

Data Migration to Minimize the Average Completion Time *

Yoo-Ah Kim †

1 Introduction

We consider the *data migration* problem, which is the problem of finding an efficient schedule to migrate data in a network. Large storage systems require careful load balancing across several devices based on the expected data access pattern. The data access pattern is not static and therefore, the data layout needs to be rearranged as the data access pattern changes over time. If the changes are drastic, we need to recompute a new optimal layout based on new workloads [4] and migrate the data from its current layout to the new layout. The migration should be done quickly since the devices may be running inefficiently until the migration is finished.

We define the problem as follows. Given an initial and final configuration of data on devices, we consider the *transfer graph* $G = (V, E)$, where each vertex $v \in V$ represents a storage device and each edge $e = (u, v) \in E$ corresponds to a migration that must be done from u to v . We assume that the underlying network is fully connected. Each edge e has its length p_e which represents the time required to migrate the data from source to destination. The crucial constraint on this problem is that any two adjacent edges *cannot be scheduled* at the same time.

In the same model, the problem of minimizing the *makespan* was studied in [2, 5]. It can be reduced to the problem of finding the chromatic index χ' of a multi-graph when the lengths of edges are all the same and there is a 9/8-approximation algorithm [3, 8]. Hall et al. [5] suggested a simple 3/2-approximation algorithm. When edges lengths are arbitrary integers, there is a simple 2-approximation algorithm [2].

Since the performance of a device is degraded while the device is involved in the migration, it is beneficial to minimize the average completion times over all devices. This problem was first suggested by Coffman et al. [2] but there has been no result for this problem. We develop constant factor approximation algorithms.

Another interesting objective is to minimize the average completion time of data migration jobs. When the lengths of edges are all the same, it was studied under the name of the *minimum edge color sum* problem: Given a graph $G = (V, E)$, find a valid edge coloring $\phi : E \rightarrow N$, that minimizes $\sum_{e \in E} \phi(e)$. Bar-Noy et al. [1] proved that it is NP-hard and any compact coloring¹ of graph G is a 2-approximation for the problem. When the lengths of edges are arbitrary integers, Halldorsson et al. [7] presented

a 12-approximation algorithm.

We can consider the data migration problem as the *resource constrained scheduling* problem in which each job requires exactly two resources. In the resource constrained scheduling problem, we are given a collection of jobs and resources and assume that each job requires an exclusive access to a particular subset of resources to process.

Our Results: We prove that minimizing the average weighted vertex completion time is NP-hard and develop a 3-approximation algorithm when each edge has unit length and a 9-approximation when each edge has an arbitrary integer length. These answer an open problem in [2]. For the edge completion time problem, we present a 10-approximation algorithm when each edge has an arbitrary integer length, for which the best previous result was 12-approximation [7]. These algorithms give constant approximation for the resource constrained scheduling problem when the number of resources each job requires is constant. In particular, we obtain a $(6m - 2)$ -approximation for the job completion time of arbitrary integer length case, which improves the previous $2m(2m - 1)$ -approximation [7].

2 Algorithm for the Vertex Completion Time

We are interested in finding a schedule that minimizes the average weighted completion time over all *vertices*, where the completion time of a vertex is the time when we finish all edges incident to it. We can prove that it is NP-hard when vertices have arbitrary weights even though all edges have the same length [9]. We formulate the problem as integer program, which draws on an IP formulation given in [6]. We have two variables C_v and C_e which represent the completion times for vertex v and edge e , respectively. Denote a set of edges incident to v as $N(v)$. Define $p(S)$ to be $\sum_{e \in S} p_e$ and $p(S^2)$ to be $\sum_{e \in S} p_e^2$. Our objective function is $\min \sum_v w_v \cdot C_v$. Constraints are

$$(2.1) \sum_{e \in S(v)} p_e C_e \geq \frac{p(S(v))^2 + p(S(v)^2)}{2} \quad \forall S(v) \subseteq N(v)$$

$$(2.2) \quad C_v \geq C_e \quad \forall v, e \text{ where } e \in N(v)$$

$$(2.3) \quad C_v \geq p(N(v)) \quad \forall v$$

Constraint (2.1) follows since edges in $N(v)$ should be processed in some order [6]. We relax the integrality restriction on C_v, C_e and find an optimal solution of the LP. Although the size of this LP is exponential, it is solvable in polynomial time via the ellipsoid algorithm since there is a polynomial-time separation algorithm for the exponentially large class of Constraints (2.1) [10]. Define C_v^* (C_e^*) to be the completion time of v (e) in an optimal solution of the LP. For a vertex v and $e \in N(v)$, let us define $S_e(v)$ as $\{e' | e' \in N(v) \wedge C_{e'}^* \leq C_e^*\}$. Constraints (2.1) give the following lemma.

*Full paper is available at <http://www.cs.umd.edu/~ykim/paper.ps>. Research supported by NSF Award CCR-0113192

†Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail : ykim@cs.umd.edu.

¹an edge coloring ϕ is compact if and only if every edge e with $\phi(e) = j$, has neighboring edges with all color $1 \dots j - 1$.

LEMMA 2.1. For $e \in N(v)$, $C_e^* \geq p(S_e(v))/2$.

2.1 When edges have unit lengths We construct our schedule, which we call *Ordered List Scheduling(OLS)* as follows: sort edges by nondecreasing order of C_e^* . At time t , we scan edges not yet scheduled and process an edge $e = (u, v)$ at t if and only if no edge in $S_e(u) \cup S_e(v)$ is assigned at t . Denote the completion time of $e(v)$ in our schedule as $\tilde{C}_e(\tilde{C}_v)$. The omitted proofs are given in [9].

LEMMA 2.2. For $e = (u, v)$, $\tilde{C}_e \leq p(S_e(u)) + p(S_e(v))$.

THEOREM 2.1. *OLS gives 3-approximation for the vertex completion time when edge lengths are all the same.*

2.2 When edges have arbitrary integer lengths

Note that Lemma 2.2 does not hold in this case. Instead, we use the following recursive algorithm: Let $p(G)$ be $\max_v p(N(v))$ in G . Sort edges in nondecreasing order of C_e^* . Initially $G_0 = G$. We split $G_{i-1} = (V, E)$ into $G_i = (V, E_i)$ and $G'_i = (V, E'_i)$ as follows. $E_i = \emptyset$ in the beginning. Scan edges in sorted order and add an edge e to E_i if $p(G_i) \leq p(G_{i-1})/2$ after adding e to E_i . Set $E'_i = E_{i-1} - E_i$. We repeat this until E_k is empty. After that, we schedule edges in the order of $E'_k, E'_{k-1} \dots E'_1$. When scheduling edges in E'_i , we use *List Scheduling(LS)*, in which whenever two adjacent vertices become available and there are any remaining edges between them, one of those edges is scheduled.

LEMMA 2.3. For $e = (u, v) \in E'_i$, $C_e^* \geq p(G_{i-1})/4$.

LEMMA 2.4. For $e = (u, v) \in E'_i$, $\tilde{C}_e \leq 3p(G_{i-1})$.

Proof. We prove this by induction. If $e \in E'_k$, then $\tilde{C}_e \leq 2p(G'_k) = 2p(G_{k-1})$. Now we will prove that if we can finish all edges in $E'_k \dots E'_{i+1}$ within $3p(G_i)$, then all edges in $E'_k \dots E'_i$ can be finished within $3p(G_{i-1})$. To do this, it is enough to show that scheduling edges in E'_i needs at most $3p(G_{i-1})/2$ additional time since $3p(G_i) \leq 3p(G_{i-1})/2$. Consider an edge $e = (u, v) \in E'_i$. Denote a set of edges incident to v in a graph H as $N_H(v)$. $LS(G'_i)$ finishes e in $p(N_{G'_i}(u) \cup N_{G'_i}(v)) \leq p(N_{G'_i}(u)) + p(N_{G'_i}(v)) - p_e$ time. Since $e \notin E_i$, w.l.o.g., we assume that $p(N_{G_i}(v)) + p_e > p(G_{i-1})/2$. Then $p(N_{G'_i}(u)) + p(N_{G'_i}(v)) - p_e < p(N_{G'_i}(u)) + p(N_{G'_i}(v)) + p(N_{G_i}(v)) - p(G_{i-1})/2 = p(N_{G'_i}(u)) + p(N_{G_{i-1}}(v)) - p(G_{i-1})/2 \leq 3p(G_{i-1})/2$.

THEOREM 2.2. *The algorithm gives us 9-approximation for the average completion times of vertices.*

Proof. Let $e^v = (u, v) \in E'_i$ be the last scheduled edge in $N(v)$. $\tilde{C}_v = \tilde{C}_{e^v} \leq 3p(G_{i-1})/2 + p(N_{G_{i-1}}(u)) + p(N_{G_{i-1}}(v)) - p(G_{i-1})/2 \leq 2p(G_{i-1}) + p(N(v)) \leq 9C_v^*$.

3 Algorithm for the Edge Completion Time

Our objective should be $\min \sum_e w_e \cdot C_e$ and we only need Constraints (2.1).

LEMMA 3.1. *By slightly modifying LS, we can find a schedule in which $\sum_{e \in E'_i} w_e \tilde{C}_e \leq \sum_{e \in E'_i} w_e (\frac{9}{4}p(G_{i-1}) + p_e)$ for all i .*

Proof. We break \tilde{C}_e into two parts: the delay caused by finishing $E'_j (j > i)$ and the delay after we start scheduling edges in E'_i . We know that the first part cannot be greater than $\frac{3}{2}p(G_{i-1})$. Denote the second part as \bar{C}_e . We find a schedule using *LS* and see if $\sum_{e \in E'_i} w_e \bar{C}_e \leq \sum_{e \in E'_i} w_e \cdot \frac{3}{4}p(G_{i-1})$. If it is satisfied, we are done. If not, we schedule the edges in the reverse order. In other words, if e was scheduled from $\bar{C}_e - p_e$ to \bar{C}_e in the given *LS*, we schedule e from $\frac{3}{2}p(G_{i-1}) - \bar{C}_e$ to $\frac{3}{2}p(G_{i-1}) - \bar{C}_e + p_e$. Then the modified completion time of e is at most $3p(G_{i-1}) - \bar{C}_e + p_e$. Thus $\sum_{e \in E'_i} w_e \tilde{C}_e \leq \sum_{e \in E'_i} w_e (3p(G_{i-1}) - \bar{C}_e + p_e) < \sum_{e \in E'_i} w_e (3p(G_{i-1}) + p_e) - \frac{3}{4}p(G_{i-1}) \sum_{e \in E'_i} w_e = \sum_{e \in E'_i} w_e (\frac{9}{4}p(G_{i-1}) + p_e)$.

THEOREM 3.1. *The algorithm gives us 10-approximation for the average completion time over all edges.*

4 Resource Constrained Scheduling

All the algorithms in this paper can be extended to the general resource constrained scheduling problem. To minimize the resource completion time in which each job uses up to m resources, we have $(2m - 1)$ -approximation when each job has unit length and $(8m - 7)$ -approximation when each job has arbitrary integer length. To minimize the job completion time in which each job has arbitrary integer length, we have $(6m - 2)$ -approximation algorithm[9].

References

- [1] A. Bar-Noy, M. Bellare, M. M. Halldorsson, H. Shachnai, and T. Tamir, On chromatic sums and distributed resource allocation. *Information and Computation*, Vol.140, pp. 183-202, 1998.
- [2] 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.
- [3] M.K. Goldberg, Edge-coloring of multigraphs: recoloring technique. *J. Graph Theory*, 8:121-137, 1984.
- [4] L. Golubchik, S. Khanna, S. Khuller, R. Thurimella and A. Zhu. Approximation algorithms for data placement on parallel disks. *11th SODA*, pp. 223-232, 2000.
- [5] J. Hall, J. Hartline, A. Karlin, J. Saia and J. Wilkes. On algorithms for efficient data migration. *12th Symposium on Discrete Algorithms*, pp. 620-629, 2001.
- [6] L.A. Hall, A. S. Schulz, D.B. Shmoys, and J. Wein. Scheduling to minimize average completion time: off-line and on-line algorithms. *Mathematics of Operations Research* 22, pp. 513-544, 1997.
- [7] M. M. Halldorsson, G. Kortsarz, H. Shachani, Minimizing average completion of dedicated tasks and partially ordered sets, *4th APPROX*, pp. 114-126, 2001.
- [8] D.S. Hochbaum, T. Nishizeki, and D.B. Shmoys. A better than "Best Possible" algorithm to edge color multigraphs. *J. of Algorithm* 7:79-104, 1986.
- [9] Y. Kim, Data migration to minimize the average completion time, Available at <http://www.cs.umd.edu/~ykim/paper.ps>.
- [10] M. Queyranne, Structure of a simple scheduling polyhedron. *Math. Programming* 58, pp. 263-285, 1993.