制限つきページ移動問題に対しての新しい枠組

スザンネ アルベルス
マックスプランクインスティテュート

古賀 久志
東京大学理学部情報科学科

本論文では、マルチプロセッサシステムにおける分散共有メモリのメモリ管理に関する問題を取り扱う。この問題の目的は書き込み可能ページについてページアクセスのコストが低くなるようにページをうまく移動させて配置するページ配置のアルゴリズムを見つけることである。この問題は、最近オンラインアルゴリズムのコンペティティブによる解析の分野で注目されている問題である。

本研究では特に、これまでほとんど結果の得られていないオンライン制限つきページ移動問題に対して、ある部分問題を提唱しそれを解析することにより、より深い考察を与えた。

A New Framework on the Constrained Migration Problem

Susanne Albers
Max-Planck-Institut für Informatik

Hisashi Koga
The University of Tokyo

This paper deals with problems that arise in the distributed shared memory management of large multiprocessor systems. The purpose of these problems is to find an efficient migration strategy which locates each writable page at an appropriate processor by migration to reduce the total access cost. This problem is focused recently in the competitive analysis for on-line algorithms.

Especially in this paper we examine the on-line constrained migration problem deeper about which only few result is known by defining and attacking some special case of the problem called *the direct-mapped constrained migration problem.*

1 Introduction

This paper deals with problems that arise in the memory management of large multiprocessor systems. Such multiprocessing environments typically consist of a network of processors, each of which has its local memory. A global shared memory is modeled by distributing the physical pages among the local memories. Accesses to the global memory are then accomplished by accessing the local memories. Suppose a processor v wants to access a memory address from page P. If P is stored in v's local memory, then this access operation can be accomplished locally. Otherwise, v determines a processor u holding the page and sends a request to u. The desired information is then transmitted from u to v, and the communication cost incurred thereby is proportional to the distance from u to v. If v has to access page P frequently, it may be worthwhile to migrate or replicate the entire page of P from u to v because subsequent accesses will become cheaper. However, transmitting an entire page incurs a high communication cost proportional to the page size times the distance from u to v.

Finding efficient migration or replication strategies is studied very much from both of applied and theoretical sides. Especially from the theoretical point of view, three main problems are defined. The *migration problem* [BS89, ABF93] deals with a writable page under the condition that the number of copies of the page is limited to one. This condition is reasonable to avoid the problem of keeping multiple copies of the page consistent. The purpose of the migration problem is to decide in which local memory the single copy of the page should be stored so that a sequence of memory accesses can be processed at low cost. On the other hand, if a page is read-only, it is possible to keep several copies of the page in the system without considering the consistency problem. In the *replication problem* [BS89, AK94] we have to determine which local memories should contain copies of the read-only page. The *allocation problem* [BFR92, ABF93] allows multiple copies of a writable page. This requires us to handle the additional broadcasting cost for keeping all copies of the page consistent. All of these three problem are focused recently in the competitive analysis area. One reason for this is the association of these problems with the important *k-server problem with excursions* [MMS88].

However, all the above problems used the assumption that each local memory has infinite capacity, i.e. there is always vacant space for replicating or migrating the page in every local memory, which made the problems rather impractical because the conflicts between multiple different pages are not considered. Recently Bartal *et al.* introduced the more practical version of the problems called the *constrained migration problem* and *constrained allocation problem*, which try to solve many migration (respectively allocation) problems each for an individual page, simultaneously in a single distributed shared memory under the constraint that every local memory has fixed finite capacity. In this version when a local memory is full and a page is copied into that local memory, some other page must be dropped from that memory to make room and must be stored somewhere else.

For the constrained migration problem we deal with pages with the equal size. This makes sense in the context of distributed virtual memory. In addition every local memory is assumed to have the same capacity. A local memory consists of k blocks, $[1], [2] \cdots, [k]$. A page copy in a memory must be stored in one of its k blocks. Therefore one local memory can accommodate at most k pages at the same time. This paper examines the on-line constrained migration problem deeper by attacking some special case of the problem. In order to analyze the performance of an on-line algorithm we will use *competitive analysis* [ST85], the worst case ratio of the cost incurred by an on-line algorithm and the cost incurred by an optimal off-line algorithm.

Although for the migration problem a lot of results have been obtained already, for the constrained migration problem only few result is known. For the migration problem Black and Sleator [BS89] proved that no deterministic on-line algorithm cannot be better than 3-competitive for any graphs, even if the graph consists of only two nodes connected by a single edge. They actually presented 3-competitive algorithms for tree and uniform graphs. A uniform graph is a complete graph in which all edges have the same length. For general graphs, Awerbuch *et al.* [ABF93] have presented a 7-competitive deterministic on-line migration algorithms.

On the other hand, for the constrained migration problem, the only result obtained so far is by Bartal *et al.* [BFR92]. They described a $O(kn)$-competitive deterministic on-line algorithm for uniform graphs with n nodes. However whether this competitive ratio is the optimal one for uniform graphs is still open. Moreover for any other topology no competitive algorithm is found yet.

In this paper to shed lights onto the hard constrained migration problem we define and study a new framework called the *direct-mapped constrained migration* problem*. The direct-mapped constrained migration problem is the constrained migration problem with the restriction that each processor manages pages in its local memory using the same hash function h. That is, whichever memory a page A belongs to, it is stored in the block $[h(A)_{\mathrm{mod}\,k} + 1]$ of that memory. Because introducing this hashing technique eliminates the possibility of the conflict between two pages A and B such that $h(A)_{\mathrm{mod}\,k} \neq h(B)_{\mathrm{mod}\,k}$, what we have to deal with is only the conflict between pages which take the same hash value, having the possibility of being stored in one identical block. Thus, we can analyze the direct-mapped constrained migration problem by dividing it according to k block numbers. Therefore in the direct-mapped constrained migration problem we concentrate on one particular block number i $(1 \leq i \leq k)$. Let F be the number of pages A such that $h(A)_{\mathrm{mod}\,k} + 1 = i$. The problem is to decide where to locate F pages among n block $[i]$s, one for each processor. Note the constrained migration problem collapses to the *direct-mapped* constrained migration problem if $k = 1$. On the other hand if $F = 1$, then the direct-mapped constrained migration problem is reduced to the migration problem. Thus this newly defined problem is positioned in the middle of the migration problem and the constrained migration problem.

In this paper we develop a number of on-line algorithms for the direct-mapped constrained migration problem. All our algorithms are quite simple. In Section 3, the lower bounds of the competitiveness of deterministic on-line algorithms are investigated. We show for any networks no deterministic on-line algorithm is better than 3-competitive. We also prove that for some specific topology no deterministic on-line algorithm is better than $(n-2)$-competitive where n is the number of nodes in the network. In Section 4 first we present an optimal 3-competitive deterministic on-line algorithm when the graph consists of only 2 nodes. Next we develop an 8-competitive deterministic on-line algorithm for uniform graphs, called Concurrent-M which is based on algorithm M [BS89] for the migration problem. What is interesting about our results is that in the constrained migration problem with $k = 1$, we can find a $O(1)$-competitive deterministic on-line algorithm for uniform graphs, thus breaking the previously best upper bound of $O(n)$-competitiveness obtained by applying the $O(kn)$-competitive constrained migration algorithm by Bartal *et al.* to the case $k = 1$. Observing this fact, in Section 5 we give an interesting future work on the constrained migration problem.

*We named this problem the direct-mapped constrained migration problem, because a similar hashing technique is adopted in the context of the direct-mapped caching protocol for snoopy caching systems.

2 Problem Statement and Competitive Analysis

Formally, the direct-mapped constrained migration problem can be described as follows. We are given an undirected graph $G = (V, E)$. Each node in G corresponds to a processor and the edges represent the interconnection network. The number of nodes $|V|$ is denote by n. Associated with each edge is a *length* that is equal to the distance between the connected processors. We assume that the edge lengths satisfy the triangle inequality. Let δ_{ij} denote the length of the shortest path between node i and node j.

Each node has its own local memory. One local memory is divided into k *blocks*, $[1], [2], \cdots, [k]$. One block can exactly store one page copy. In the direct-mapped constrained migration problem, all nodes use the same hash function $h(A)$ to determine the unique block in which page A will reside. A node cannot contain two pages A and B at the same time in its local memory, such that $h(A)_{\mathrm{mod}\,k} = h(B)_{\mathrm{mod}\,k}$. On the contrary, at any node there is no conflict between two pages such that $h(A)_{\mathrm{mod}\,k} \neq h(B)_{\mathrm{mod}\,k}$. Thus we can analyze the direct-mapped constrained migration problem by dividing it according to k block numbers. In the following, we concentrate on one particular block number i $(1 \leq i \leq k)$. Let F be the number of pages such that $h(A)_{\mathrm{mod}\,k}+1 = i$ and $f_1, f_2, \cdots, f_F$ be the F pages whose hash value equal to i.

We say that *a node v has a page A* if the page A is contained in block $[i]$ of v's memory. A node v is said to be *empty* if v does not hold any page in block $[i]$ of its memory. *A request to page A at a node v* occurs if v wants to read or write A. The request can be satisfied at zero cost if v has A. Otherwise the request incurs a cost equal to the distance from v to the node u with page A (i.e. δ_{uv}). Immediately after a request to page A at a node v, if v is empty, A may be migrated into v's local memory. The cost incurred by this *migration* is $d \cdot \delta_{uv}$. Here d denotes the page size factor. In case v has another page B already, in order to move A to v either after B is dropped from v and migrated to some other processor, A is migrated to v or A and B are swapped. The cost incurred by this *swapping* is $2d \cdot \delta_{uv}$.

A direct-mapped constrained migration algorithm is usually presented with an entire sequence of requests that must be served with low total cost. A direct-mapped constrained migration algorithm is *on-line* if it serves every request without knowledge of any future requests.

We study the direct-mapped constrained migration problem under the assumption that $F \leq n$, which can be realized easily by the proper choice of h. If $F = 1$, since a direct-mapped constrained migration algorithm need not to use swapping operations to change the location of the only file f_1, this case is reduced to the migration problem. On the contrary, if $F = n$, then an algorithm necessarily has to use swapping operations to change the configuration of pages. Throughout this paper although our algorithms are described for any arbitrary value of F $(\leq n)$, we always analyze them assuming that $F = n$ for the simplicity.

Competitive analysis [ST85] is a powerful means to analyze the performance of an on-line algorithm. In a competitive analysis, the cost incurred by an on-line algorithm is compared to an *optimal off-line algorithm.* An optimal off-line algorithm knows the entire request sequence in advance and can serve it with minimum cost. Given a request sequence σ, let $C_A(\sigma)$ denote the cost incurred by on-line algorithm A in serving σ and let $C_{OPT}(\sigma)$ denote the cost incurred by the optimal off-line algorithm OPT. Usually a deterministic on-line algorithm A is called c-competitive if there exists a constant a such that for every request sequence

$$C_A(\sigma) \leq c \cdot C_{OPT}(\sigma) + a.$$

The *competitive factor* of an on-line algorithm A is the infimum of all c such that A is c-competitive.

3 Lower Bounds

Before introducing our algorithms, this section investigates the lower bounds of the competitiveness achievable by deterministic on-line algorithms. We obtain two kinds of lower bounds. The first lower bound represents the limitation of the power of on-line algorithms no matter how simple the underlying graph structure may be. The second lower bound demonstrates how badly any deterministic on-line algorithm behaves for some specific topology.

Theorem 1 *Let A be a deterministic on-line algorithm for the direct-mapped constrained migration problem. Then A cannot be better than 3-competitive, even on a graph consisting of two nodes.*

Proof: Let s and t be two nodes connected by an edge of length 1, and f_1 and f_2 be pages whose location need to be managed. Consider the situation in which requests are generated only to f_1. We can view this situation as the migration problem where the page size factor is $2d$. Thus the lower bound of 3-competitiveness for the migration problem in [BS89] also holds for the direct-mapped constrained migration problem. □

Next we prove the existence of some specific topology against which no deterministic on-line algorithm is better than $(n-2)$-competitive. The following star H is considered as the example. Let $v_1, v_2, \cdots, v_n$ be the nodes in H and v_1 be the center node of the star. The length of the edges is set as follows.

$$\delta_{v_1 v_i} = \begin{cases} 1 & \text{for } 2 \leq i \leq n-1 \\ n-2 & \text{for } i = n \end{cases}$$

Theorem 2 *Let A be a deterministic on-line algorithm for H for the direct-mapped constrained migration problem. Then A cannot be better than $(n-2)$-competitive.*

Interestingly this theorem certifies the difference between the migration problem and the direct-mapped constrained migration problem, because for the migration problem Black and Sleator [BS89] developed a 3-competitive deterministic on-line algorithm for all trees including any star.

Proof of Theorem 2: For any deterministic on-line algorithm A, we will construct a request sequence σ which can be arbitrary long, such that $C_A(\sigma)$ is at least $(n-2)$ times the cost incurred by some off-line algorithm OFF which may not necessarily be the optimal off-line algorithm. This yields the theorem. Assume that initially both A and OFF have the same page p at node v_n. Let $V' = V - \{v_n\}$ in the subsequent proof.

We will construct σ as follows. The first request is generated at v_1 to the page p. This request is repeated until A swaps p and some other page q which is stored in one of the nodes in V'. The next request is also generated at v_1 to the page q which has just been stored at v_n. This request is also repeated until A moves q to some processor in V'. Similarly from them on at every point a request is generated at v_1 to the page which v_n holds. Therefore every time there is a request at v_1, it takes $\delta_{v_1 v_n} = n-2$ for A to satisfy the request.

The request sequence can be partitioned into phases in the following way. The first phase begins with the first request. The first phase ends after $n-1$ distinct pages including p have been requested at v_1 during the phase and just before the remaining nth page r is requested. The second phase begins with the request at v_1 to this page r. The second phase ends in the same way as the first phase. The subsequent phases are determined similarly.

We show that at any phase, the cost incurred by A is at least $(n-2)$ times the cost incurred by OFF. Suppose σ' is the subsequence of σ which corresponds to the phase and the length of σ' is l. First consider the cost incurred while A processes σ'. Let m be the number of swappings by A for processing σ', counting the swapping which takes place after A satisfies the last request of σ'. The cost of a single swapping is at least $2(n-2)d$. More precisely speaking, it costs $2(n-2)d$ if the swapping is done between v_1 and v_n. Else if the swapping occurs between v_i and v_n $(2 \leq i \leq n-1)$, it costs $2(n-1)d$. Since a swapping takes place if and only if the kind of the page requested at v_1 is changed and v_1 requests $n-1$ distinct pages during this phase, $m \geq n-1$. Thus the total cost for m swappings is at least $2(n-1)(n-2)d$.
The cost incurred by satisfying requests in σ' whose length is l is $(n-2)l$, since each request is satisfied at the cost of $n-2$. Thus we have

$$C_A(\sigma') \geq 2(n-1)(n-2)d + (n-2)l. \qquad (1)$$

Next we show that the following off-line algorithm OFF can serve σ' at the cost of $2(n-1)d + l$. At the beginning of σ' before the first request OFF swaps the page located at v_n and the page r which is first requested at v_1 in the next phase and after this swapping OFF never changes the location of any pages throughout the phase. We remark that r is never requested in the entire σ'. Now consider $C_{OFF}(\sigma')$. The cost incurred by the only swapping which occurs at the beginning of the phase is at most $2(n-1)d$. The cost of OFF satisfying a single request is at most 1, since any page requested at v_1 during the phase is necessarily located at one of the nodes in V' in OFF's configuration. Since the length of σ' is l, the total cost for OFF to satisfy requests contained in σ' is at most l. Thus we have

$$C_{OFF}(\sigma') \leq 2(n-1)d + l. \qquad (2)$$

By comparing (1) to (2), we can conclude

$$C_A(\sigma') \geq (n-2)C_{OFF}(\sigma').$$

At the beginning of each round v_n has the same page both in A and in OFF. Therefore we can extend σ arbitrarily by repeating this construction. Thus the proof ends. □

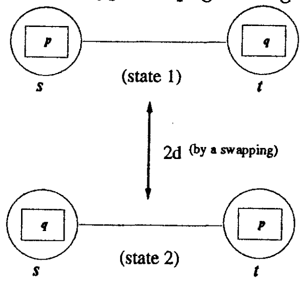
4 Our Algorithms

In this section we describe our on-line algorithms for the direct-mapped constrained migration problem when the network topology is a practical one such as 2 nodes, uniform graphs. Section 4.1 deals with an optimal 3-competitive deterministic algorithm for 2 nodes. In section 4.2, we present a $O(1)$-competitive deterministic algorithm for uniform graphs.

4.1 2 nodes

This section introduces a deterministic on-line algorithm for 2 nodes that is 3-competitive. From Theorem 1, we conclude that this algorithm is optimal. To describe our algorithm, we need some notation. Again let s ant t be two nodes connected by a single edge of length 1. We denote two pages whose location need to be managed by p and q. In 2-node networks, there are only two types of page configurations as illustrated in Figure 1: (state 1) s has p and t has q and (state 2) s has q and t has p. Any algorithm must take either (state 1) or (state 2) at any time. The transition between (state 1) and (state 2) costs $2d$. Under these notations our algorithm is described as follows.

Figure 1: Two types of page configurations



Algorithm TN: Every node v has two counters c_v^p and c_v^q. Initially all counters are set to 0. If TN is in (state 1) and $c_s^q + c_t^p < 4d$, every time s requests q or t requests p, respectively c_s^q or c_t^p is incremented by 1. If $c_s^q + c_t^p = 4d$ after the increment, the algorithm changes its configuration to (state 2) using a swapping operation and c_s^q and c_t^p are reset to 0. On the other hand, if TN is in (state 2) and $c_s^p + c_t^q < 4d$, every time s requests p or t requests q, respectively c_s^p or c_t^q is incremented by 1. If $c_s^p + c_t^q = 4d$ after the increment, the algorithm changes its configuration to (state 1) using a swapping operation and c_s^p and c_t^q are reset to 0.

Theorem 3 *Algorithm TN is 3-competitive for two nodes.*

4.2 Uniform Graphs

This section deals with the case when the network topology is the uniform graph. W.l.g. we may assume that the length of each edge equals to 1. We present an 8-competitive deterministic on-line algorithm for uniform graphs. As the name implies, this algorithm is thought of as the concurrent version of algorithm M [BS89] for the migration problem.

Algorithm Concurrent-M : Each processor v has F counters $c_v^{f_i}$ $(1 \leq i \leq F)$. All counters are initialized to 0. Concurrent-M processes a request at node v to page f_i as follows. If v has f_i already, then the request is free and nothing happens. If v does not have f_i and $c_v^{f_i} < 2d$, then the algorithm increments $c_v^{f_i}$ and chooses some other non-zero counter among $\{c_u^{f_i} | u \in V\}$ if there is one, and decrement it. When $c_v^{f_i}$ reaches $2d$ after the increment, if v is empty, f_i is migrated to v and $c_v^{f_i}$ is reset to 0. Otherwise f_i is swapped with the page $f_j (i \neq j)$ which v currently holds, and $c_v^{f_i}$ and $c_u^{f_j}$ are reset to 0, where u denotes the node which stores f_i before the swapping.

In the above swapping, we say f_i is swapped *actively* and f_j is swapped *passively*. The description of the algorithm implies any counter value does not exceed $2d$.

Theorem 4 *Concurrent-M is 8-competitive for uniform graphs.*

Before the proof, we remark one lemma which plays an important role. Nearly the same lemma is proved in [BS89].

Lemma 1 $\forall$ *page* f, $\sum_{v \in V} c_v^f \leq 2d$.

Proof: We prove this lemma by induction. Initially $\sum_{v \in V} c_v^f = 0$, since all counters are zero. $\sum_{v \in V} c_v^f$ is incremented only if one counter is incremented and all other counter values are 0. This is because if there is another non-zero counter, that counter is decremented so that $\sum_{v \in V} c_v^f$ remains unchanged. Since a counter value cannot exceed $2d$ from the description of the algorithm, also $\sum_{v \in V} c_v^f$ cannot be larger than $2d$. □

This lemma implies two important facts.

1. Just before a page f is swapped actively to node v, $c_v^f = 2d$ and all other counters associated with f are 0.
2. After the page f is swapped actively, all counters associated with f are 0.

Proof of Theorem 4: As is mentioned in Section 2, we analyze the algorithm assuming that $F = n$. Let $C_{CM}(\sigma)$ be the cost incurred by Concurrent-M to process a request sequence σ. We shall show that, for any algorithm A, $C_{CM}(\sigma) \leq 8C_A(\sigma)$ for any request sequence σ. Our proof uses the standard technique comparing simultaneous runs of Concurrent-M and A on σ by merging the actions generated by Concurrent-M and A into a single sequence of events. This sequence contains three types of events: (Type I) Concurrent-M swaps pages, (Type II) A swaps pages, and (Type III) a request is satisfied by both Concurrent-M and A. We shall give a non-negative potential function Φ (initially 0) such that the following inequality holds for all kinds of events.

$$\Delta C_{\mathrm{CM}} + \Delta\Phi \leq 8\Delta C_{\mathrm{A}} \qquad (3)$$

where the Δ indicates the change in the value as a result of the event. Summing up (3) for all events results in

$$C_{CM}(\sigma) + \Phi_{\mathrm{end}} - \Phi_{\mathrm{start}} \leq 8C_A(\sigma).$$

Since $\Phi_{\mathrm{start}} = 0$ and $\Phi_{\mathrm{end}} \geq 0$ from the definition of the potential function, we have

$$C_{CM}(\sigma) \leq 8C_A(\sigma).$$

Thus the proof ends. It remains to specify the potential and verify (3) for all events.

Let s^f be the node which has f in Concurrent-M and t^f be the node which has f in A. The potential function Φ is defined as:

$$\Phi = \sum_f \Phi_f$$

$$\Phi_f = \begin{cases} 5\sum_{v \in V} c_v^f & \text{if } s^f = t^f. \\ 4d - c_t^f + 3\sum_{\substack{v \in V \\ v \neq t}} c_v^f & \text{if } s^f \neq t^f \end{cases}$$

Note the initial value of Φ equals to 0 and always Φ is non-negative.

From now on (3) is proved for all events. In the subsequent proof we omit the specification of the page in the expression of counters when obvious.

(Type I): Concurrent-M swaps pages.
Suppose that page p is swapped actively from s to s' and page q is swapped passively from s' to s. Therefore as the result of the swapping, $c^p_{s'}$ is reset from $2d$ to 0 and c^q_s is reset from some positive value l to 0. Furthermore let t be the location of p in A and u be the location of q in A. In this case $\Delta C_{CM} = 2d$ and $\Delta C_A = 0$. So we must show that $\Delta\Phi \leq -2d$. Trivially, $\Delta\Phi = \Delta\Phi_p + \Delta\Phi_q$. We calculate $\Delta\Phi_p$ and $\Delta\Phi_q$ separately and then obtain $\Delta\Phi$ by summing up them.

First consider $\Delta\Phi_p$. There are three cases depending on whether s, s' coincide with t. Lemma 1 and deduced two facts make the calculation of $\Delta\Phi_p$ so simple.

$$\begin{aligned}
s' = t \quad &: \quad \Delta\Phi_p = 5\sum 0 - (4d - 2d + \sum 0) \\
&\qquad = -2d \\
s = t \quad &: \quad \Delta\Phi_p = (4d - 0 - 3\sum 0) - 5 \cdot 2d \\
&\qquad = -6d \\
s, s' \neq t \quad &: \quad \Delta\Phi_p = (4d - 0 - 3\sum 0) \\
&\qquad - (4d - 0 - 3 \cdot 2d) = -6d
\end{aligned}$$

Next calculate $\Delta\Phi_q$. For clearness, we denote the counter value of c_s before the swapping simply by c_s and that after the swapping by $c'_s(=0)$. There are also three cases depending on whether s, s' coincides with u. In the right side of three formulas below, the first term represents the value of $\Delta\Phi_q$ after the swapping and the second term represents $\Delta\Phi_q$ before the swapping.

$$\begin{aligned}
s = u \quad &: \quad \Delta\Phi_q = 5\sum_{v\in V} c_v - (4d - c_u + 3\sum_{\substack{v\in V\\ v\neq u}} c_v) \\
&\qquad = 2\sum_{\substack{v\in V\\ v\neq s}} c_v + 5c'_s + c_s - 4d \\
&\qquad = 2\sum_{\substack{v\in V\\ v\neq s}} c_v + c_s - 4d \\
&\qquad \leq 2\sum_{v\in V} c_v - 4d \leq 0. \\
s' = u \quad &: \quad \Delta\Phi_q = (4d - c_u + 3\sum_{\substack{v\in V\\ v\neq u}} c_v) - 5\sum_{v\in V} c_v \\
&\qquad \leq (4d + 3\sum_{\substack{v\in V\\ v\neq s'}} c_v) - 5\sum_{\substack{v\in V\\ v\neq s'}} c_v \\
&\qquad \leq 4d + 3c'_s - 5c_s - 2\sum_{\substack{v\in V\\ v\neq s,s'}} c_v \\
&\qquad = 4d - 5c_s - 2\sum_{\substack{v\in V\\ v\neq s,s'}} c_v \leq 4d \\
s, s' \neq u \quad &: \quad \Delta\Phi_q = (4d - c_u + 3\sum_{\substack{v\in V\\ v\neq u}} c_v) \\
&\qquad - (4d - c_u + 3\sum_{\substack{v\in V\\ v\neq u}} c_v) \\
&\qquad = 3(c'_s - c_s) = -3l \leq 0.
\end{aligned}$$

Now that $\Delta\Phi_p$ and $\Delta\Phi_q$ are obtained, the addition of $\Delta\Phi_p$ to $\Delta\Phi_q$ calculates $\Delta\Phi$. Table 1 displays $\Delta\Phi$ as the sum of $\Delta\Phi_p$ and $\Delta\Phi_q$. For example, if $s = t$ and $s' = u$ then $\Delta\Phi = \Delta\Phi_p + \Delta\Phi_q \leq -6d + 4d = -2d$. Since one node cannot have p and q at the same time, t cannot be identical with u. Therefore the case when $s = t = u$ and the case when $s' = t = u$ are impossible. As Table 1 shows, $\Delta\Phi \leq -2d$ in all cases, which implies (3) holds for this case.

Table 1: $\Delta\Phi$ as the sum of $\Delta\Phi_p$ and $\Delta\Phi_q$

		$\Delta\Phi_q$		
		$s = u$	$s' = u$	$s, s' \neq u$
$\Delta\Phi_p$	$s' = t$	$-2d$	*null*	$-2d$
	$s = t$	*null*	$-2d$	$-6d$
	$s, s' \neq t$	$-6d$	$-2d$	$-6d$

(Type II): A swaps pages.
Suppose that page p is swapped from t to t' and page q is swapped from t' to t. Furthermore let s be the location of p in Concurrent-M and w be the location of q in Concurrent-M. In this case $\Delta C_{CM} = 0$ and $\Delta C_A = 2d$. So we must show that $\Delta\Phi \leq 16d$. Again we calculate $\Delta\Phi_p$ and $\Delta\Phi_q$ separately and obtain $\Delta\Phi$ after that.

First consider $\Delta\Phi_p$. There are three cases depending on whether t, t' coincide with s.

$$\begin{aligned}
t' = s \quad &: \quad \Delta\Phi_p = 5\sum_{v\in V} c_v - (4d - c_t + 3\sum_{\substack{v\in V\\ v\neq t}} c_v) \\
&\qquad = 6c_t - 4d + 2\sum_{\substack{v\in V\\ v\neq t}} c_v \\
&\qquad \leq 6\sum_{v\in V} c_v - 4d \leq 12d - 4d = 8d \\
t = s \quad &: \quad \Delta\Phi_p = (4d - c_{t'} + 3\sum_{\substack{v\in V\\ v\neq t'}} c_v) - 5\sum_{v\in V} c_v \\
&\qquad = 4d - 6c_{t'} - 2\sum_{\substack{v\in V\\ v\neq t'}} c_v \leq 4d
\end{aligned}$$

$$\begin{aligned} t,t' \neq s \quad : \quad \Delta\Phi_p &= (4d - c_{t'} + 3 \sum_{\substack{v \in V \\ v \neq t'}} c_v) \\ &\quad - (4d - c_t + 3 \sum_{\substack{v \in V \\ v \neq t}} c_v) \\ &= -4c_{t'} + 4c_t = 4(c_t - c_{t'}) \leq 8d \end{aligned}$$

As is shown above, we have obtained $\Delta\Phi_p \leq 8d$ regardless of whether t, t' coincide with s.

In the same way, we can prove $\Delta\Phi_q \leq 8d$. Thus, about the total change of Φ

$$\Delta\Phi = \Delta\Phi_p + \Delta\Phi_q \leq 16d,$$

which proves (3) holds for (Type II).

(Type III) A request is satisfied by both A and Concurrent-M.
Let v be the node at which the request is generated and f be the requested page. We denote the node which has f in Concurrent-M by s and the node which has f in A by t. There are two cases to consider depending on whether $v = s$ or $v \neq s$.

Case 1: $v = s$.
$\Delta C_{CM} = 0$. ΔC_A is at least 0. $\Delta\Phi = 0$, since Concurrent-M never changes the counter values. Thus (3) is satisfied.

Case 2: $v \neq s$.
In this case $\Delta C_{CM} = 1$ since v does not have f in Concurrent-M. And c_v^f is incremented by 1. We need to consider the following three cases.

Case (a): Suppose that $v = t$.
$\Delta C_A = 0$. So we have to show that $\Delta\Phi \leq -1$. Note that $s \neq t$. The increment of c_v^f decreases Φ by 1. If another counter is decremented, then Φ decreases by 3. Thus $\Delta\Phi \in \{-4, -1\} \leq -1$. Thus (3) holds for this case.

Case (b): Suppose that $v \neq t = s$.
$\Delta C_A = 1$. So we must show that $\Delta\Phi \leq 7$. The increment of c_v^f increases Φ by 5. If another counter is decremented, then Φ decreases by 5. Thus $\Delta\Phi \in \{0, 5\} \leq 7$.

Case (c) Suppose that $v \neq t \neq s$.
$\Delta C_A = 1$ and we must show that $\Delta\Phi \leq 7$. The increment of c_v^f increases Φ by 3. If no decrement takes place $\Delta\Phi = 3$. Else if another counter except c_t is decremented, Φ decreases by 3 and totally $\Delta\Phi = 0$. If c_t is decremented, Φ increases by 1 and totally $\Delta\Phi = 4$. Thus the claim that $\Delta\Phi \leq 7$ is proved.

Since (3) is proved for all kinds of events, the entire proof of Theorem 4 is completed. □

We can treat Concurrent-M as an on-line algorithm for uniform stars. In that case its competitive ratio grows 2 times larger than for uniform graphs.

Theorem 5 *Concurrent-M is 16-competitive for uniform stars.*

5 Furure Works

In this paper we obtained an $O(1)$-competitive deterministic on-line algorithm for uniform graphs in the constrained migration problem when $k = 1$ by examining the direct-mapped constrained migration problem. This algorithm breaks the previous best upper bound of $O(n)$-competitiveness obtained by applying the $O(kn)$-competitive constrained migration algorithm by Bartal *et al.* to the case $k = 1$. We conjecture that it is also possible to break the upper bound of $O(kn)$ by Bartal *et al.* for larger values of k. In other words, we conjecture that in the constrained migration problem there exists a $O(k)$-competitive deterministic on-line algorithm for uniform graphs.

References

[ABF93] B. Awerbuch, Y. Bartal and A. Fiat. Competitive distributed file allocation. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, pages 164-173, 1993.

[AK94] S. Albers and H. Koga. New On-line Algorithms for the Page Replication Problem. In *Proceedings of 4th Scandinavian Workshop on Algorithm Theory*, pages 25–36, 1994.

[BFR92] Y. Bartal, A. Fiat and Y. Rabani. Competitive algorithms for distributed data management. In *Proc. 24th Annual ACM Symposium on Theory of Computing*, pages 39-50, 1992.

[BS89] D.L. Black and D.D. Sleator. Competitive algorithms for replication and migration problems. Technical Report Carnegie Mellon University, CMU-CS-89-201, 1989.

[MMS88] M. Manasse, L. McGeoch and D. Sleator. Competitive algorithms for on-line problems. In *Proc. 20th Annual ACM Symposium on Theory of Computing*, pages 322-33, 1988.

[ST85] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, **28**:202-208, 1985.