qquad \forall\, i \in [\Delta], \tag{4.1}$$

$$\sum_{i\in[\Delta]} x_{i,j} \le \alpha \qquad \forall\, j \in [\Delta], \tag{4.2}$$

$$\sum_{j\in[\Delta]} \max\{d_i, j\}\, x_{i,j} \le d_i + \Delta - 1 \qquad \forall\, i \in [\Delta], \tag{4.3}$$

$$x_{i,j}, \alpha \ge 0 \qquad \forall\, i, j \in [\Delta],$$

$$d_i \in [\Delta] \qquad \forall i \in [\Delta].$$

Note that (NLP_Δ) is feasible and its optimal value is always at least 1. For example, setting $x_{i,j} = \frac{2\Delta-1}{\Delta^2}$ and $d_i = \Delta$ for all $i, j \in [\Delta]$ leads to a solution with $\alpha = 2 - \frac{1}{\Delta}$.

The plan is to show that (NLP_Δ) is upper bounded by $1 + \phi$. To that end, let us first derive some structural properties of the solutions of (NLP_Δ). We can picture x as a Δ by Δ matrix of positive reals, where every row sums up to at least α (4.1) and every column sums up to at most α (4.2). Note that this implies that every row and every column of x sum up exactly to α; this fact will be useful in our proofs.

For a given row of x, consider the smallest set of consecutive entries from the left to right that adds up to at least 1, and call this set the *first unit* of the row. The first lemma essentially says that once x and α are chosen, there is a simple rule for choosing d: Simply set d_i to be the largest index in the first unit of the ith row.

LEMMA 4.1. *Let (x, d, α) be an optimal solution for (NLP_Δ). Then there exists another optimal solution (x, d', α), where $d'_i = \min\{k \mid \sum_{j=1}^{k} x_{i,j} \ge 1\}$.*

Proof. First note that the new degree sequence d' is well defined since $\alpha \ge 1$. The plan is to transform d into d' step by step. Suppose that $d_i < \min\{k \mid \sum_{j=1}^{k} x_{i,j} \ge 1\}$ for some $i \in [\Delta]$. Then we can increment d_i by 1, which will increase the left-hand side of (4.3) by less than 1 (since $\sum_{j=1}^{d_i} x_{i,j} < 1$) and its right-hand side by 1. Therefore, the modification preserves feasibility, and we can keep doing this until $d_i = d'_i$. Similarly, if $d_i > \min\{k \mid \sum_{j=1}^{k} x_{i,j} \ge 1\}$, we can safely decrease d_i without violating feasibility, because the left-hand side of (4.3) decreases by at least 1 (since $\sum_{j=1}^{d_i-1} x_{i,j} \ge 1$), and the right-hand side decreases by 1. ∎

To prove our upper bound on the cost of (NLP_Δ), we need an optimal solution where the contribution of the first unit of each row is concentrated on one or two adjacent entries. The next lemma provides this crucial property. Figure 4.1 provides an example matrix obeying the conditions stated in the lemma.

LEMMA 4.2. *There is an optimal solution (x, d, α) for (NLP_Δ) such that for all i*

(i) $d_i = \min\{k \mid \sum_{j=1}^{k} x_{i,j} \ge 1\}$,

(ii) $x_{i,j} = 0$ *for all* $j \le d_i - 2$, *and*

(iii) *if* $x_{i,d_i-1} \ne 0$, *then* $x_{k,d_i} = 0$ *for all* $k < i$.

Proof. Note that (i) follows from Lemma 4.1. The plan is to modify x row by row until (ii) and (iii) are satisfied for all rows. After each modification we can invoke Lemma 4.1, so we can assume that (i) always holds throughout the proof.

First sort the rows of x so that $d_1 \le d_2 \le \cdots \le d_\Delta$. For the base case consider the last row of x; that is, we prove (ii) and (iii) for $i = \Delta$. Suppose there exists $k < d_\Delta$ such that $x_{\Delta,k} > 0$. Recall that every row and every column of x sum up

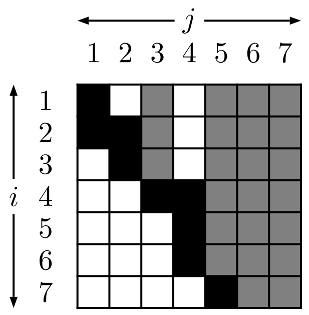


FIG. 4.1. *An example matrix complying with Lemma 4.2. Black entries correspond to nonzero entries in the first unit of each row. White entries are zero, and gray entries are nonzero.*

to α. Therefore $x_{\Delta,d_\Delta} < \alpha$, and there must be an $i' < \Delta$ such that $x_{i',d_\Delta} > 0$. Let $\epsilon = \min\{x_{\Delta,k}, x_{i',d_\Delta}\}$. Decrease $x_{\Delta,k}$ and x_{i',d_Δ} by ϵ, and increase x_{Δ,d_Δ} and $x_{i',k}$ by ϵ. Note that the update does not affect constraint (4.3) for i or i' and that it preserves (4.1) and (4.2). However, it may decrease $d_{i'}$ (as a function of x), in which case we must resort the rows. The update is repeated until $x_{\Delta,k} = 0$ for all $k < d_\Delta$.

Assuming that rows in $[i+1, \Delta]$ obey (ii) and (iii), we show how to modify the ith row. The idea is very similar to the base case. Suppose there exists $k < d_i$ such that $x_{i,k} > 0$. If there exists $i' < i$ such that $x_{i',d_i} > 0$, then we use the same update as in the base case to decrease $x_{i,k}$. We can continue to do this until $x_{i',d_i} = 0$ for all $i' < i$ or $x_{i,k} = 0$ for all $k < d_i$. In the latter case we are done, as the ith row now obeys (ii) and (iii). Suppose, then, that $x_{i',d_i} = 0$ for all $i' < i$. If $x_{i,k} = 0$ for all $k \le d_i - 2$, then again we are done. Otherwise if $x_{i,k} > 0$ for some $k \le d_i - 2$, then we run the update for x_{i,d_i-1}. In order to do so, we need to argue that there always exists $i' < i$ such that $x_{i',d_i-1} > 0$. To that end, we show that $x_{i'',d_i-1} = 0$ for all $i'' > i$. If $d_{i''} > d_i + 1$, then by (ii) and the fact that $d_{i''} \ge d_i$, we get $d_{i'',d_i-1} = 0$. If $d_{i''} = d_i$, then by (iii) and the fact that $d_{i,d_i} > 0$, we get $d_{i'',d_i-1} = 0$. Thus, since $x_{i,d_i-1} < \alpha$, there must exist $i' < i$ such that $x_{i',d_i-1} > 0$, which we can use to perform the update. We repeat the update until $x_{i,k} = 0$ for all $k \le d_i - 2$, at which point properties (ii) and (iii) hold for the ith row. ▯

Now everything is in place to prove the upper bound on the value of (NLP$_\Delta$).

LEMMA 4.3. *For any Δ, the objective value of (*NLP$_\Delta$*) is upper bounded by* $(1+\phi) + \frac{2}{\Delta-1}$.

Proof. Let (x, d, α) be an optimal solution for (NLP$_\Delta$) as in Lemma 4.2. Our goal is to show that α is less than $1 + \phi + \frac{2}{\Delta-1}$. As before, we sort the rows of x so that $d_1 \le d_2 \le \cdots \le d_\Delta$.

Let k be the largest index such that $(\phi - 1)k < d_k$. Notice that we are guaranteed k's existence since the condition is always met for $k = 1$.

If $k = \Delta$, then by constraint (4.3) we have

$$\sum_{j\in[\Delta]} \max\{d_\Delta, j\}\, x_{i,j} \le d_\Delta + \Delta - 1,$$

which together with (4.1) gives us

$$d_\Delta\, \alpha \le d_\Delta + \Delta - 1. \tag{4.4}$$

Rearranging the terms in (4.4) and using the fact that $(\phi-1)\Delta < d_\Delta$, we get

$$\alpha \le \frac{\Delta-1}{d_\Delta} + 1 < \frac{\Delta}{(\phi-1)\Delta} + 1 = \phi + 1.$$

Let us consider the case where $k < \Delta$. Adding up constraints (4.3) for all i such that $k < i \le \Delta$, we get

$$\sum_{i=k+1}^{\Delta}\sum_{j=1}^{\Delta} \max\{d_i, j\}\, x_{i,j} \le \sum_{i=k+1}^{\Delta} (d_i + \Delta - 1). \tag{4.5}$$

Let $\lambda_j = \sum_{i=k+1}^{\Delta} x_{i,j}$. We can lower bound the left-hand side of (4.5) as follows:

$$\sum_{i=k+1}^{\Delta}\sum_{j=1}^{\Delta} \max\{d_i, j\}\, x_{i,j} \;\ge\; \sum_{i=k+1}^{\Delta}\sum_{j=d_k}^{\Delta} \max\{d_i, j\}\, x_{i,j} \;\ge\; \sum_{j=d_k}^{\Delta} \lambda_j\, j. \tag{4.6}$$

From property (i) of Lemma 4.2, it follows that $x_{k,d_k} > 0$. We claim that $x_{i,j} = 0$ for all $i > k$ and $j < d_k$. Indeed, if $j \le d_i - 2$, then by property (ii) of Lemma 4.2 it follows that $x_{i,j} = 0$. Note that if $d_i > d_k$, then $j < d_k < d_i$ and so $x_{i,j} = 0$. Hence, we need only consider the case $j = d_i - 1 = d_k - 1$. By property (iii) of Lemma 4.2, if $x_{i,d_i-1} \neq 0$, then we should have $x_{k,d_i} = 0$, which would contradict the fact that $x_{k,d_i} = x_{k,d_k} > 0$. Therefore, $x_{i,j} = 0$ for all $i > k$ and $j < d_k$. This implies that $\sum_{j=d_k}^{\Delta} \lambda_j = (\Delta - k)\,\alpha$. Also, recall that every row and every column of x sum up to α and therefore $\lambda_j \le \alpha$. Given these constraints on the λ_j values, the right-hand side in (4.6) is at its minimum when $\lambda_j = \alpha$ for $d_k \le j \le d_k + \Delta - k - 1$, which yields

$$\begin{aligned}
\sum_{i=k+1}^{\Delta}\sum_{j=1}^{\Delta} \max\{d_i, j\}\, x_{i,j} \;&\ge\; \sum_{j=d_k}^{d_k+\Delta-k-1} \alpha\, j, \\
&=\; \alpha\, \frac{(\Delta-k)\,(2d_k + \Delta - k - 1)}{2}, \\
&> \alpha\, \frac{(\Delta-k)}{2}\Big[(2\phi-3)k + \Delta - 1\Big],
\end{aligned} \tag{4.7}$$

where the last inequality uses the fact that $(\phi-1)k < d_k$.

Now let us upper bound the right-hand side of (4.5) using the fact that, by the definition of k, we have $d_i \le (\phi-1)\,i$ for all $i > k$:

$$\begin{aligned}
\sum_{i=k+1}^{\Delta} (d_i + \Delta - 1) &\le (\Delta-k)(\Delta-1) + \sum_{i=k+1}^{\Delta} (\phi-1)\, i \\
&= \frac{(\Delta-k)}{2}\Big[2\,(\Delta-1) + (\phi-1)(\Delta+k+1)\Big].
\end{aligned} \tag{4.8}$$

Plugging (4.7) and (4.8) back into (4.5) and rearranging the terms, we get

$$\alpha \le \frac{(\phi-1)\,(\Delta+k+1) + 2\,(\Delta-1)}{(2\phi-3)k + \Delta - 1}. \tag{4.9}$$

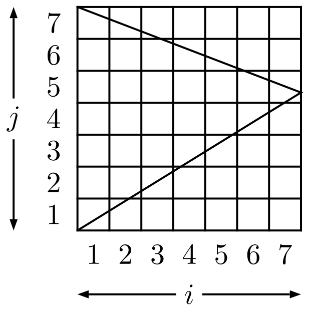


FIG. 4.2. *How to construct a solution for* (NLP$_\Delta$) *for* $\Delta = 7$.

Using the fact that $\phi - 1 = (1+\phi)(2\phi - 3)$, we can simplify (4.9) to get

$$\begin{aligned}\alpha &\le \frac{(\phi-1)k + (\phi-1)(\Delta-1) + 2(\phi-1) + 2(\Delta-1)}{(2\phi-3)k + \Delta - 1},\\ &= \frac{(\phi+1)(2\phi-3)\,k + (\phi+1)(\Delta-1) + 2(\phi-1)}{(2\phi-3)k+\Delta-1},\\ &= 1+\phi+\frac{2(\phi-1)}{(2\phi-3)k+\Delta-1},\\ &\le 1+\phi+\frac{2}{\Delta-1},\end{aligned}$$

where the last inequality follows from the setting $k = 1$. ▯

By Lemmas 3.6 and 4.3, we get that $\rho \le 1+\phi$. The next lemma finishes the proof of Theorem 3.8 by showing that ρ can be arbitrarily close to $1+\phi$.

LEMMA 4.4. *For every* Δ, *the objective value of* (NLP$_\Delta$) *is lower bounded by* $(1+\phi)\left(1-\frac{3}{\Delta}\right)$.

Proof. The plan is to construct a feasible solution (x, d, α) for (NLP$_\Delta$) with $\alpha = (1+\phi)\left(1-\frac{3}{\Delta}\right)$. Since the cost of the optimal solution can only be larger than this, the lemma follows.

Imagine drawing on the Cartesian plane a Δ by Δ grid and lines $l_1 = (\phi-1)x$ and $l_2 = \Delta - (2-\phi)x$. Figure 4.2 shows the grid for $\Delta = 7$. Define the cell (i,j) to be the square $[i-1,i]\times[j-1,j]$. Suppose the intersection of cell (i,j) and l_1 defines a segment of length L_1 and that its intersection with l_2 defines a segment of length L_2; then we set

$$x_{i,j} = c\,L_1\,\frac{\phi}{\sqrt{3-\phi}} + c\,L_2\,\frac{1}{\sqrt{6-3\phi}},$$

where c is a constant that will be chosen shortly to make the solution feasible.

Let $\alpha = c(1+\phi)$ and $d_i = \lceil(\phi-1)i\rceil$ for $i \in [\Delta]$. Our goal is show that (x, d, α) is a feasible solution for (NLP$_\Delta$). First let us consider constraint (4.1) for some $i \in [\Delta]$. Let $A_i = \{(i,j) \mid j \in [\Delta]\}$ be the cells used by this constraint. The total length of l_1 covered by cells in A_i is

$$\sqrt{1^2 + (\phi-1)^2} = \sqrt{3-\phi},$$

while the total length of l_2 covered by cells in A_i is

$$\sqrt{1^2 + (2-\phi)^2} = \sqrt{6-3\phi}.$$

Therefore, it follows that

$$\sum_{j\in[\Delta]} x_{i,j} = c\sqrt{3-\phi}\,\frac{\phi}{\sqrt{3-\phi}} + c\sqrt{6-3\phi}\,\frac{1}{\sqrt{6-3\phi}} = c\,(1+\phi).$$

Now consider constraint (4.2) for some $j \in [\Delta]$. Let $B_j = \{(i,j) \mid i \in [\Delta]\}$ be the cells used by this constraint. Consider a horizontal strip of width γ_1 intersecting l_1, but not l_2; then the total length of l_1 covered by the strip is

$$\sqrt{\left(\frac{\gamma_1}{\phi-1}\right)^2 + \gamma_1^2} = \gamma_1\frac{\sqrt{3-\phi}}{\phi-1} = \gamma_1\,(1+\phi)\,\frac{\sqrt{3-\phi}}{\phi}.$$

Similarly, consider a horizontal strip of width γ_2 intersecting l_2, but not l_1; then the total length of l_2 covered by the strip is

$$\sqrt{\left(\frac{\gamma_2}{2-\phi}\right)^2 + \gamma_2^2} = \gamma_2\frac{\sqrt{6-3\phi}}{2-\phi} = \gamma_2\,(1+\phi)\,\sqrt{6-3\phi}.$$

We can think of B_j as two horizontal strips of height γ_1 and γ_2 intersecting l_1 and l_2, respectively, where $\gamma_1 + \gamma_2 = 1$. It follows that

$$\sum_{i\in[\Delta]} x_{i,j} = c\,(1+\phi)\,\gamma_1 + c\,(1+\phi)\,\gamma_2 = c\,(1+\phi).$$

It remains only to show that constraint (4.3) is satisfied for all $i \in [\Delta]$ when $c = 1 - \frac{3}{\Delta}$. The contribution of L_1 to the x-value of cells in A_i is $c\phi$, and it is concentrated on cells $(i, \lceil(\phi-1)i\rceil)$ and $(i, \lceil(\phi-1)i\rceil - 1)$. The contribution of L_2 to the x-value of cells in A_i is c, and it is concentrated on cells $(i, \lceil\Delta - (2-\phi)(i-1)\rceil)$ and $(i, \lceil\Delta - (2-\phi)(i-1)\rceil - 1)$. Therefore,

$$\begin{aligned}
\sum_{j\in[\Delta]} \max\{d_i, j\}\, x_{i,j} &\le \lceil(\phi-1)i\rceil\, c\,\phi + \lceil\Delta - (2-\phi)(i-1)\rceil\, c,\\
&= c\left(\lceil(\phi-1)i\rceil + \Delta - 1 + (\phi-1)\lceil(\phi-1)i\rceil + \lceil 1 - (2-\phi)(i-1)\rceil\right),\\
&< c\left(\lceil(\phi-1)i\rceil + \Delta - 1\right) + (\phi-1)\lceil(\phi-1)i\rceil + \lceil 1 - (2-\phi)(i-1)\rceil,\\
&< c\left(\lceil(\phi-1)i\rceil + \Delta - 1\right) + (\phi-1)((\phi-1)i+1) + 2 - (2-\phi)(i-1),\\
&= \left(1 - \frac{3}{\Delta}\right)\left(\lceil(\phi-1)i\rceil + \Delta - 1\right) + (\phi-1)^2\, i - (2-\phi)i + 3,\\
&\le \left(\lceil(\phi-1)i\rceil + \Delta - 1\right) - \frac{3}{\Delta}\Delta + 3,\\
&= d_i + \Delta - 1,
\end{aligned}$$

where the third line follows from the fact that $c < 1$, the fourth from the relation $\lceil z\rceil < z + 1$, and the sixth because $\lceil(\phi-1)i\rceil + \Delta - 1 \ge \Delta$ for all $i \ge 1$. $\square$

5. A fractional local-ratio algorithm. In this section we present an alternative algorithm for the data migration problem that involves solving a single large LP and then round the fractional solution using ALR. The algorithm is based on the *fractional local-ratio* technique [9] and the following linear programming formulation:

$$\text{(LP4)} \qquad \min \sum_{u \in V} w_u \, C_u$$

subject to

$$\text{(5.1)} \qquad \sum_{t \in T} \max\{d_u, t\} \, r_{e,t} \le C_u \qquad \forall\, u \in V,\ e \in E(u),$$

$$\text{(5.2)} \qquad \sum_{t \in T} r_{e,t} \ge 1 \qquad \forall\, e \in E,$$

$$\text{(5.3)} \qquad \sum_{e \in E(u)} r_{e,t} \le 1 \qquad \forall\, u \in V,\ t \in T,$$

$$\text{(5.4)} \qquad r_{e,t},\, C_u \ge 0 \qquad \forall\, u \in V,\ e \in E,\ t \in T,$$

where T is a set of time slots and $E(u)$ is the set of edges incident on vertex u. Notice that in any reasonable solution the edge (u, v) will be scheduled earlier than $\deg(u) + \deg(v)$ (otherwise, it would be possible to move the edge earlier without altering the rest of the schedule). Therefore, using $T = \{1, \ldots, 2n\}$ is enough to capture any reasonable schedule. The variable $r_{e,t}$ indicates whether edge e is scheduled at time t, and C_u denotes the finishing time of vertex u. Constraint (5.2) asks that every edge is scheduled, while constraint (5.3) enforces that two edges incident on the same vertex are not scheduled in the same time slot. Finally, constraint (5.1) captures the finishing time of vertex u. It is worth noting that this formulation is stronger than the standard LP formulation for the problem—usually (5.1) is decoupled into two constraints.

The idea is to first solve (LP4) to obtain an optimal fractional solution (r^*, C^*). Then run ALR using the lower bound offered by (r^*, C^*) to guide the algorithm when choosing the model $\hat{w}$. More specifically, we will select weights $\hat{w}$ so as to minimize

$$\text{(5.5)} \qquad \frac{\sum_{v \in \mathrm{UN}(u)} (d_v + |\mathrm{UN}(u)| - 1) \, \hat{w}_v}{\sum_{v \in \mathrm{UN}(u)} \hat{w}_v \, C^*_v}.$$

This can be computed easily, as the minimum ratio must be obtained by setting a single entry of $\hat{w}$ to be nonzero. Thus, we just run ALR implementing line 7 as follows:

$$\text{(5.6)} \qquad \hat{w}(v) \leftarrow \mathbb{I}\left[v = \operatorname*{argmin}_{q \in \mathrm{UN}(u)} \frac{d_q + |\mathrm{UN}(u)| - 1}{C^*_q} \right].$$

THEOREM 5.1. *There is an LP rounding* $(1 + \phi)$*-approximation algorithm for the data migration problem.*

Proof. To show that this is indeed a $(1 + \phi)$-approximation, we need to argue that (5.5) is upper bounded by $1 + \phi$. Suppose for a moment that this is the case. Let S be the schedule produced by the algorithm, and let (C^*, r^*) be the fractional used to guide it. By our assumption, for every model $\hat{w}$ used we have

$$\hat{w}(S) \le (1 + \phi) \, \hat{w}(C^*).$$

Adding up over all models, we get that S is a $(1 + \phi)$-approximation:

$$w(S) \le (1 + \phi) \, w(C^*).$$

It remains only to show that (5.5) is always less than or equal to $1+\phi$. Suppose we replace C_v^* in (5.5) with the lower bound offered by constraint (5.1) for the edge (u,v); then we get that the ratio (5.5) is less than or equal to

$$\min_{\hat{w}\geq 0} \frac{\sum_{v\in \mathrm{UN}(u)}(d_v+\Delta-1)\,\hat{w}_v}{\sum_{v\in \mathrm{UN}(u)}\sum_{t\in T}\hat{w}_v\ \max\{d_v,t\}\, r^*_{(u,v),t}}. \tag{5.7}$$

Consider the polytope that captures the problem of assigning edges between u and $\mathrm{UN}(u)$ to time slots in T:

$$\mathcal{R}=\left\{ r\in[0,1]^{\Delta\times 2n} \;\middle|\; \begin{array}{c} \sum_{t\in T} r_{e,t}\geq 1 \quad \forall\, e\in(u,\mathrm{UN}(u)), \\ \sum_{e\in(u,\mathrm{UN}(u))} r_{e,t}\leq 1 \quad \forall\, t\in T \end{array} \right\}.$$

Then (5.7) is upper bounded by

$$\max_{r\in\mathcal{R}}\ \min_{\hat{w}\geq 0} \frac{\sum_{v\in \mathrm{UN}(u)}(d_v+\Delta-1)\,\hat{w}_v}{\sum_{v\in \mathrm{UN}(u)}\sum_{t\in T}\hat{w}_v\ \max\{d_v,t\}\, r_{(u,v),t}},$$

which in turn is less than or equal to

$$\min_{\hat{w}\geq 0}\ \max_{r\in\mathcal{R}} \frac{\sum_{v\in \mathrm{UN}(u)}(d_v+\Delta-1)\,\hat{w}_v}{\sum_{v\in \mathrm{UN}(u)}\sum_{t\in T}\hat{w}_v\ \max\{d_v,t\}\, r_{(u,v),t}}. \tag{5.8}$$

We can think of (5.7) as an easier version of (5.8). Indeed, while in (5.7) we are to choose weights $\hat{w}$ for a specific solution r^*, in (5.8) we have to choose weights that work for all $r\in\mathcal{R}$. Without loss of generality we can assume that the inner maximization problem assigns the edges to the first Δ time slots. In fact, because the $\mathcal{R}$ is integral, we can equivalently optimize over its extreme points; that is, we can focus on one-to-one assignments between $\mathrm{UN}(u)$ and $[\Delta]$. Letting $d_1,\ldots,d_\Delta$ be the degree sequence of nodes in $\mathrm{UN}(u)$, we get that (5.8) equals

$$\min_{\hat{w}\geq 0} \frac{\sum_{i\in[\Delta]}(d_i+\Delta-1)\,\hat{w}_i}{\min_{\sigma:[\Delta]\to[\Delta]}\sum_{i\in[\Delta]}\hat{w}_i\ \max\{d_i,\sigma(i)\}} = \min_{\hat{w}\geq 0}\frac{\mathrm{UB}(d,\hat{w},)}{\mathrm{LB}(d,\hat{w})}. \tag{5.9}$$

Note that (5.9) is precisely the problem that (LP1) is trying to solve. By Theorem 3.8 it follows that (5.9), and therefore (5.5), is bounded by $1+\phi$. ∎

6. Concluding remarks.

Throughout the paper we have assumed that the transfer graph G is simple. In practice G is typically a multigraph. Luckily, it is easy to modify ALR to handle multigraphs. Let $E(u,v)$ denote the set of edges between u and v. Two modifications must be made to Algorithm 1. First, in line 5 of ALR we choose a vertex u maximizing $\Delta=\sum_{v\in\mathrm{UN}(u)}|E(u,v)|$. Then in line 7, to compute the model $\hat{w}$ we create a degree sequence $d'_1,\ldots,d'_\Delta$ by making $|E(u,v)|$ copies of d_v for each $v\in N(u)$. We solve (LP2) to get weights $\hat{w}'$, and then set $\hat{w}(v)$ to be the sum of $\hat{w}'_i$ for the indices i induced by d_v. With these two modifications, the analysis easily carries over to multigraphs.

Arguably, the main drawback of ALR is that we are required to solve (LP2) in each iteration of the algorithm or, alternatively, to solve the larger (LP4) once. We leave as an open problem the design of purely combinatorial algorithms with an approximation ratio of $1+\phi$ or better.

Appendix A. Bad example for a reasonable heuristic. In this section we study a seemingly reasonable heuristic for unweighted instances, which is based on the scheduling stage of ALR but using the degrees instead of ℓ values to sort the edges. GREEDY-DEGREE first sorts the edges $(u, v) \in E$ in nondecreasing value of $\min(d_u, d_v)$, breaking ties with $\max(d_u, d_v)$; the edges are then scheduled (in sorted order) as early as possible without creating a conflict with the partial schedule built so far.

Gandhi and Mestre [19] showed a family of graphs for which an arbitrary greedy schedule can be a factor $\Omega(\sqrt[3]{n})$ away from the optimum, where n is the number of vertices in the transfer graph. However, for those instances the above heuristic performs well—in fact it computes an optimal solution. Unfortunately GREEDY-DEGREE is not a constant-factor approximation either.

LEMMA A.1. GREEDY-DEGREE *can produce a schedule with cost* $\Omega(\sqrt[4]{n})$ *times the optimum.*

Proof. Our bad instance, shown in Figure A.1, has three layers. The first, second, and third layers contain q, s, and s^2 nodes, respectively. The first and second layers are connected with a complete bipartite graph $K_{q,s}$. The nodes in the third layer are divided into s groups, each forming a K_s that is connected to a single node in the second layer. The parameters q and s will be chosen to get the desired gap.

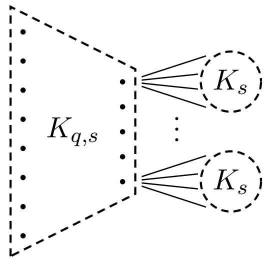


FIG. A.1. *A bad instance for* GREEDY-DEGREE.

Note that nodes in the first and third layers have degree s; thus, the heuristic first schedules the edges in the K_s's, and then we are free to schedule the remaining edges in *any* order as their endpoints have degree s and $q + s$. Suppose a solution S_1 first schedules the edges from the first to the second layer, while S_2 first schedules the edges from the second to the third layer. In S_1 the second-layer nodes are busy for the first q time steps working on the $K_{q,s}$ edges; as a result, every node in the third layer finishes by $\Omega(q)$, and the overall cost is $\Omega(s^2 q)$. On the other hand, in S_2 the third-layer nodes finish by $O(s)$, and the first and second-layer nodes finish by $O(q + s)$; thus, the overall cost is $O((q + s)^2 + s^3)$. Choosing $q = s^{\frac{3}{2}}$, the ratio of the cost of S_1 and S_2 is $\Omega(\sqrt{s})$. Our example has $O(s^2)$ nodes; therefore the greedy schedule can be an $\Omega(\sqrt[4]{n})$ factor away from optimum. $\square$

Acknowledgments. Thanks to Rajiv Gandhi and Dror Rawitz for helpful discussions, and to the anonymous referees whose detailed comments have greatly improved the presentation of the paper. Special thanks to Reuven Bar-Yehuda for explaining the local-ratio technique.

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*, in Proceedings of the 5th International Workshop on Algorithm Engineering (WAE), Springer, Berlin, 2001, pp. 145–158.

[2] N. Bansal, L. Fleischer, T. Kimbrel, M. Mahdian, B. Schieber, and M. Sviridenko, *Further improvements in competitive guarantees for QoS buffering*, in Proceedings of the 14th International Colloquium on Automata, Languages, and Programming (ICALP), Springer, Berlin, 2004, pp. 196–207.

[3] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. S. Naor, and B. Schieber, *A unified approach to approximating resource allocation and scheduling*, J. ACM, 48 (2001), pp. 1069–1090.

[4] A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, and T. Tamir, *On chromatic sums and distributed resource allocation*, Inform. and Comput., 140 (1998), pp. 183–202.

[5] R. Bar-Yehuda, *One for the price of two: A unified approach for approximating covering problems*, Algorithmica, 27 (2000), pp. 131–144.

[6] R. Bar-Yehuda, *Using homogeneous weights for approximating the partial cover problem*, J. Algorithms, 39 (2001), pp. 137–144.

[7] R. Bar-Yehuda, K. Bendel, A. Freund, and D. Rawitz, *Local ratio: A unified framework for approximation algorithms*, ACM Comput. Surv., 36 (2004), pp. 422–463.

[8] R. Bar-Yehuda and S. Even, *A local-ratio theorem for approximating the weighted vertex cover problem*, in Analysis and Design of Algorithms for Combinatorial Problems, North-Holland Math. Stud. 109, North–Holland, Amsterdam, 1985, pp. 27–45.

[9] R. Bar-Yehuda, M. M. Halldórsson, J. (S.) Naor, H. Shachnai, and I. Shapira, *Scheduling split intervals*, SIAM J. Comput., 36 (2006), pp. 1–15.

[10] R. Bar-Yehuda and D. Rawitz, *On the equivalence between the primal-dual schema and the local ratio technique*, SIAM J. Discrete Math., 19 (2005), pp. 762–797.

[11] B. E. Birnbaum and K. J. Goldman, *An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs*, Algorithmica, 55 (2009), pp. 42–59.

[12] F. Chudak, M. X. Goemans, and D. S. Hochbaum, *A primal-dual interpretation of recent 2-approximation algorithms for the feedback vertex set problem in undirected graphs*, Oper. Res. Lett., 22 (1998), pp. 111–118.

[13] K. L. Clarkson, *A modification of the greedy algorithm for vertex cover*, Inform. Process.Lett., 16 (1983), pp. 23–25.

[14] E. G. Coffman, Jr., M. R. Garey, D. S. Johnson, and A. S. LaPaugh, *Scheduling file transfers*, SIAM J. Comput., 14 (1985), pp. 744–780.

[15] W. J. Cook, W. H. Cunningham, W. R. Pullyblank, and A. Schrijver, *Combinatorial Optimization*, John Wiley & Sons, New York, 1998.

[16] R. Gandhi, M. M. Halldórsson, G. Kortsarz, and H. Shachnai, *Improved results for data migration and open shop scheduling*, ACM Trans. Algorithms, 2 (2006), pp. 116–129.

[17] R. Gandhi, M. M. Halldórsson, G. Kortsarz, and H. Shachnai, *Improved bounds for scheduling conflicting jobs with minsum criteria*, ACM Trans. Algorithms, 4 (2008), article 11.

[18] R. Gandhi, S. Khuller, and A. Srinivasan, *Approximation algorithms for partial covering problems*, J. Algorithms, 53 (2004), pp. 55–84.

[19] R. Gandhi and J. Mestre, *Combinatorial algorithms for data migration to minimize average completion time*, Algorithmica, 54 (2009), pp. 54–71.

[20] M. X. Goemans and D. P. Williamson, *The primal-dual method for approximation algorithms and its application to network design problems*, in Approximation Algorithms for NP-Hard Problems, PWS Publishing, Boston, 1996, pp. 144–191.

[21] J. Hall, J. Hartline, A. R. Karlin, J. Saia, and J. Wilkes, *On algorithms for efficient data migration*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2001, pp. 620–629.

[22] N. Immorlica, M. Mahdian, and V. S. Mirrokni, *Cycle cover with short cycles*, in Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS), Springer, Berlin, 2005, pp. 641–653.

[23] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani, *Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP*, J. ACM, 50 (2003), pp. 795–824.

[24] S. Khuller, Y.-A. Kim, and Y.-C. (J.) Wan, *Algorithms for data migration with cloning*, SIAM J. Comput., 33 (2004), pp. 448–461.

[25] Y.-A. Kim, *Data migration to minimize the average completion time*, J. Algorithms, 55 (2005), pp. 42–57.

[26] Y.-A. Kim, S. Khuller, and A. Malekian, *Improved algorithms for data migration*, in Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), Springer, Berlin, 2006, pp. 164–175.

[27] D. Marx, *Complexity results for minimum sum edge coloring*, Discrete Appl. Math., 157 (2009), pp. 1034–1045.

[28] T. Nishizeki and K. Kashiwagi, *On the* 1.1 *edge-coloring of multigraphs*, SIAM J. Discrete Math., 3 (1990), pp. 391–410.

[29] M. Queyranne and M. Sviridenko, *New and improved algorithms for minsum shop scheduling*, in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2000, pp. 871–878.

[30] D. P. Williamson, *The primal-dual method for approximation algorithms*, Math. Program., 91 (2002), pp. 447–478.

[31] D. P. Williamson, L. A. Hall, J. A. Hoogeveen, C. A. J. Hurkens, J. K. Lenstra, S. V. Sevast'janov, and D. B. Shmoys, *Short shop schedules*, Oper. Res., 45 (1997), pp. 288–294.