TECHNICAL REPORT 94-13

Direct-mapped Constrained Migration Problem

Susanne Albers and Hisashi Koga

November 2, 1994

Department of Information Science
Faculty of Science, University of Tokyo

7–3–1 Hongo, Bunkyo-Ku Tokyo, 113 Japan

TITLE
Direct-mapped Constrained Migration Problem

AUTHORS
Susanne Albers and Hisashi Koga

KEY WORDS AND PHRASES
competitive analysis, distributed shared memory, constrained migration problem, on-line algorithm

ABSTRACT
The constrained migration problem arises in the memory management of large multiprocessor systems. Given a network of processors, each of which has its own local memory, the purpose of this problem is to find an efficient migration strategy which locates each writable page at an appropriate processor by migration to reduce the total access cost, while considering the actual capacity of each local memory. Although this problem is focused recently in the competitive analysis area for on-line algorithms, only a few results are known so far about this problem. This paper examines the on-line constrained migration problem deeper by attacking some special case of the problem which we named the *direct-mapped constrained migration problem*, where all processors are assumed to manage pages in their local memory using the same hash function.
For the direct-mapped constrained migration problem we present an optimal 3-competitive deterministic on-line algorithm when the network consists of only 2 nodes. Furthermore we develop two $O(1)$-competitive on-line algorithms for uniform graphs which are either deterministic or memoryless. Interestingly this competitive ratio breaks the previous best upper bound obtained by applying known algorithms for the constrained migration problem to this problem. Observing this fact, we also give an important conjecture on the constrained migration problem.

REPORT DATE	**WRITTEN LANGUAGE**
November 2, 1994	English
TOTAL NO. OF PAGES	**NO. OF REFERENCES**
19	14

ANY OTHER IDENTIFYING INFORMATION OF THIS REPORT

DISTRIBUTION STATEMENT
First issue 35 copies.

SUPPLEMENTARY NOTES
Part of this work was done when the second author visited Max-Planck-Institut für Informatik, Saarbrücken, Germany.

DEPARTMENT OF INFORMATION SCIENCE
Faculty of Science, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan

Direct-Mapped Constrained Migration Problem*

Susanne Albers
Max-Planck-Institut für Informatik
Saarbrücken, Germany
albers@mpi-sb.mpg.de

Hisashi Koga
The University of Tokyo
Tokyo, Japan
nwa@is.s.u-tokyo.ac.jp

November 2, 1994

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 [DF82, SD89, BS89, W91, BFR92, ABF93, CLRW93, K93, AK94, LRWY94]. Especially from the theoretical point of view, three main problems are defined. The *migration problem* [BS89, W91, ABF93, CLRW93, LRWY94] 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, K93, AK94, LRWY94] we have to determine which local memories should contain copies of the read-only page. The *allocation problem* [BFR92, ABF93, LRWY94] allows multiple copies of a writable page. This

*Part of this work was done when the second author visited Max-Planck-Institut für Informatik, Saarbrücken, Germany.

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 a 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 where the entire system is viewed as one large address space. 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.

While for the migration problem a lot of results have been obtained already, for the constrained migration problem only a few results are known. For the migration problem Black and Sleator [BS89] who initiated the study of this problem 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. For the randomized case, for general graphs Westbrook [W91] found a 3-competitive algorithm against any adaptive on-line adversary and an $(1 + \phi)$-competitive algorithm against the oblivious adversary, where $\phi \simeq 1.62$ is the golden ratio. For specific topologies like uniform and tree graphs $(2 + \frac{1}{2d})$-competitive algorithms are found against the oblivious adversary [CLRW93, LRWY94], where d is the page size factor.

On the other hand, for the constrained migration problem, the only result obtained so far is by Bartal *et al.* [BFR92]. They described an $O(kn)$-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. Besides no competitive algorithm is found for any other topology.

In this paper to shed light onto the hard constrained migration problem we define and study

[*]We can similarly define the constrained replication problem.

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 $[h(A)_{\bmod k}+1]$ of that memory. Because introducing this hashing technique eliminates the possibility of the conflict between two pages A and B such that $h(A)_{\bmod k} \neq h(B)_{\bmod k}$, what we have to deal with is only the conflicts 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 $h(A)_{\bmod k}+1 = i$. 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 are 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 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. This algorithm achieves the competitiveness of 16 against uniform stars also. In the last of this section we describe a 5-competitive randomized on-line algorithm for uniform graphs against any adaptive on-line adversary. This algorithm has the advantage of being memoryless. What is interesting about our results is that in the constrained migration problem with $k = 1$, we can find an $O(1)$-competitive deterministic on-line algorithm for uniform graphs, thus breaking 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 $k = 1$. Observing this fact, in Section 5 we give an important conjecture that there exists an $O(k)$-competitive deterministic on-line algorithm for uniform graphs in the constrained migration problem against the current best upper bound of $O(kn)$ and discuss grounds on that conjecture.

[†]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 Preliminaries

2.1 Problem Statement

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 C and D such that $h(C)_{\mathrm{mod}k} \neq h(D)_{\mathrm{mod}k}$. 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 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 $F = n$. However our analysis method can be extended to the case $F < n$ in an obvious way.

2.2 Competitive Analysis

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 be 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

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

In case A is a randomized algorithm, [BBKTW94] gives the general framework for the competitive analysis. They consider the on-line problem as the request-answer game where the adversary makes a sequence of requests served by the player (i.e. on-line algorithms). Several kinds of adversaries are introduced in [BBKTW94]. A randomized on-line algorithm A is c-competitive against the *oblivious adversary* if there exists a constant a such that for every request sequence

$$E[C_A(\sigma)] \leq c \cdot C_{OPT}(\sigma) + a,$$

where $E[C_A(\sigma)]$ denotes the expected cost incurred by A. Here the expectation is taken over the random choices made by A, see [BBKTW94].

An *adaptive on-line adversary* ADV constructs the request sequence σ on-line, making the next request based on the answer of the on-line algorithm to previous requests and must handle every request on-line using another auxiliary on-line algorithm as soon as he makes it. Since each request generated by ADV depends on the random choice of the on-line algorithm, a randomized on-line algorithm generates a probability distribution over request sequences σ. Let $C_{ADV}(\sigma)$ denote the cost incurred by the adaptive on-line adversary ADV in serving σ. A randomized on-line algorithm A is c-competitive against an adaptive on-line adversary ADV if there exists a constant a such that

$$E[C_A(\sigma)] \leq c \cdot E[C_{ADV}(\sigma)] + a.$$

Note $C_{ADV}(\sigma)$ is also a random variable because σ is constructed probabilistically according to the random choice of the on-line algorithm 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 in the next section, 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 request sequences consisting of requests to f_1 only. To process such request sequences, we can naturally think that A concentrates on the location of f_1, because the location of f_2 does not influence the cost of satisfying requests. However in order to change the location of f_1, f_2 must also be moved as the result of a swapping operation. Thus we can view this situation as the migration problem where the page size factor is $2d$. Since for the migration problem it is proved in [BS89] that no deterministic algorithm is better than 3-competitive even on a graph consisting of two nodes independently of the value of the page size factor, the lower bound of 3-competitiveness also holds for the direct-mapped constrained page migration problem. □

Next we prove the existence of some specific topology for 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 was stored in one of the nodes in V'. The number of this repetition is assumed to be finite, since if it is infinite, the trivial algorithm that swaps p to node v_1 can break A to an arbitrary extent. The next request is also generated at v_1 to the page q which has just been newly 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 end of 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 to serve 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 $(2 \leq i \leq n-1)$ and v_n, 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

$$2(n-1)(n-2)d. \tag{1}$$

The cost incurred by satisfying requests in σ' whose length is l is

$$(n-2)l, \tag{2}$$

since each request is satisfied at the cost of $n-2$. From (1) and (2), we have

$$C_A(\sigma') \geq 2(n-1)(n-2)d + (n-2)l. \tag{3}$$

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 locations of any pages throughout the phase. Note because A is deterministic, OFF can predict the behavior of A and identify the page r. 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

$$2(n-1)d. \tag{4}$$

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

$$l. \tag{5}$$

From (4)and (5) we have

$$C_{OFF}(\sigma') \leq 2(n-1)d + l. \tag{6}$$

By comparing (3) to (6), we can conclude

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

At the beginning of each phase node 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, and uniform stars. Section 4.1 deals with an optimal 3-competitive deterministic algorithm for 2 nodes. In section 4.2, we present two $O(1)$-competitive algorithms which are either deterministic or memoryless. One of the two algorithms achieves $O(1)$-competitive for uniform stars.

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 4.1: (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 $2d$ that is the cost incurred by one swapping operation. Under these notations our algorithm is described as follows.

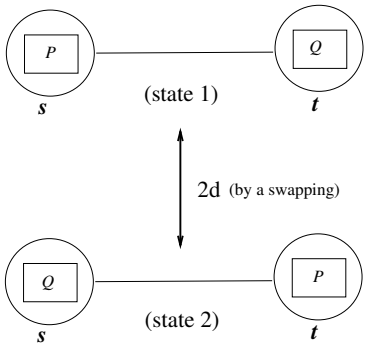

Figure 1: Two types of page configurations

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 $c_s^Q + c_t^P < 4d$, whenever 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.

It is easy to observe that two invariants below are established in algorithm TN.

1. If TN is in (state 1), then $c_s^P = c_t^Q = 0$. On the other hand, if TN is in (state 2), then $c_s^Q = c_t^P = 0$.
2. At ant time, $c_s^P + c_t^Q \leq 4d$ and $c_s^Q + c_t^P \leq 4d$.

These invariants lead to the following lemma. This lemma is crucial in the analysis.

Lemma 1 *In algorithm TN, at any time* $c_s^P + c_s^Q + c_t^P + c_t^Q \leq 4d$. *The equal sign holds just before TN is about to change its page configuration.*

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

Proof: We shall show that, for any (on-line and off-line) algorithm A, $C_{TN}(\sigma) \leq 3C_A(\sigma)$ for any request sequence σ. Our proof uses the standard technique comparing simultaneous runs of TN and A on σ by merging the actions generated by TN and A into a single sequence of events tagged to indicate the algorithm involved. This sequence contains three types of events: (Type I) TN changes its page configuration, (Type II) A changes its page configuration, and (Type III) a request is satisfied by both A and TN. We shall give a non-negative potential function Φ (initially 0) such that the following inequality holds for all kinds of events (i.e. (Type I), (Type II) and (Type III)).

$$\Delta C_{TN} + \Delta\Phi \leq 3\Delta C_A \tag{7}$$

where the Δ indicates the change in the value as a result of the event. If the potential function satisfies the above property for all events, summing up (7) for all events results in

$$C_{TN}(\sigma) + \Phi_{\text{end}} - \Phi_{\text{start}} \leq 3C_A(\sigma),$$

where Φ_{start} denotes the initial value of Φ and Φ_{end} denotes the value of Φ after TN and A finish processing the request sequence σ.

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

$$C_{TN}(\sigma) \leq 3C_A(\sigma),$$

Thus the proof ends. It remains to specify the potential and verify (7) for all events.

Let tn represent the page configuration of TN and a represent the page configuration of A. We define the potential function as:

$$\Phi \quad = \quad \begin{cases} 2(c_s^P + c_s^Q + c_t^P + c_t^Q) & \text{if } tn = a. \\ \\ 6d - (c_s^P + c_s^Q + c_t^P + c_t^Q) & \text{if } tn \neq a \end{cases}$$

Because $0 \leq c_s^P + c_s^Q + c_t^P + c_t^Q \leq 4d$, $\Phi \geq 0$ at any time. Furthermore at the beginning $\Phi = 0$ because $tn = a$ and all counters are initialized to 0. From now on we prove (7) for all kinds of events.

(Type I) The event is TN changing its page configuration. Without loss of generality suppose that TN changes its configuration from (state 1) to (state 2). In this case $\Delta C_{TN} = 2d$ and $\Delta C_A = 0$. With respect to the change of the potential function, there are two cases to consider about the page configuration of A.

Case 1: a = (state 1).
Before this event, $a = tn$ and $c_s^P + c_s^Q + c_t^P + c_t^Q = 4d$ because the equal sign of Lemma 1 holds just before TN changes its configuration. Therefore the value of Φ before the event equals to $2 \cdot 4d = 8d$. On the contrary, after the event $a \neq tn$ and $c_s^P + c_s^Q + c_t^P + c_t^Q = 0$ because c_s^Q and c_t^P are reset to 0. Therefore the value of Φ after the event equals to $6d$. Thus we have

$$\Delta C_{TN} + \Delta\Phi = 2d + (6d - 8d) = 0 = 3\Delta C_A.$$

The above equation implies (7).

Case 2: a = (state 2).
Since $a \neq tn$ before the event, the value of Φ before the event equals to $6d - 4d = 2d$. On the other hand the value of Φ after the event equals to $2 \cdot 0 = 0$, because $a = tn$ after the event. Therefore

$$\Delta C_{TN} + \Delta\Phi = 2d + (0 - 2d) = 0 = 3\Delta C_A.$$

Thus (7) holds for this case.

(Type II) The event is A changing its page configuration. Assume A changes its state from (state 1) to (state 2). In this case $\Delta C_{TN} = 0$ and $\Delta C_A = 2d$. Again there are two cases regarding the page configuration of TN.

Case 1: tn = (state 1).
Note $a = tn$ before the event and $a \neq tn$ after the event. Thus the change of the potential function is $\Delta\Phi = (6d - (c_s^P + c_s^Q + c_t^P + c_t^Q)) - 2(c_s^P + c_s^Q + c_t^P + c_t^Q) \leq 6d$. Thus, we have

$$\Delta C_{TN} + \Delta\Phi \leq 0 + 6d = 6d = 3\Delta C_A.$$

Therefore (7) is satisfied.

Case 2: tn = (state 2).
This time $a \neq tn$ before the event and $a = tn$ after the event. Thus the change of the potential function is $\Delta\Phi = 2(c_s^P + c_s^Q + c_t^P + c_t^Q) - (6d - (c_s^P + c_s^Q + c_t^P + c_t^Q)) = 3(c_s^P + c_s^Q + c_t^P + c_t^Q) - 6d \leq 12d - 6d = 6d$. Therefore we obtain

$$\Delta C_{TN} + \Delta\Phi \leq 0 + 6d = 6d = 3\Delta C_A.$$

Thus (7) is satisfied.

(Type III) The event is a request being satisfied by both algorithms. Let v be the node at which the request is generated and f be the requested page. There are two cases to consider:

Case 1: node v already had f in TN.
$\Delta C_{TN} = 0$. ΔC_A is at least 0. $\Delta\Phi = 0$, because TN performs no action. Thus (7) is satisfied.

Case 2: node v did not have f in TN yet.
In this case $\Delta C_{TN} = 1$ since TN does not have f at v. In addition c_v^f is incremented by 1 while the request is satisfied. We need to consider following two cases depending on whether algorithm A is in the same state as TN's state.

Case (A): Suppose that $a = tn$. Since A is in the same state as TN's state, A does not have page f at node v either. Therefore $\Delta C_A = 1$. The increment of c_v^f increases Φ by 2. Thus,

$$\Delta C_{TN} + \Delta\Phi = 1 + 2 = 3 = 3\Delta C_A.$$

Case (B): Suppose that $a \neq tn$. Since A has f at v, $\Delta C_A = 0$. The increment of c_v^f decreases Φ by 1. Thus,

$$\Delta C_{TN} + \Delta\Phi = 1 + (-1) = 0 = 3\Delta C_A.$$

The above equation concludes the entire proof of Theorem 3. □

4.2 Uniform Graphs

This section deals with the case when the network topology is the uniform graph. Without loss of generality, we may assume that the length of each edge equals to 1. First we will describe and analyze an 8-competitive deterministic on-line algorithm for uniform graphs. This algorithm is $O(1)$-competitive for uniform stars also. 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 decrements it. When $c_v^{f_i}$ reaches $2d$ after the increment, f_i is migrated to v and $c_v^{f_i}$ is reset to 0 if v is empty,. 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 in the analysis. Nearly the same lemma is proved in [BS89].

Lemma 2 $\forall$ *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 when one counter is incremented under the situation that 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 leads to 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 $F = n$, i.e. an algorithm necessarily has to use swapping operations to change the configuration of pages. However the analysis method below can be easily extended to the case $F < n$. Let $C_{CM}(\sigma)$ be the cost incurred by Concurrent-M to process a request sequence σ.

Just as in Section 4.1, we shall give a non-negative potential function Φ (initially 0) and show that, for any algorithm A and for any request sequence σ,

$$\Delta C_{CM} + \Delta\Phi \leq 8\Delta C_A \tag{8}$$

for all kinds of events contained in the merged sequence of actions generated in the simultaneous runs of Concurrent-M and A on σ. This yields the theorem. The events in the action sequence are classified into three types: (Type I) Concurrent-M swaps pages, (Type II) A swaps pages, and (Type III) a request is satisfied by both A and Concurrent-M.

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

$$\Phi = \sum_f \Phi_f$$

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

The initial value of Φ equals to 0, because at the beginning Concurrent-M and A have the identical configuration and all counter values are 0. Furthermore Φ is always non-negative because the counter values are bounded by 0 and $2d$.

From now on we prove (8) for all kinds of 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 this swapping, $c_{s'}^p$ is reset from $2d$ to 0 and c_s^q 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 2 and deduced two facts make the calculation of $\Delta\Phi_p$ so simple.

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

Next calculate $\Delta\Phi_q$. For clearness, we express the counter value of c_s before the swapping simply by $c_s(=l)$ and that after the swapping by $c'_s(=0)$. There are also three cases depending on whether s, s' coincide 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.

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

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 both 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.

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

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

As Table 1 shows, $\Delta\Phi \leq -2d$ in all cases, which implies (8) holds for this case.

(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.

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

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

Next consider $\Delta\Phi_q$. Since there is no distinction between p and q different from (Type I), we can calculate $\Delta\Phi_q$ in the same way as $\Delta\Phi_p$ so that we obtain $\Delta\Phi_q \leq 8d$.

Thus, about the total change of the potential function Φ

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

which shows (8) 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 (8) 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_t^f decreases Φ by 1. If another counter is decremented, then Φ decreases by 3. Thus $\Delta\Phi \in \{-4, -1\} \leq -1$ Thus (8) 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^f is decremented, Φ decreases by 3 and totally $\Delta\Phi = 0$. If c_t^f is decremented, Φ increases by 1 and totally $\Delta\Phi = 4$. Thus the claim that $\Delta\Phi \leq 7$ is proved.

Since (8) 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.*

Next we present another randomized on-line algorithm for uniform graphs. This algorithm becomes 5-competitive against adaptive on-line adversaries, which is better than Concurrent-M. Besides it has the advantage of being *memoryless*, i.e. it doesn't need any memory (for instance for counters) in order to determine when a migration or a swapping should take place.

Algorithm COINFLIP: COINFLIP processes a request at node v to page f_i as follows. If v has f_i already, the algorithm performs no action. If v does not have f_i, the algorithm serves the request by accessing the node u which has f_i. Then with probability $\frac{1}{3d}$, the algorithm migrates f_i from u to v if v is empty and moves f_i from u to v by a swapping operation if v is not empty.

Theorem 6 *The algorithm COINFLIP is 5-competitive against any adaptive on-line adversary A.*

Proof: To prove this theorem, we use the request-answer game framework of [BBKTW94]. We denote the request sequence constructed by A as $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(m)$. The game proceeds as a sequence of m rounds, such that in the tth round $\sigma(t)$ is generated by the adversary A. We show that the expected cost incurred by COINFLIP on σ is no more than 5 times the cost incurred by the adversary A on σ, no matter how A makes $\sigma(t)$ at time t, knowing the answer of COINFLIP to previous requests. This yields the theorem. Hereafter let $C_{CF}(\sigma)$ be the cost incurred when COINFLIP serves σ.

Concretely again we compare simultaneous runs of COINFLIP and A on σ by merging the actions generated by COINFLIP and A into a single sequence of events. This sequence contains two types of events: (Type I) A swaps some page p and some other page q, and (Type II) both COINFLIP and A respond to the request. (Type II) may accompany a swapping by COINFLIP.

We shall give a non-negative potential function Φ (initially 0) such that the following inequality holds for all kinds of events.

$$E[\Delta C_{CF}] + E[\Delta\Phi] \leq 5\Delta C_A. \tag{9}$$

The expectation of the potential change is taken over the random choice made by the algorithm COINFLIP. Summing this inequality for all events results in

$$E[C_{CF}(\sigma)] + E[\Phi_{\text{end}}] - E[\Phi_{\text{start}}] \leq 5C_A(\sigma).$$

From the definition of the potential function, we have

$$E[C_{CF}(\sigma)] \leq 5C_A(\sigma).$$

Thus the proof ends. It remains to specify the potential function and verify (9) for all kinds of events. Let S be the set of nodes where COINFLIP and A store different pages. The potential function Φ is defined as:

$$\Phi = 5d \cdot |S|.$$

Obviously initially $\Phi = 0$ and always $\Phi \geq 0$.

From now on, we prove (9) for all kinds for events.

(Type I) The event is A swapping some page p and some other page q. Let u be the node which has p before the swapping and v be the node which has q before the swapping in algorithm A. In this case, $\Delta C_A = 2d$ and $\Delta C_{CF} = 0$, so we must show $E[\Delta\Phi] \leq 10d$. $\Delta|S|$ is at most 2 since in algorithm A all nodes except node u and node v does not change the page in their local memory, while COINFLIP does not change its page configuration. Therefore $\Delta\Phi = 5d \cdot \Delta|S| \leq 10d$. Thus (9) holds.

(Type II) The event is a request being satisfied by both algorithms. Let v be the node at which the request is generated and f be the requested page. There are two cases to consider:

Case 1: node v already had f in COINFLIP.
$\Delta C_{CF} = 0$. ΔC_A is at least 0. $E[\Delta\Phi] = 0$, because there is no swapping in COINFLIP. Thus (9) is satisfied.

Case 2: node v did not have f in COINFLIP yet.
Let u be the node which holds f in algorithm COINFLIP. Since COINFLIP satisfies the request at node v by accessing u, it costs 1 for COINFLIP to respond to the request and with the probability $\frac{1}{3d}$, the swapping takes place. Thus $E[\Delta C_{CF}] = 1 + \frac{1}{3d} \cdot 2d = \frac{5}{3}$. We need to consider two cases depending on whether A has page f at node v or not.

Case (a): A has page f at node v already.
In this case $\Delta C_A = 0$. To obtain $\Delta\Phi$ we investigate $\Delta|S|$ instead. If the swapping is not done in COINFLIP, $\Delta|S| = 0$. Next consider the case COINFLIP makes the swapping. Note both of $u, v \in S$ before the swapping. After the swapping, $v \notin S$ since both of A and CF have f at v. Node u does not contribute to the increase of $|S|$, since u belongs to S before the swapping. Thus we have $\Delta|S| \leq -1$ when COINFLIP makes the swapping. The expected change of $|S|$ is $E[\Delta|S|] \leq \frac{1}{3d}(-1) = -\frac{1}{3d}$. Therefore we obtain

$$E[\Delta C_{CF}] + E[\Delta\Phi] = E[\Delta C_{CF}] + 5d \cdot E[\Delta|S|] \leq \frac{5}{3} - \frac{5}{3} = 0 = 5\Delta C_A.$$

The above equation implies (9).

Case (b): A does not have page f at node v.
In this case $\Delta C_A = 1$. Regarding the change of $|S|$, $\Delta|S| = 0$ if COINFLIP makes no swapping.

Otherwise $\Delta|S| \leq 2$, since A does not change its page configuration, while in COINFLIP only node u and node v change the page in their local memory. Thus, $E[\Delta|S|] \leq \frac{1}{3d} \cdot 2 = \frac{2}{3d}$. Therefore

$$E[\Delta C_{CF}] + E[\Delta\Phi] = E[\Delta C_{CF}] + 5d \cdot E[\Delta|S|] \leq \frac{5}{3} + 5d \cdot \frac{2}{3d} = 5 = 5\Delta C_A.$$

Thus (9) holds for this case.

Since we have proved (9) for all types of events, we conclude the proof of Theorem 6. □

Remark: To analyze COINFLIP when $F < n$, we need to replace the definition of the potential function Φ as $\Phi = 5d \cdot (|S| + |T|)$, where T is the set of nodes which is empty only in CM's configuration. The case analyses need to be finer than in the above proof.

5 Conjecture on the Constrained Migration Problem

In the previous section, we find an $O(1)$-competitive deterministic on-line algorithm for uniform graphs in the direct-mapped constrained migration problem. Since this problem is thought of as the constrained migration problem with $k = 1$, we've obtained an $O(1)$-competitive deterministic on-line algorithm in the constrained migration problem when $k = 1$. Observing this fact, this section presents an important conjecture about the constrained migration problem for uniform graphs and mention two grounds on that.

First we review the only result for the constrained migration problem by Bartal *et al.* [BFR92]. They presented an $O(kn)$-competitive deterministic on-line algorithm for uniform graphs. Therefore their algorithm grows $O(n)$-competitive for uniform graphs when $k = 1$. However, in this paper we could break this upper bound of $O(n)$-competitiveness by finding an $O(1)$-competitive deterministic on-line algorithm for uniform graphs when $k = 1$. We conjecture that it is also possible to break the upper bound of $O(kn)$-competitiveness by Bartal *et al.* for larger values of k. That is,

Conjecture 1 *In the constrained migration problem there exists an $O(k)$-competitive deterministic on-line algorithm for uniform graphs.*

From now on, we mention two grounds on this conjecture.

Ground 1: The difference between the constrained migration problem and the direct-mapped constrained migration problem is as follows: When a node v is not empty and some page needs to be brought into v, in the constrained migration problem an algorithm must choose which page to drop among k pages currently stored in v, whereas in the direct-mapped constrained migration problem an algorithm does not have to make this choice since there is only one candidate page in v. This operation of choosing which page to drop resembles to the famous *paging problem* [ST85, RS90].

> **Paging problem:** Consider a two-level memory system. There is a fast memory, a cache that can store k pages and a slow memory that can potentially store an infinite number of pages. We have a sequence of requests, each of which specifies a page.

A request is served if the corresponding page is in the cache. If the requested page is not in the cache, a *page fault* occurs. Then one of the k pages in cache must be evicted and dropped to the slow memory so as to make room for the incoming page. A paging algorithm specifies on-line which page to evict on a page fault so that the number of page faults may be reduced.

Thus the constrained migration problem is viewed as the combination of the direct-mapped constrained migration problem and the paging problem. In fact using the technique for paging problem [ST85, RS90], one can easily obtain the lower bound for the constrained migration problem.

Theorem 7 *No deterministic on-line algorithm for the constrained migration problem is better than k-competitive even on a graph consisting of two nodes.*

Since in the paging problem some deterministic on-line algorithms, for example LRU and FIFO, are $\Theta(k)$-competitive and in the direct-mapped constrained migration problem there is a $\Theta(1)$-competitive deterministic on-line algorithm for uniform graphs, it seems to be reasonable that there is a deterministic on-line algorithm that is $O(k)$-competitive for uniform graphs in the constrained migration problem.

Table 2: The competitiveness for various data management problems

	allocation problem	migration problem
General constrained version	$\Theta(kn)$	$O(kn)$
Direct-mapped version	$\Theta(n)$	$\Theta(1)^*$

* this paper

Ground 2: In the constrained *allocation* problem, Bartal *et al.* gave an $O(kn)$-competitive deterministic on-line algorithm for uniform graphs. This algorithm can be applied to the constrained migration problem without increasing its competitive ratio. On the other hand, they showed that no deterministic on-line algorithm is better than $\Omega(kn)$-competitive for uniform graphs in the constrained allocation problem, though this lower bound technique cannot be applied to the constrained migration problem. Then what would be the competitiveness if we consider the direct-mapped constrained allocation problem defined in the same way as the direct-mapped constrained migration problem? The answer for this question is that the results by Bartal *et al.* remain valid even if we restrict the value of k to 1. In other words, an $O(n)$-competitive deterministic on-line algorithm and the corresponding tight lower bound exist for uniform graphs. See Table 2. Thus the competitiveness for the constrained allocation problem is k times that of the direct-mapped version. We infer that this relation still holds between the constrained migration problem and the direct-mapped constrained migration problem. Since the competitiveness for the direct-mapped constrained migration problem for uniform graphs is $\Theta(1)$, it seems to be true that the competitiveness for the constrained migration problem for uniform graphs is $O(k)$.

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.

[BBKTW94] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, special issue on on-line algorithms, **11**:2-14, 1994. A preliminary version appeared at the *22nd Annual ACM Symposium on Theory of Computing, 1990*.

[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.

[CLRW93] M. Chrobak, L.L. Larmore, N. Reingold and J. Westbrook. Page migration algorithms using work functions. In *Proc. 4th International Annual Symposium on Algorithms and Computation*, pages 406-415, 1993.

[DF82] D. Downey and D. Foster. Comparative models of the file assignment problem. *Computing Surveys*, **14**(2), 1982

[K93] H. Koga. Randomized on-line algorithms for the page replication problem. In *Proc. 4th International Annual Symposium on Algorithms and Computation*, pages 436-445, 1993.

[LRWY94] C. Lund, N. Reingold, J. Westbrook and D. Yan. On-line distributed data management. To appear in *Proc. 2nd Annual European Symposium on Algorithms*, 1994.

[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.

[RS90] P. Raghavan and M. Snir. Memory versus randomization in on-line algorithms. Research Report IBM T.J. Watson Research Center, RC 15622, 1990.

[SD89] C. Scheurich and M. Dubois. Dynamic page migration in multiprocessors with distributed global memory. *IEEE Transactions on Computers*, **38**(8):1154-1163, 1989.

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

[W91] J. Westbrook. Randomized Algorithms for the multiprocessor page migration. In *Proc of the DIMACS Workshop on On-Line Algorithms*, pages 135-149, 1992