1-1-2004

The randomized server problem

Qin Zhang
University of Nevada, Las Vegas

THE RANDOMIZED SERVER PROBLEM

by

Qin Zhang

Bachelor of Science
Mathematics Department, Shanghai Teacher's University, P. R. China
1992

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree in Computer Science
School of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
May 2004

UNLV
UNIVERSITY OF NEVADA LAS VEGAS

Thesis Approval
The Graduate College
University of Nevada, Las Vegas

March 26, 2004

The Thesis prepared by

QIN ZHANG

Entitled

The Randomized Server Problem

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative

ABSTRACT

The Randomized Server Problem

by

Qin Zhang

Dr. Wolfgang W. Bein, Examination Committee Chair
Professor of Computer Science
University of Nevada, Las Vegas

In the k-server problem there are $k \geq 2$ identical servers which are located at k points in a metric space M. If there is a request to a point $r \in M$, one of the servers must be moved to the request point in order to "serve" this request. The cost of this service is the distance between the points where the server "resided" before the service and after the service. A k-server algorithm A must decide which server should be moved at each step. The goal of A is to minimize the total service cost. Competitiveness makes sense as a concept when A lacks timely access to all input data. We consider the version of the problem where requests must be served "online", *i.e.*, the algorithm must decide which server to move without knowledge of future requests. Randomization is a strong tool to derive algorithms with better competitiveness .

The main contributions of this thesis are:

- An explicit detailed proof of the 2-competitiveness of the Random Slack Algorithm, which has never been given before. We note that Random Slack is a trackless algorithm.

- An essay-style description of a new concept called the *knowledge state* approach, which has recently been developed by Bein, Larmore, and Reischuk.

- We give optimally competitive randomized algorithms for 2 and 3 cache paging with few bookmarks. We note that the paging problem is a special case of the server problem, and that it is desirable to minimize the number of bookmarks, as such bookmarks pose a considerable challenge in real world applications such as cache management of pages on the world wide web.

Furthermore, the thesis summarizes a number of basic results for both the randomized and the deterministic server problem.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank Dr. Wolfgang W. Bein for chairing my committee and advising my thesis work. He is such a good professor and I feel deeply indebted to him for his kindness and being generous with his time whenever I needed it.

Special thanks go to his 'TA', Doina Bein for her patience and being generous with her time.

I would like to thank Dr. Lawrence L. Larmore for his kindness and helping me with my thesis.

I would also like to thank Dr. Lawrence L. Larmore, Dr. Kazem Taghva, and Dr. Shashi Nambisan for being my committee members.

I would like to thank all of the faculty members of the School of Computer Science of the University of Nevada, Las Vegas for my formal education along with generous financial support.

I am grateful to my parents, my sister's family for taking care of my baby while I immersed myself in the Master's degree program, and for their inspiration and encouragement. Finally and most importantly, I thank my husband, Shimin Luo, for his love and support.

CHAPTER 1

INTRODUCTION

1.1 k-Server Problem

Suppose you are going to ski, and this is the first time you ski in your life. You have to decide whether to rent a pair of skis or to buy. Say, renting costs $10, and buying costs $100. The problem is you do not know if you will enjoy skiing after this first try. In other words, how many times you will go skiing altogether? If you knew you would go skiing at least 10 times, then, of course, you should buy, otherwise, you should rent. This is an online problem since we are dealing with a decision without knowing clearly what will happen in the future. If we tackle this situation we are essentially using some algorithm. Such an algorithm is called an online algorithm. In fact, the above ski rental problem is really a special case of a well studied problem in online algorithms, the server problem. We will now begin with formal definitions to make this clear.

Definition 1 (Metric Space) *A pair (S,d) is called a Metric Space where S is a set of points and d is a metric distance function:*

$$d : S \times S \rightarrow R^{+}$$

which satisfies:

1. $d(i,j) > 0, \forall i \neq j, i,j \in S;$

2. $d(i,i) = 0, \forall i \in S;$

3. $d(i,j) + d(j,k) \geq d(i,k), \forall i,j,k \in S;$ *(triangle inequality)*

4. $d(i,j) = d(j,i), \forall i,j \in S.$ *(symmetry)*

Definition 2 (The k-Server Problem) *There are $k \geq 2$ servers at each step which are located at k points in a metric space M. At each step, there is a request which is actually a point $r \in M$. One of the servers must be moved to r in order to serve this request. The cost of this service is the distance between the points where the server resided before the service and after the service.*

A k-server algorithm A must decide which server should be moved at each step. The goal of A is to minimize the total service cost. The following figures explain how the ski-rental problem reduces to the 2-server problem.

We reduce the ski-rental problem to the 2-server problem in the following metric space:

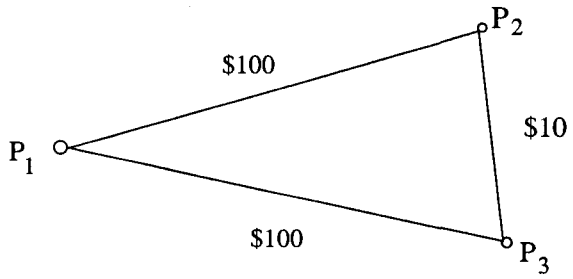

Figure 1.1: The Ski-Rental Problem: a 2-server problem.

This problem has 3 states (or configurations) $\{P_1, P_2\}$, $\{P_1, P_3\}$ and $\{P_2, P_3\}$ as illustrated by Figure 1.2.

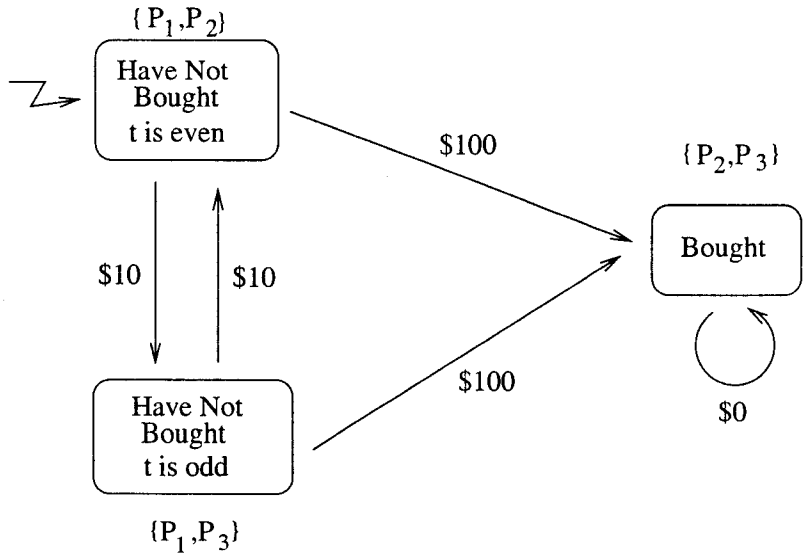

Figure 1.2: The Ski-Rental Problem - three states.

Figure 1.2 assumes that there have been t requests. The start configuration is $\{P_1, P_2\}$. For example, When we are at $\{P_1, P_2\}$, if P_3 is requested, we can either go to configuration $\{P_1, P_3\}$ (which means we decide to rent skis), or to configuration $\{P_2, P_3\}$ (which means we decide to buy skis).

1.2 The Competitive Ratio and Competitiveness

Competitiveness also called the *Competitive Ratio* has been a standard measuring stick for online algorithms. Computer scientists use it to evaluate the performance of online algorithms. We call this evaluation "competitive analysis". This analysis compares the performance of the online algorithm against the performance of the optimal offline algorithm on every request sequence and considers the worst-case ratio.

Definition 3 (c-Competitive) *Let $cost_A(\rho)$ represent the cost incurred by an online algorithm A for the request sequence ρ. Let $cost_{opt}(\rho)$ represent the cost incurred by the optimal offline algorithm for the request sequence ρ.*

An online algorithm A is c-competitive if there exists a constant k such that for any finite request sequence ρ,

$$cost_A(\rho) \leq c \cdot cost_{opt}(\rho) + k$$

where k must be independent of the request sequence ρ.

We also say that A attains a *competitive ratio* c, if A is c-competitive. The *competitive ratio* of the algorithm A, which we write c_A is the infimum over c such that A is c-competitive.

Definition 4 (Competitiveness of an Online Problem) *The competitiveness of an online problem P is $inf\{competitiveness\ of\ A | A\ is\ an\ online\ algorithm\ for\ P\}$.*

For example, the above *ski rental problem* is 2-competitive. Because there exists an online algorithm A and a constant k (which is 0), such that for any finite request sequence ρ,

$$cost_A(\rho) \leq 2 \cdot cost_{opt}(\rho)$$

CHAPTER 2

DETERMINISTIC ALGORITHMS

In this chapter we survey a number of known deterministic algorithms for the server problem. Along with this, we also give some background on the problem.

2.1 BALANCE

BALANCE is an example of k-server algorithm. BALANCE is optimal for metric spaces consisting of $k+1$ points. The idea of this algorithm is trying to keep the total distance traversed by the k servers roughly the same.

BALANCE is simply defined as follows. Let D_i represent the cumulative distance traversed by the server at point i. Let d_i represent the distance from the server at point i to the next request. The server with the minimum value of $D_i + d_i$ will be sent to next request.

We note that a more simplistic local greedy algorithm, which always sends the closest server, is not competitve, as it is possible that the online algorithm may ping-pong a server a between two points which are alternately requested. BALANCE will not have this problem, because when server a ping-pongs for a total cost greater than the cost of moving b to the request, BALANCE will move b. Moreover, we can prove that BALANCE reaches the lower bound of the competitiveness of k-server problem for metric spaces with $k+1$ points.

Theorem 1 (Theorem On BALANCE) *[MMS90] Let C_{bal} be the competitive ratio of*

BALANCE on a metric space with $k+1$ points. Then, $C_{bal} \leq k$.

Proof: We use S_i to denote the configuration which has a server on every point except i as illustrated by Figure 2.1. We use $opt_t(S_i)$ to denote the optimal way of serving the first t requests and ending up in configuration S_i. We use h^t to represent BALANCE's hole (i.e., the one point where BALANCE does not have a server) after t requests.

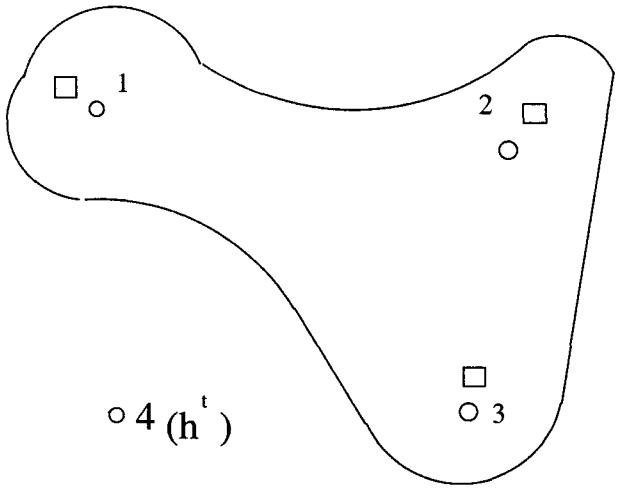

Figure 2.1: S_4 configuration

We use D_i^t to represent the distance traversed after the first t requests by BALANCE's server which occupies point i as illustrated by Figure 2.2, Figure 2.3 and Figure 2.4.

Figure 2.2 illustrates D_i^0, when $t = 0$.

Figure 2.3 illustrates D_i^1, when $t = 1$.

Figure 2.4 illustrates D_i^2, when $t = 2$.

Because we have k servers for metric spaces with $k+1$ points, there is only one point which is not occupied by any server which we call the *hole*. Without loss of generality, a worst request sequence ρ always requests the hole, since a request to another point will cost BALANCE nothing and could possibly increase the cost of OPT. When $t+1^{th}$ request

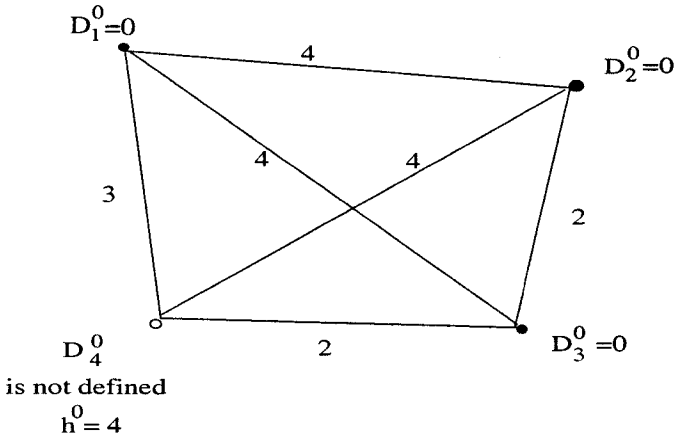

Figure 2.2: D_i^0 (when $t = 0$)

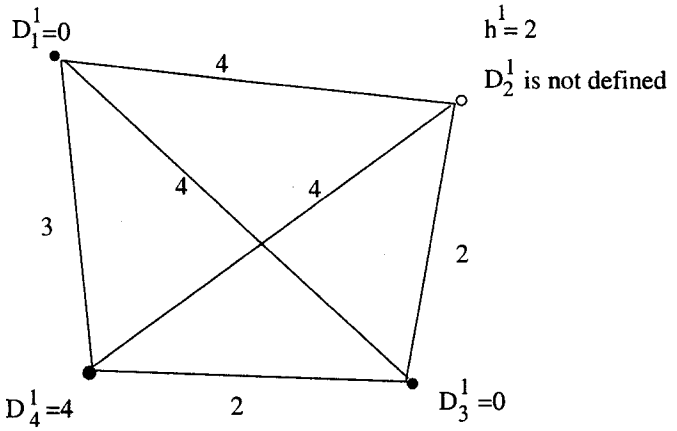

Figure 2.3: D_i^1 (after $t = 1$)

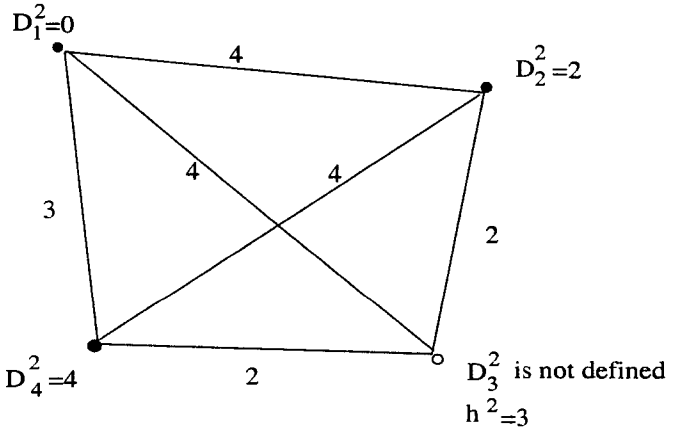

Figure 2.4: D_i^2 (after $t = 2$)

comes, it is to point h^t. We claim the following:

Claim 1 (Claim on BALANCE) *For all* $i \neq h^t, D_i^t \leq opt_t(S_i)$.

We can prove our theorem using Claim 1 as follows: the total work done by BALANCE till time t is $\Sigma_{i \neq h^t} D_i^t$. By this claim, we get:

$$\Sigma_{i \neq h^t} D_i^t \leq \Sigma_{i \neq h^t} opt_t(S_i) \tag{2.1}$$

We observe that the cost of moving from S_i to S_j is $d_{i,j}$, so we have:

$$opt_t(S_i) \leq opt_t(S_j) + d_{i,j}$$

For clarity, let us expand Inequality 2.1 and suppose $opt_t(S_*)$ is the minimal one for all i.

When,

$i = 1, D_1^t \leq opt_t(S_1) \leq opt_t(S_*) + d_{1,*}$

$i = 2, D_1^t \leq opt_t(S_2) \leq opt_t(S_*) + d_{2,*}$

:

:

$\Sigma_{i \neq h^t} D_i^t \leq \Sigma_{i \neq h^t} opt_t(S_i) \leq k opt_t(S_*) + \Sigma_{i \neq h^t} d_{i,*}$

$\leq k[opt_t(S_*)] + k \max d_{i,*}$

$\leq k[COST_{OPT}] + K$

(Where $K = k \cdot \max_{i,j} \{d_{i,j}\}$ which has no relationship with the length of the sequence)

□

We now prove the claim by induction on t:

Proof:

When $t = 0$, the claim is obviously true;

Suppose the claim is true on t, we get:

When $i \neq h^t$,

$$opt_t(S_i) \geq D_i^t \tag{2.2}$$

Now we need to prove:

$$opt_{t+1} S_{h^t} \geq D_{h_t}^{t+1} \tag{2.3}$$

At time $t+1$, there is a request to h^t. Let $i \neq h^t$ be the point which minimizes $D_i^t + d_{i,h^t}$. The server at i will go to h^t. So, the point i becomes the hole.

Therefore,

$$D_{h^t}^{t+1} \leftarrow D_i^t + d_{i,h^t}$$

Meanwhile,

$$opt_{t+1}(S_{h^t}) \leftarrow \min_{i \neq h^t} opt_t(S_i) + d_{i,h^t}$$

Because we have Inequality 2.2, we get Inequality 2.3.

If $j \neq h^t$, $j \neq i$

$$opt_{t+1}(S_j) = opt_t(S_j) \geq D_j^t = D_j^{t+1}$$

□

If there are more than $k+1$ points in the metric space, BALANCE does not work very well. Even when $k = 2$, BALANCE is not competitive.

There is a simple variant of BALANCE (see Irani and Rubinfeld in [IR91]) which minimizes $D_i + 2d_i$ and is 10-competitive for $k = 2$. But Chrobak and Larmore show in [CL91] that this variant is no better than 6-competitive. Kleinberg shows in [Kle94] that when $k = 2$, any sort of balance algorithm which sends the server which minimizes $D_i + f(d_i)$ can be no better than 3.82-competitive, where f can be any function.

Early in 1988, Manasse, McGeoch and Sleator [MMS90] posed *the k-server problem.* They proved that the lower bound on the competitive ratio on this problem is k, and that for the special cases when $k = 2$ and $n = k+1$ the upper bound is also k. The famous (and notorious) *k-server conjecture* is that there is a k-competitive deterministic online algorithm for any metric space.

But for years whether there is any competitive algorithm at all (the competitive ratio of which could be bounded by any function of k on all metric spaces) was unknown. In 1990 in one of the earlier results in the field, Fiat, Rabani and Ravid [FRR90] proved that one algorithm has competitive ratio of $O(k^k)$.

Koutsoupias and Papadimitriou [KP94b] proved in 1994 that WFA, an algorithm which had long been conjectured [CL92b] to be k-competitive, is $(2k-1)$-competitive. However, WFA is still conjectured to be k-competitive. This is the best known result for the k-server problem. It is based on *work functions.*

2.2 Work Functions

In this section, we will introduce *work functions* and the *work function algorithm.* We will use the following notation:

- A configuration S is defined to be a multiset of k points in the metric space.
- Let S_k be the set of all configurations.
- Let $d(X, Y)$ be the minimum matching distance between X and Y, where $X \in S_k$, and $Y \in S_k$.
- Let ρ be the request sequence, and ρ_t be the t^{th} request in ρ.
- Let $opt_t(\rho, X)$ be the minimum cost of serving the first t requests in ρ, ending up in state X.
- Let X_0 be the starting configuration.

We use dynamic programming to compute $opt_t(\rho, X)$ as follows:

$$opt_0(\rho, X) = d(X, X_0)$$

If $\rho_t \in X$,

$$opt_t(\rho, X) = opt_{t-1}(\rho, X)$$

If $\rho_t \notin X$,

$$opt_t(\rho, X) = min_{Y|\rho_t \in Y}[opt_t(\rho, Y) + d(X, Y)]$$

We can combine these two cases into one:

$$opt_t(\rho, X) = min_{Y|\rho_t \in Y}[opt_{t-1}(\rho, Y) + d(X, Y)]$$

$w_t = opt_t(\rho, \cdot)$ is called *work function.*

The work function has the following properties:

1. $\forall X, Y \in S_k, opt_t(\rho, X) \leq opt_t(\rho, Y) + d(X, Y)$

2. $opt_t(\rho, X) = min_{x \in X}[opt_t(\rho, X - x + \rho_t) + d(x, \rho_t)]$

3. $opt_t(\rho, X) \geq opt_{t-1}(\rho, X)$

We are now ready to describe the work function algorithm (WFA). After $t-1$ requests, the algorithm is at a configuration which we call X_{t-1}. The next request is ρ_t. WFA serves the request by moving to that configuration X which minimizes

$$opt_t(\rho, X) + d(X, X_{t-1}) \tag{2.4}$$

The so-called *retrospective algorithm* minimizes the first term of 2.4. It is not competitive by itself.

The so-called *local greedy algorithm* minimizes the second term of 2.4. It is not competitive, either.

By combining these two algorithms into one, WFA achieves the best known competitiveness for the general k-server problem in a metric space which consists of any number of points, with the competitive ratio $2k - 1$. This result is given by Koutsoupias and Papdimitriou [KP94a] in 1994.

2.3 Lower Bounds

Definition 5 (Lower Bound) *A problem has lower bound of D on competitiveness if the following hold:*

If $\forall A, \exists \rho, cost_A(\rho) \geq D cost_{opt}(\rho)$

To prove lower bound for the best possible competitive ratio an online algorithm can achieve on a specific problem, the standard technique is to employ a cruel adversary. The cruel adversary's goal is to find a request sequence to force the cost of the online algorithm to be high, while the optimal cost is low.

Although the server problem is in *P-Time*, other online problems that have been studied might be NP-complete. How to determine the optimal offline cost for those problems may be very difficult. However, we can give the upper bound for the optimal cost in proving a lower bound on the competitive ratio. To average over a collection of m specific algorithms can be used to bound the optimal cost above. That is, for any request sequence ρ constructed by the cruel adversary, the sum of the costs of these algorithms is at most b times the cost of A on ρ , i.e.,

$$\Sigma_{1 \leq i \leq m} A_i[\rho] \leq bA[\rho]$$

There exists an algorithm in the collection whose cost is at most the average. This implies a lower bound of m/b on the competitive ratio. We can use this technique to prove the lower bound for an online algorithm for the k-server problem in any metric space.

Theorem 2 (Theorem on k-server) *[MMS90] Suppose A is any online algorithm for the k-server problem. Then there is a lower bound on the competitiveness of A of k.*

Proof: Since we need to use a specific request sequence to prove the lower bound. The adversary can choose $k+1$ points inside of which the k points are initially occupied by the k servers. Let A be any deterministic online algorithm for the k-server problem. The adversary will always requests the point where A does not have a server. The cost of A is

$$cost_A(\rho) = \Sigma_{j=1}^{N-1} d_{p_j, p_{j+1}}$$

In order to upper bound the cost of OPT, we give k specific algorithms A_1,..., A_k whose total cost on the request sequence ρ is equal to A's cost. The k algorithms are as follows:

On the first request, each of the k algorithms will move a different server to the request. Then, the k algorithms maintain the invariant that each algorithm has a server on the point which was last requested. But, the vacant point of any two of the k algorithms are different. Therefore, on every request, exactly one of the A_i must move its server. The invariants of the k algorithms are maintained. The total cost of these k algorithms is:

$$\Sigma_{i=1}^{k} cost_{A_i}(\rho) = \Sigma_{j=1}^{k} d_{p_j,p_{k+1}} + \Sigma_{j=1}^{N-1} d_{p_{i_j},p_{i_{j+1}}} = \Sigma_{j=1}^{k} d_{p_j,p_{k+1}} + cost_A(\rho)$$

The $\Sigma_{j=1}^{k} d_{p_j,p_{k+1}}$ is constant, which accounts for the first move of each k algorithms.

Since we have

$$k cost_{opt}(\rho) \leq k \min_i cost_{A_i}(\rho) \leq \Sigma_{i=1}^{k} cost_{A_i}(\rho)$$

we get that

$$k cost_{opt}(\rho) \leq cost_A(\rho) + K$$

The constant K can be neglected, because the request sequence can be arbitrarily costly, and hence the $cost_A(\rho)$ can be arbitrarily large.

□

CHAPTER 3

RANDOMIZED ALGORITHMS

3.1 Randomized Algorithms

In order to achieve better competitiveness, some researchers have introduced *randomization* into the area of online algorithms. A *randomized online algorithm* is an online algorithm which may make choices using randomization when responding to requests. The cost of such an algorithm is a random variable since the decision at each step is made randomly. Therefore, its performance is measured using its expected cost for a request sequence.

Online algorithms can be viewed as competing against an adversary. When we deal with randomized algorithms the choice of adversary becomes improtant.

3.1.1 Adversaries for Randomized Algorithms

In fact, in the study of *randomized algorithms*, we consider more than one kind of the adversary. They are the *strong adversary*, the *oblivious adversary*, and the *adaptive online adversary*. They are distinguished by how they know the outcomes of random choices made by the algorithms and how they themselves serve the request sequence. Of course, every adversary knows the code of the online algorithm.

- The strong adversary knows the past and knows the choice of A at every step. In fact, against the strong adversary, randomization does not help.

- The oblivious adversary knows nothing except for the code.
- The adaptive online adversary knows the past choices of the algorithm, but not the future choices. The adversary must make its own choice at each step before it knows the algorithm's choice at that step. (After the algorithm moves, it will know the move).

The most powerful adversary is the *strong adversary*, since it actually knows everything. The least powerful adversary is the *oblivious adversary*, because it knows the least.

A number of other adversaries have been used in the literature, but these are the most important.

Definition 6 (c-Competitive) *A randomized online algorithm A is* c-competitive *(against the optimal offline algorithm) if* $\exists$ *a constant* K, $\forall$ *request sequence* ρ, *such that:*

$$Ecost_A(\rho) \leq c \cdot cost_{opt}(\rho) + K.$$

E here means the expected value. If more information about the adversary is given, c increases such as:

$$c_{obl} \leq c_{adon} \leq c_{strong}$$

Where c_{obl} stands for the *Competitiveness against the Oblivious Adversary*; c_{adon} stands for the *Competitiveness against the Adaptive Online Adversary*; c_{strong} stands for the *Competitiveness against the Strong Adversary.* Note that c_{obl} is equal to c in the above inequality.

3.1.2 Models of Randomized Online Algorithms

We introduce four models of randomized online algorithms in this thesis as follows:

- Distribution of deterministic online algorithms.

- Behavioral model.
- Distributional model.
- Mixed model.

Distribution of Deterministic Online Algorithms. $\mathcal{A}$ here is a random variable whose value is a deterministic online algorithm. $\mathcal{A}$ is *barely random* if the random variable has a finite distribution.

Behavioral Online Algorithms. $\mathcal{A}$ here selects the next configuration using randomization to serve the request.

Distributional Online Algorithms. At each step, $\mathcal{A}$ chooses a distribution on configurations.

A distributional online algorithm is in fact deterministic, because the distributions are computed deterministically.

There is a theorem regarding the above three models.

Theorem 3 *All the above three models are equivalent. Because if $\mathcal{A}$ is an algorithm of one of the models, there exist two algorithms each of which belongs to the two other models respectively, such that, given any request sequence ρ, the expected cost of each of the two other algorithms for ρ is no greater than the expected cost of $\mathcal{A}$.*

The Mixed Model. The mixed model is a generalization of the behavioral model and the distributional model. It selects a distribution of its possible configurations at every step, but, unlike a distributional algorithm, which must make the selection deterministically, can use randomization.

Please note that the mixed model for randomized online algorithms is equivalent to the other three models.

Here note that, the behavior of the *oblivious adversary* without loss of generality is deterministic, therefore, the request sequence ρ it generates as well as its $cost_{obl}(\rho)$ is not a random variable.

The rest of the thesis will focus on *randomized server algorithms* only.

3.2 Random Slack - a Randomized 2-Server Online Algorithm

In this chapter, we introduce a randomized 2-server online algorithm, Random Slack. Random Slack is 2-competitive against the adaptive online adversary, and therefore also against the oblivious adversary. The reason we introduce it here is because this is a simple algorithm. An explicit proof which has never been given by anyone else will be given here.

In Figure 3.1, we define *random slack.*

Definition 7 (Random Slack) *Given servers at $s_1, s_2 \in M$, and a request $r \in M$. Define x, y, z as follows:*

$$x = 0.5(|s_1r| + |s_1s_2| - |s_2r|)$$

$$y = 0.5(|s_2r| + |s_1s_2| - |s_1r|)$$

$$z = 0.5(|s_1r| + |s_2r| - |s_1s_2|)$$

Note that x, y, z is the unique solution to the following system, and that $x, y, z \geq 0$

$$x + z = |s_1r|$$

$$y + z = |s_2r|$$

$$x + y = |s_1s_2|$$

We call the Y-shaped figure defined in this way, and shown in Figure 3.1 the *abstract convex hull* of s_1, s_2, r.

Note that the point in the center of Figure 3.1 is a virtual point, meaning that there may be no such point in M.

The probability of moving s_1 to r is $y/|s_1s_2|$; and of moving s_2 to r is $x/|s_1s_2|$.

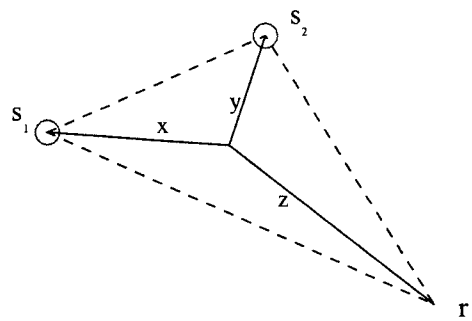

Figure 3.1: The "Y" with virtual point shown

3.2.1 Behavioral Algorithms - Proof Using Potential

Theorem 4 (Theorem on Slack) *Random Slack is 2-competitive.*

Proof: Random Slack is a behavioral algorithm. To prove this theorem, we define a behavioral potential Φ here to be:

$$\Phi = |s_1s_2| + 2|s_2a_2| = \alpha + 2\beta$$

We are to prove that:

$$E(cost_{RS}) + E(\Delta\Phi) \leq 2cost_{adv} \tag{3.1}$$

Figure 3.2 illustrates that we have three different abstract convex hulls for this algorithm depending on the position of the request r: abstract convex hull (a)(i.e., Figure 3.3),

abstract convex hull (b)(i.e., Figure 3.9) and abstract convex hull (c)(i.e., Figure 3.6).

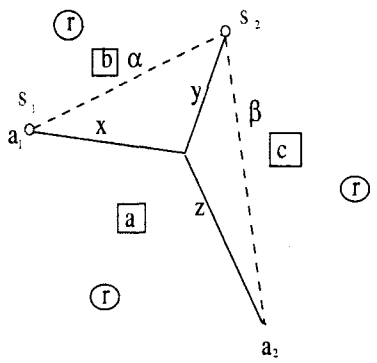

Figure 3.2: Slack: 3 Abstract Convex Hulls

Let's first look at the abstract convex hull(a) as illustrated in Figure 3.3.

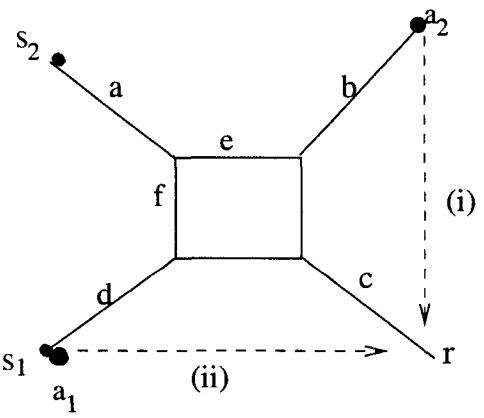

Figure 3.3: Abstract convex hull in case(i), (ii)

Note that the four points in the middle of the pictures in Figures 3.3 to 3.11, are virtual points.

In Figure 3.3, (i) and (ii) express two cases in our algorithm. TThe two cases consider two possible adversary moves. We explain it as follows:

Case (i):

At the step $t-1$:

$$\alpha_{t-1} = a + f + d$$

$$\beta_{t-1} = a + e + b$$

Therefore,

$$\Phi_{t-1} = a + f + d + 2a + 2e + 2b = 3a + 2b + 2e + f + d$$

The probability of moving s_1 to r is:

$$x/|s_1 s_2| = \frac{a+f}{a+f+d}$$

And the probability of moving s_2 to r is:

$$y/|s_1 s_2| = \frac{d}{a+f+d}$$

At the step t, if we move s_1, our cost of this move is $d + e + c$.

$$\alpha_t = a + f + e + c$$

$$\beta_t = a + f + d$$

Therefore,

$$\Phi_t = a + f + e + c + 2(a + f + d) = 3a + 3f + 2d + e + c$$

At the step t, if we move s_2, our cost of this move is $a + f + e + c$.

$$\alpha_t = d + e + c$$

$$\beta_t = 0$$

Therefore,

$$\Phi_t = d + e + c + 0 = d + e + c$$

The adversary cost for this move is $b + c + f$.

Therefore, we obtain Figure 3.4.

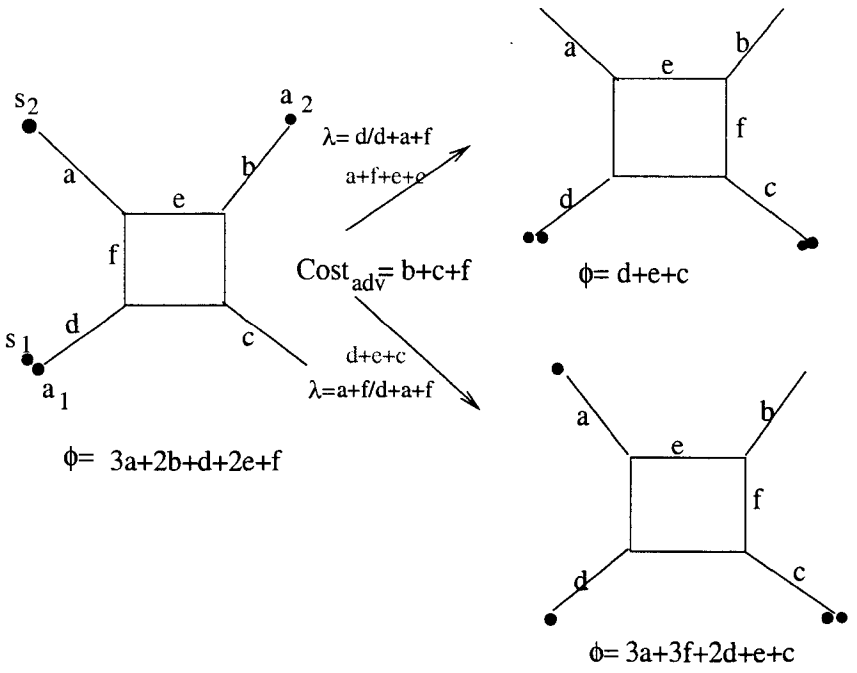

Figure 3.4: Case (i)

$$E(cost_{RS}) = \frac{d(a+c+e+f) + (a+f)(d+c+e)}{d+a+f}$$

$$E(\Phi) = \frac{d(d+e+c) + (a+f)(3a+3f+2d+e+c)}{d+a+f}$$

We need to prove:

$$E(cost_{RS}) + E(\Delta\Phi) \leq 2cost_{adv} \tag{3.2}$$

When we expand Inequality 3.2, we get:

$$\frac{d(a+c+e+f) + (a+f)(d+c+e)}{d+a+f}$$

$$
\begin{aligned}
&+ \quad \frac{d(d+e+c)+(a+f)(3a+3f+2d+e+c)}{d+a+f} \\
&- \quad (3a+2b+d+2e+f) \\
&\leq \quad 2(b+c+f)
\end{aligned}
$$

$$
\begin{aligned}
LHS &= da+dc+de+df+ad+ac+ae+fd+fc+fe+d^2+de+dc \\
&\quad +3a^2+3af+2ad+ae+ac+3af+3f^2+2fd+ef+fc \\
&\quad -3ad-2bd-d^2-2ed-df-3a^2-2ab-ad-2ea-af-3af-2bf \\
&\quad -df-2ef-f^2 \\
&= 2dc+2ac+2fc+2af+2f^2+2fd-2bd-2ab-af-2bf-f^2 \\
&= 2(dc+ac+fc+af+f^2+fd-bd-ab-bf)
\end{aligned}
$$

$$
RHS = 2(b+c+f)(d+a+f) = 2(db+dc+df+ab+ac+af+fb+fc+f^2)
$$

$$
\begin{aligned}
LHS-RHS &= dc+ac+fc+af+f^2+fd-bd-ab-fb-db-dc-df \\
&\quad -ab-ac-af-fb-fc-f^2 \\
&\leq 0
\end{aligned}
$$

Therefore, case (i) is proved.

Case (ii):

At the step $t-1$:

$$\alpha_{t-1} = a+f+d$$

$$\beta_{t-1} = a+e+b$$

Therefore,

$$\Phi_{t-1} = a + f + d + 2a + 2e + 2b = 3a + 2b + 2e + f + d$$

The probability of moving s_2 to r is:

$$y/|s_1 s_2| = \frac{d}{a + f + d}$$

And the probability of moving s_1 to r is:

$$x/|s_1 s_2| = \frac{a + f}{a + f + d}$$

At the step t, if we move s_1 to r, our cost of this move is $d + e + c$.

$$\alpha_t = a + f + e + c$$

$$\beta_t = a + e + b$$

Therefore,

$$\Phi_t = a + f + e + c + 2a + 2e + 2b = 3a + 3e + 2b + f + c$$

At the step t, if we move s_2 to r, our cost of this move is $a + f + e + c$.

$$\alpha_t = d + e + c$$

$$\beta_t = d + e + f + b$$

Therefore,

$$\Phi_t = d + e + c + 2(d + e + f + b) = 3d + 3e + 2f + 2b + c$$

The adversary cost for this move is $d + e + c$.

Therefore, we obtain Figure 3.5.

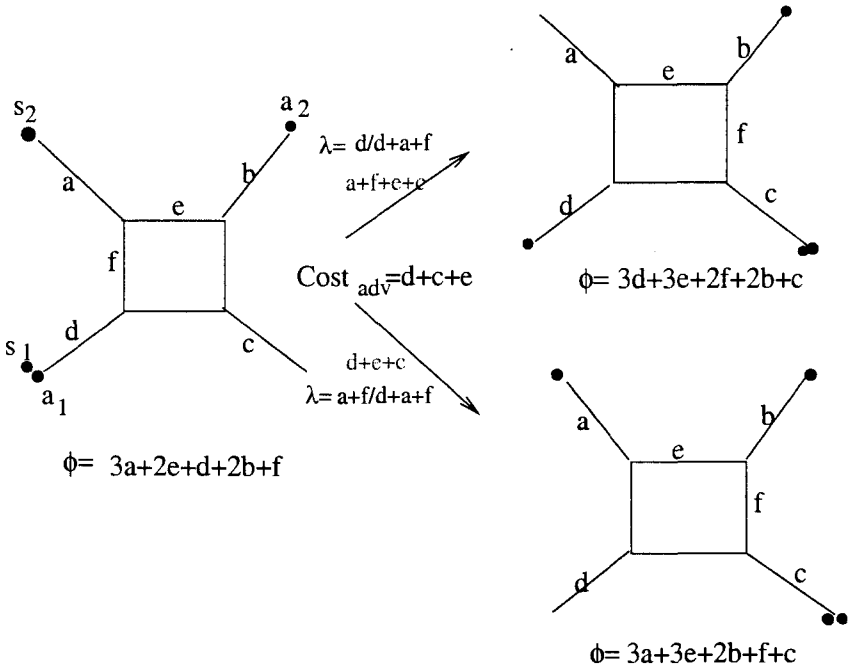

Figure 3.5: Case (ii)

$$E(cost_{RS}) = \frac{d(a+c+e+f)+(a+f)(d+c+e)}{d+a+f}$$

$$E(\Phi) = \frac{d(3d+3e+2f+2b+c)+(a+f)(3a+3e+2b+f+c)}{d+a+f}$$

We need to prove:

$$E(cost_{RS}) + E(\Delta\Phi) \leq 2cost_{adv} \tag{3.3}$$

When we expand Inequality 3.3, we get:

$$\begin{aligned} & \frac{d(a+c+e+f)+(a+f)(d+c+e)}{d+a+f} \\ + \quad & \frac{d(3d+3e+2f+2b+c)+(a+f)(3a+3e+2b+f+c)}{d+a+f} \\ - \quad & (3a+2b+d+2e+f) \\ \leq \quad & 2(d+e+c) \end{aligned}$$

$$
\begin{aligned}
LHS &= da + df + de + dc + ad + df + ae + ef + ac + fc + 3d^2 + 3ed \\
&\quad +2df + 2bd + dc + 3a^2 + 3af + 3ae + 3fe + 2ab + 2bf + af + f^2 \\
&\quad +ac + fc - 3a^2 - 2ae - 2ab - fa - da - 3af - 2fe - 2bf - f^2 \\
&\quad -fd - 3ad - 2ed - 2bd - fd - d^2 \\
&= 2ae + 2fe - 2ad + 2ed + 2d^2 + 2dc + 2ac + 2fc + 2df
\end{aligned}
$$

$$
RHS = 2(ad + ae + ac + fd + fe + fc + d^2 + de + dc)
$$

$$
\begin{aligned}
LHS - RHS &= ae + fe - ad + ed + d^2 + dc + ac + fc + df - ad \\
&\quad -ae - ac - fd - fe - fc - d^2 - de - cd = -2ad \\
&\leq 0
\end{aligned}
$$

Therefore, case (ii) is proved.

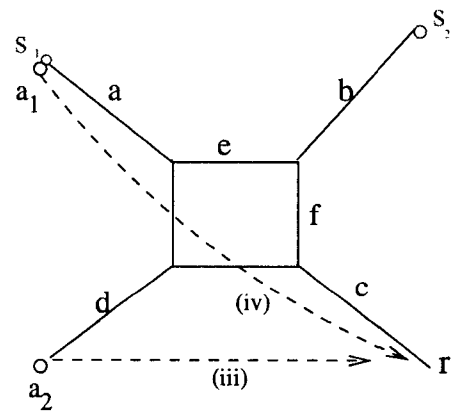

Figure 3.6: Abstract convex hull in case(iii), (iv)

In Figure 3.6, (*iii*) and (*iv*) express the two cases in our algorithm. The two case are two possible adversary moves.

Case (*iii*):

At the step $t-1$:

$$\alpha_{t-1} = a + e + b$$

$$\beta_{t-1} = d + e + f + b$$

Therefore,

$$\Phi_{t-1} = a + e + b + 2(d + e + f + b) = 2d + 3e + 2f + 3b + a$$

The probability of moving s_1 to r is:

$$y/|s_1 s_2| = \frac{b}{a + e + b}$$

And the probability of moving s_2 to r is:

$$x/|s_1 s_2| = \frac{a + e}{a + e + b}$$

At the step t, if we move s_1 to r, our cost of this move is $a + f + e + c$.

$$\alpha_t = b + f + c$$

$$\beta_t = a + e + b$$

Therefore,

$$\Phi_t = b + f + c + 2a + 2e + 2b = 3b + 2a + 2e + f + c$$

At the step t, if we move s_2 to r, our cost of this move is $b + f + c$.

$$\alpha_t = a + f + e + c$$

$$\beta_t = 0$$

Therefore,

$$\Phi_t = a + f + e + c + 0 = a + f + e + c$$

The adversary cost for this move is $d + e + c$.

Therefore, we obtain Figure 3.7.

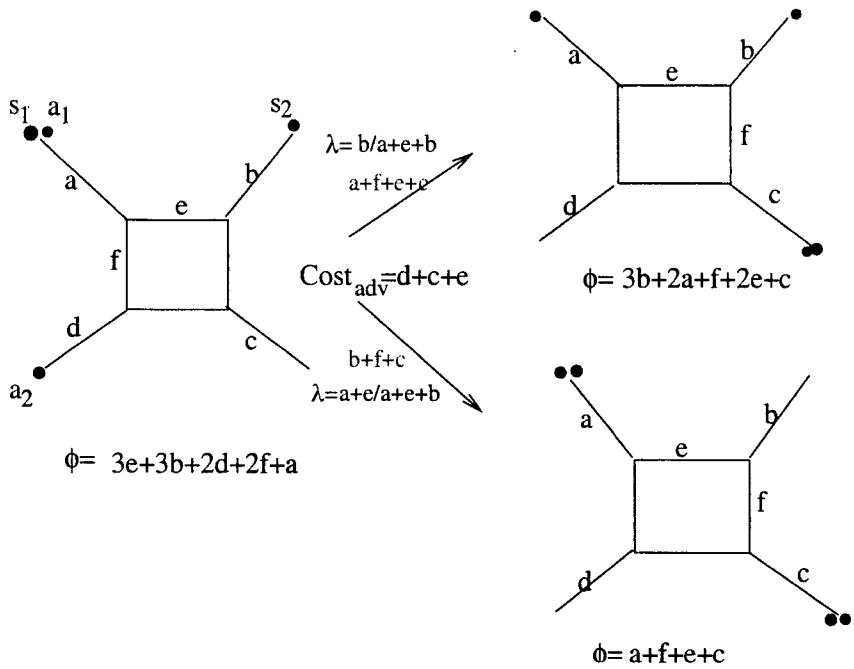

Figure 3.7: Case (iii)

$$E(cost_{RS}) = \frac{b(a + c + e + f) + (a + e)(b + c + f)}{e + a + b}$$

$$E(\Phi) = \frac{b(3b + 2a + 2e + f + c) + (a + e)(a + e + f + c)}{a + e + b}$$

We need to prove:

$$E(cost_{RS}) + E(\Delta\Phi) \leq 2cost_{adv} \tag{3.4}$$

When we expand Inequality 3.4, we get:

$$\begin{aligned}
& \frac{b(a+c+e+f)+(a+e)(b+c+f)}{b+a+e} \\
+ \; & \frac{b(3b+2a+2e+f+c)+(a+e)(a+e+f+c)}{e+a+b} \\
- \; & (3e+3b+2d+a+2f) \\
\leq \; & 2(d+e+c)
\end{aligned}$$

$$\begin{aligned}
LHS &= ab+bf+be+bc+ab+eb+af+ef+ac+ec+3b^2 \\
&\quad +2ab+2eb+bf+bc+a^2+ae+ae+e^2+fa+ef+ac+ec \\
&\quad -3ea-3ab-2af-2ad-a^2-3e^2-3eb-2ef-2ed-ae \\
&\quad -3eb-3b^2-2bf-2bd-ab \\
&= be+bc+ab+eb+ac+ec+2ab+2eb+bc+ae+e^2+ac \\
&\quad +ec-3ea-3ab-2ad-3e^2-3eb-2ed-3eb-2bd \\
&= -2eb+2bc+2ac+2ec-2ae-2e^2-2ad-2ed-2bd
\end{aligned}$$

$$RHS = 2(ad+ae+ac+ed+e^2+ec+bd+be+bc)$$

$$\begin{aligned}
LHS-RHS &= -eb+bc+ac+ec-ae-e^2-ad-ed-bd-ad \\
&\quad -ae-ac-ed-e^2-ec-bd-be-bc \\
&\leq 0
\end{aligned}$$

Therefore, case (iii) is proved.

Case (iv):

At the step $t-1$:

$$\alpha_{t-1} = a + e + b$$

$$\beta_{t-1} = d + e + f + b$$

Therefore,

$$\Phi_{t-1} = a + e + b + 2(d + e + f + b) = 3e + 3b + 2f + 2d + a$$

The probability of moving s_1 to r is:

$$y/|s_1 s_2| = \frac{b}{a + e + b}$$

And the probability of moving s_2 to r is:

$$x/|s_1 s_2| = \frac{a + e}{a + e + b}$$

At the step t, if we move s_1, our cost of this move is $a + f + e + c$.

$$\alpha_t = b + f + c$$

$$\beta_t = b + f + e + d$$

Therefore,

$$\Phi_t = b + f + c + 2b + 2f + 2e + 2d = 3b + 3f + c + 2d + 2e$$

At the step t, if we move s_2, our cost of this move is $b + f + c$.

$$\alpha_t = a + f + e + c$$

$$\beta_t = a + f + d$$

Therefore,

$$\Phi_t = a + f + e + c + 2(a + f + d) = 3a + 3f + 2d + e + c$$

The adversary cost for this move is $a + f + e + c$.

Therefore, we obtain Figure 3.8.

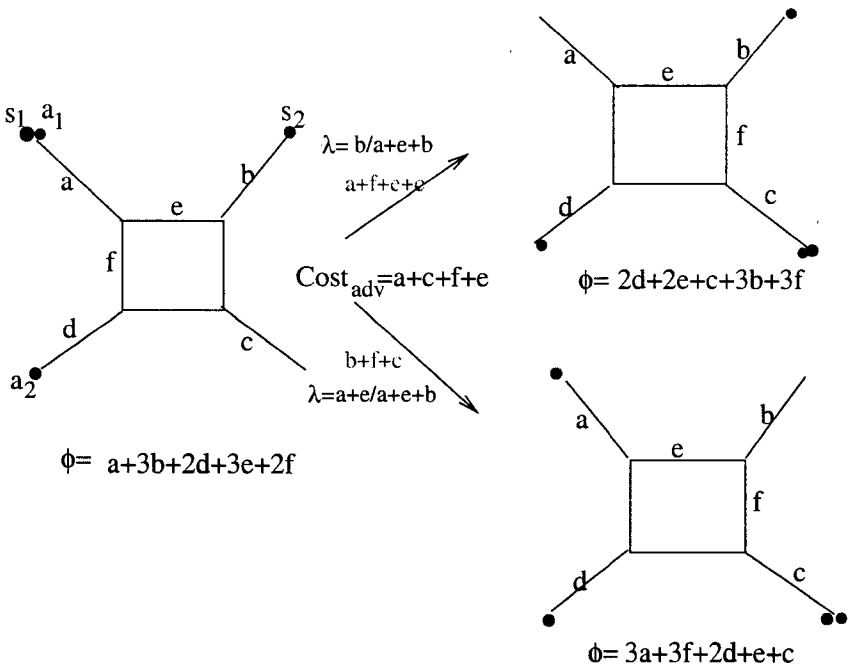

Figure 3.8: Case (iv)

$$E(cost_{RS}) = \frac{b(a+c+e+f)+(a+e)(b+c+f)}{e+a+b}$$

$$E(\Phi) = \frac{b(3b+3f+2e+2d+c)+(a+e)(3a+3f+2d+e+c)}{a+e+b}$$

We need to prove:

$$E(cost_{RS}) + E(\Delta\Phi) \leq 2cost_{adv} \tag{3.5}$$

When we expand Inequality 3.5, we get:

$$\frac{b(a+c+e+f)+(a+e)(b+c+f)}{b+a+e}$$

$$+ \quad \frac{b(3b+3f+2e+2d+c)+(a+e)(3a+3f+2d+e+c)}{e+a+b}$$

$$- \quad (3e+3b+2d+a+2f)$$

$$\leq \quad 2(a+f+e+c)$$

$$\begin{aligned} LHS &= ab+bf+be+bc+ab+eb+af+ef+ac+ec+3b^2 \\ &\quad +3bf+2eb+2bd+bc+3a^2+3ae+3af+3ef+2ad+2ed \\ &\quad +ae+e^2+ac+ec-3ea-3ab-2af-2ad-a^2-3e^2-3eb \\ &\quad -2ef-2ed-ae-3eb-3b^2-2bf-2bd-ab \\ &= bf+be+bc+ab+af+ef+ac+ec+3bf+bc+3a^2+3af \\ &\quad +3ef+e^2+ac+ec-3ba-2af-a^2-3e^2-3be-2ef-2bf \\ &= 2bf-2be+2bc-2ba+2af+2ef+2ac+2ec+2a^2-2e^2 \end{aligned}$$

$$RHS = 2(a^2+af+ae+ac+ae+ef+e^2+ec+ab+bf+eb+bc)$$

$$\begin{aligned} LHS-RHS &= bf-be+bc-ba+af+ef+ac+ec+a^2-e^2-a^2 \\ &\quad -af-ae-ac-ae-ef-e^2-ec-ab-bf-eb-bc \\ &\leq 0 \end{aligned}$$

Therefore, case (iv) is proved.

In Figure 3.9, (v) and (vi) express the two cases in our algorithm. The two case are two possible adversary moves.

Case (v):

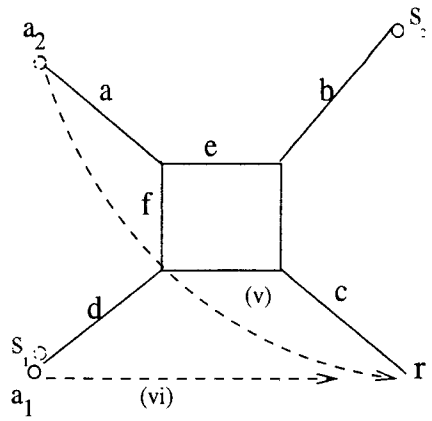

Figure 3.9: Abstract convex hull in case(v), (vi)

At the step $t-1$:

$$\alpha_{t-1} = e + f + d + b$$

$$\beta_{t-1} = a + e + b$$

Therefore,

$$\Phi_{t-1} = e + f + d + b + 2a + 2e + 2b = 3e + 3b + 2a + d + f$$

The probability of moving s_1 to r is:

$$y/|s_1 s_2| = \frac{b+f}{d+e+f+b}$$

And the probability of moving s_2 to r is:

$$x/|s_1 s_2| = \frac{d+e}{d+e+f+b}$$

At the step t, if we move s_1 to r, our cost of this move is $d + e + c$.

$$\alpha_t = b + f + c$$

$$\beta_t = e + b + f + d$$

Therefore,

$$\Phi_t = b + f + c + 2(e + b + f + d) = 3b + 3f + 2e + 2d + c$$

At the step t, if we move s_2 to r, our cost of this move is $b + f + c$.

$$\alpha_t = e + d + c$$

$$\beta_t = 0$$

Therefore,

$$\Phi_t = d + e + c + 0 = d + e + c$$

The adversary cost for this move is $a + f + e + c$.

Therefore, we obtain Figure 3.10.

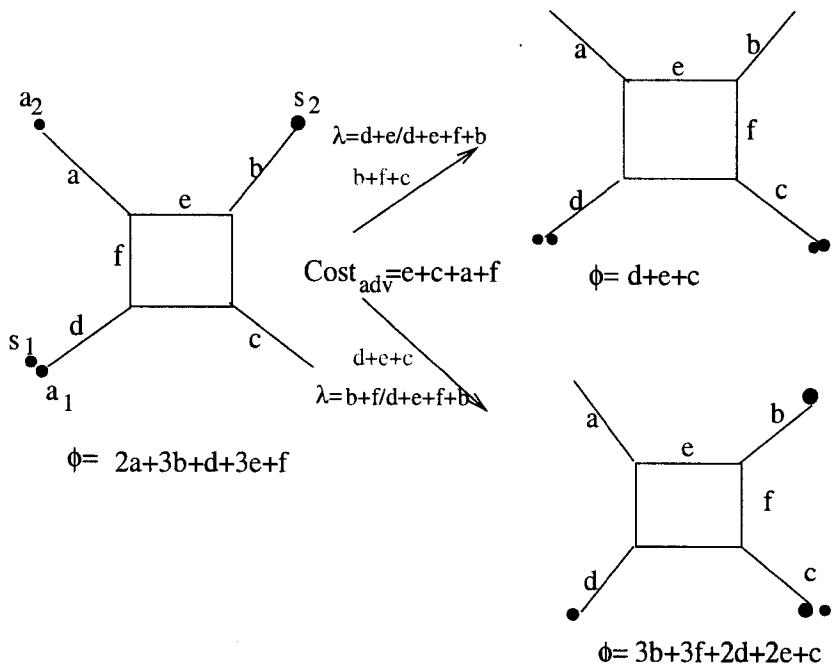

Figure 3.10: Case (v)

$$E(cost_{RS}) = \frac{(d+e)(b+f+c) + (b+f)(d+e+c)}{d+e+f+b}$$

$$E(\Phi) = \frac{(d+e)(d+e+c) + (b+f)(3b+3f+2e+2d+c)}{d+e+f+b}$$

We need to prove:

$$E(cost_{RS}) + E(\Delta\Phi) \leq 2cost_{adv} \tag{3.6}$$

When we expand Inequality 3.6, we get:

$$\begin{aligned}
& \frac{(d+e)(b+f+c)+(b+f)(d+e+c)}{d+e+f+b} \\
+ \; & \frac{(d+e)(d+e+c)+(b+f)(3b+3f+2e+2d+c)}{d+e+f+b} \\
- \; & (3b+3e+2a+d+f) \\
\leq \; & 2(a+f+e+c)
\end{aligned}$$

$$\begin{aligned}
LHS &= db+eb+df+ef+dc+ec+bd+fd+be+fe+bc+fc \\
&\quad +d^2+ed+de+e^2+dc+ec+3b^2+3bf+3bf+3f^2+2eb \\
&\quad +2ef+2bd+2df+bc+fc-3bd-3ed-2ad-d^2-df-3be \\
&\quad -3e^2-2ae-de-ef-3bf-3ef-2af-df-f^2-3b^2-3be \\
&\quad -2ab-bd-bf \\
&= db+eb+dc+ec+be+bc+fc+de+e^2+dc+ec+3bf+3f^2 \\
&\quad +2eb+2bd+2df+bc+fc-3bd-3ed-2ad-3e^2-2ae-2af \\
&\quad -f^2-2ab-bf-3be-3be \\
&= -2be+2dc+2ec+2bc+2fc-2de-2e^2+2bf+2f^2+2df \\
&\quad -2ad-2ae-2af-2ab
\end{aligned}$$

$$RHS = 2(ad+ae+af+ab+fd+fe+f^2+fb+ed+e^2+ef+eb$$

$$+dc + ec + fc + bc)$$

$$\begin{aligned} LHS - RHS &= -be + dc + ec + bc + fc - de - e^2 + bf + f^2 + df - ad \\ &\quad -ae - af - ab - ad - ae - af - ab - fd - fe - f^2 - fb \\ &\quad -ed - e^2 - ef - eb - dc - ec - fc - bc \\ &\leq 0 \end{aligned}$$

Therefore, case (v) is proved.

Case (vi):

At the step $t-1$:

$$\alpha_{t-1} = d + f + e + b$$

$$\beta_{t-1} = a + e + b$$

Therefore,

$$\Phi_{t-1} = d + f + e + b + 2(a + e + b) = 3e + 3b + 2a + d + f$$

The probability of moving s_1 to r is:

$$y/|s_1 s_2| = \frac{b + f}{d + e + f + b}$$

And the probability of moving s_2 to r is:

$$x/|s_1 s_2| = \frac{d + e}{d + e + f + b}$$

At the step t, if we move s_1 to r, our cost of this move is $d + e + c$.

$$\alpha_t = b + f + c$$

$$\beta_t = a + e + b$$

Therefore,

$$\Phi_t = b + f + c + 2(a + e + b) = 3b + 2e + 2a + f + c$$

At the step t, if we move s_2 to r, our cost of this move is $b + f + c$.

$$\alpha_t = d + e + c$$

$$\beta_t = a + f + d$$

Therefore,

$$\Phi_t = d + e + c + 2(a + f + d) = 3d + 2f + 2a + e + c$$

The adversary cost for this move is $d + e + c$.

Therefore, we obtain Figure 3.11.

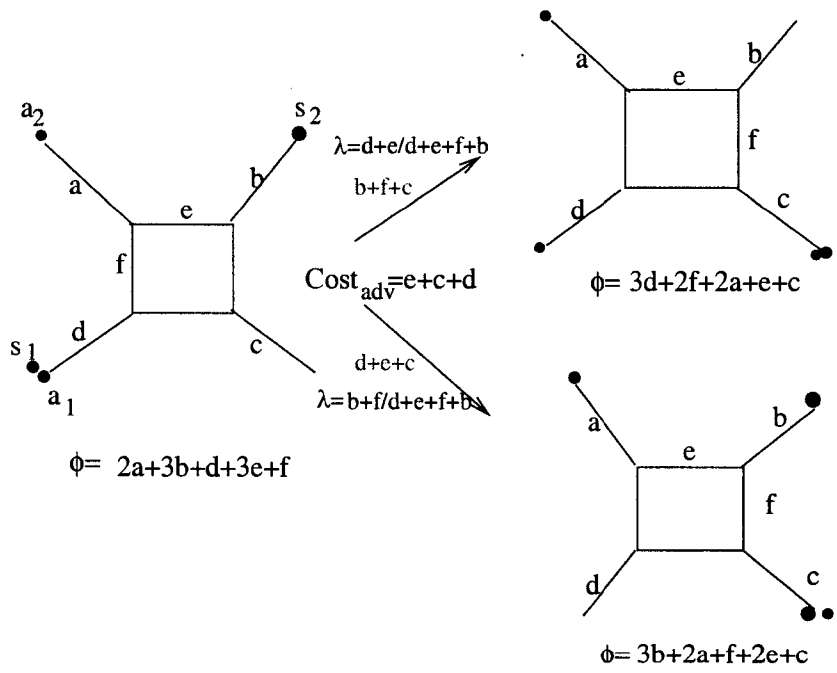

Figure 3.11: Case (vi)

$$E(cost_{RS}) = \frac{(d + e)(b + f + c) + (b + f)(d + e + c)}{d + e + f + b}$$

$$E(\Phi) = \frac{(d+e)(3d+2f+2a+e+c)+(b+f)(3b+2a+2e+f+c)}{d+e+f+b}$$

We need to prove:

$$E(cost_{RS}) + E(\Delta\Phi) \leq 2cost_{adv} \tag{3.7}$$

When we expand Inequality 3.7, we get:

$$\begin{aligned}
& \frac{(d+e)(b+f+c)+(b+f)(d+e+c)}{d+e+f+b} \\
+ \quad & \frac{(d+e)(3d+2f+2a+e+c)+(b+f)(3b+2a+2e+f+c)}{d+e+f+b} \\
- \quad & (3b+3e+2a+d+f) \\
\leq \quad & 2(d+e+c)
\end{aligned}$$

$$\begin{aligned}
LHS \quad = \quad & db+eb+df+ef+dc+ec+bd+fd+be+fe+bc+fc \\
& +3d^2+3ed+2df+2ef+2ad+2ae+de+e^2+dc+ec+3b^2 \\
& +3bf+2ab+2af+2be+2ef+fb+f^2+bc+fc-3bd-3ed-2ad \\
& -d^2-df-3be-3e^2-2ae-de-ef-3bf-3ef-2af-df-f^2 \\
& -3b^2-3eb-2ab-db-fb \\
= \quad & -2eb+2df+2ef+2dc+2ec-2bd+2bc+2fc+2d^2-2e^2
\end{aligned}$$

$$RHS \quad = \quad 2(d^2+de+fd+bd+ed+e^2+ef+be+cd+ec+fc+bc)$$

$$LHS - RHS \quad = \quad -be+df+ef+dc+ec-bd+bc+fc+d^2-e^2-d^2-de$$

$$
\begin{aligned}
& -fd - bd - ed - e^2 - ef - be - cd - ec - fc - bc \\
= \; & -2be - 2bd - 2e^2 - 2de \\
\leq \; & 0
\end{aligned}
$$

Therefore, case (vi) is proved.

□

Coopersmith, Doyle, Raghavan and Snir [CDRS93] proved that any memoryless k-server algorithm on any metric space has competitive ratio at least k against an oblivious adversary.

For the k-server problem, HARMONIC is a memoryless and trackless randomized online algorithm. Its competitiveness is at least $\begin{pmatrix} k+1 \\ 2 \end{pmatrix}$ [RS94]. It is 3-competitive when $k = 2$ [CL92a]. Bartal and Grove showed in [BG00] that HARMONIC is $O(2^k \log k)$ for all k. This is so far the best upper bound known for HARMONIC.

The simple algorithm RANDOM SLACK [CDRS93] has the best known competitive ratio for any randomized online algorithm for the 2-server problem for arbitrary metric spaces. This algorithm is trackless and memoryless.

CHAPTER 4

PAGING AND KNOWLEDGE STATE ALGORITHMS

4.1 Background, Definitions, and the Knowledge State Approach

This thesis is about the server problem. A natural question would be: "Why do we consider the cache problem here?" As we know, there are k mobile identical servers in a metric space M in the *server problem*. One point in M will be "requested" and have to be "served" by moving one of the k servers to it. An online algorithm for the server problem decides which server should be moved without knowing the entire request sequence. The paging problem, or cache problem, is in fact a special case of the server problem with the stipulation that the metric space is uniform. A metric space is uniform if the distance between any two points in the space M is the same.

The Figure 4.1 illustrates why the k-cache problem really is the k-server problem in a metric space where all distances are the same. In the figure all of the edges have same length of one. If we have *CNN* and *QINZ* in our cache, and if there is a request to *Yahoo*, we have to eject either *CNN* or *QINZ* from the cache and then we can bring *Yahoo* into our cache. The cost of this move is the default cost of one. But, if *CNN* is the requested page, the cost will be zero, because it is already in the cache. In general, the cache can hold k pages, for some given $k \geq 2$.

One tool we have used earlier in this thesis is the concept of a *work function*. For

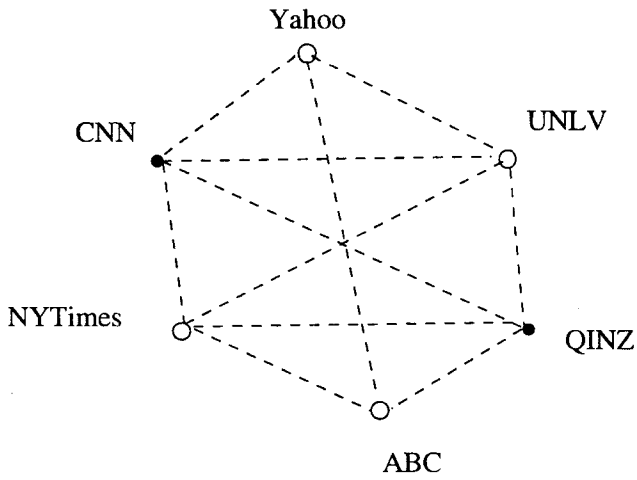

Figure 4.1: A special case of the server problem

a request sequence ρ, by $\omega^\rho(x)$, we denote the minimum cost of serving ρ and ending in configuration x. A *configuration* is simply an unordered k-tuple of pages, and represents a possible cache configuration. We denote the set of all configurations as $\mathcal{X}$. If $x, y \in X$, we let $d(x, y)$ represent the minimum matching distance. (To be more precise, at the beginning or after several invalidation misses the cache may hold fewer than k pages, but it suffices to consider only complete configurations.) The optimal cost to service a request sequence ρ is given by $opt(\rho) := \min_x \omega^\rho(x)$. It can be computed by dynamic programming. As time increases, the work function grows without bound. It is convenient to consider $\omega^\rho - opt(\rho) : \mathcal{X} \to \mathbb{N}$, which is called the *offset function*, and which is non-negative.

For k-cache problem, the "bar notation" is a kind of notation used to describe the *offset functions* for that problem. A bar notation is a string of symbols consisting of page name(s) or bar(s).

The maximal substring of page names in a bar notation is called a *block*. A bar notation must satisfy the following rules:

- Each page name can appear only once;

- Exactly k bars appear in the notation;
- The last symbol in the notation must be a bar;
- At least i page names must appear before the i^{th} bar;
- If the first i symbols are page names, the next i symbols must be bars.
- If exactly i page names appear in a block, and if the block is preceded by a bar, then the block cannot be followed by i consecutive bars.

Usually, lower case letters are used to represent page names.

We use the bar notation as the name of each ω, defined as follows:

- $\omega(x) = 0$, $x \in S$.
- Let S_i be the set of all pages whose names appear before the i^{th} bar in the bar notation. Let x be a set of k pages, then $x \in S$ if and only if $x \cap S_i$ has cardinality at least i, for each i.

A page a is active for ω if and only a is in the bar notation of ω.

The position of any page name in any given block can be changed without changing the meaning of the bar notation.

After this introduction we will describe two new algorithms for the paging problem. We will use a novel technique, called the *knowledge state* approach, which we will briefly summarize now. This technique is described in detail in [BLR04] and a number of definitions are taken from that paper and inserted here. The *knowledge state algorithm* belongs to the mixed online algorithm which keeps an update-offset system and uses the current estimator to indicate its memory state. The work summarized in this chapter also appears in [BLR04].

A *knowledge state algorithm* [BLR04] is a mixed online algorithm that computes an offset and an estimator at each step, and uses the current estimator as its memory state. The estimator is a real-valued function on configurations that is updated at every step, and which estimates the cost of the optimal offline algorithm, while the offset is a real number that is computed at every step. We note that a function $\omega : \mathcal{X} \to \mathbf{R}$ is *Lipschitz* if $\omega(y) \leq \omega(x) + d(x,y)$ for all $x, y \in \mathcal{X}$ (We will define $d(x,y)$ later in this chapter). Generally we stipulate an *estimator* to be a non-negative Lipschitz function $\mathcal{X} \to \mathbf{R}$. If $S \subseteq \mathcal{X}$, we say that S *supports an estimator* ω if, for any $y \in \mathcal{X}$ there exists some $x \in S$ such that $\omega(y) = \omega(x) + d(x,y)$. If ω is supported by a finite set, then there is a unique minimal set S which supports ω, which we call the *support* of ω. We note that all estimators considered in this paper have finite support. We say that an estimator ω has *zero minimum* if $\min_{x \in \mathcal{X}} \omega(x) = 0$.

Both the estimator and the offset may be calculated using randomization. More formally [BLR04], if $\mathcal{A}$ is a knowledge-state algorithm, then:

1. At any given step, the full state of $\mathcal{A}$ is a pair (π, ω), where $\pi \in \Pi$ and $\omega : \mathcal{X} \to \mathbf{R}$ is the current estimator. Π is the set of all distributions, on the set of configurations. We call that pair the *current knowledge state.*

2. If $k = (\pi, \omega)$ is the knowledge state and the next request is r, then $\mathcal{A}$ computes an offset, a number which we call $\mathit{offset}_{\mathcal{A}}(k, r)$, and uses randomization to pick a new knowledge state $k' = (\pi', \omega')$. More precisely, there are subsequent knowledge states $k_i = (\pi_i, \omega_i)$ and subsequent positive weights λ_i for $i = 1, \ldots m$, $\sum_{i=1}^{m} \lambda_i = 1$, such that

   (a) $(\omega \wedge r)(x) \geq \mathit{offset}_{\mathcal{A}}(k, r) + \sum_{i=1}^{m} \lambda_i \omega_i(x)$ for each $x \in \mathcal{X}$, where we define function

$\omega \wedge r$ as $(\omega \wedge r)(y) = \min_{x \in \mathcal{X}} \{\omega(x) + cost(x, r, y)\}$.

(b) For each i, $\mathcal{A}$ chooses k' to be k_i with probability λ_i.

For the k-cache problem for fixed $k \geq 2$ we have:

1. There is a set of *pages*. $\mathcal{X}$ is the set of all k-tuples of distinct pages. If the configuration of an algorithm is $x \in \mathcal{X}$, that means that the pages that constitute x are in the cache. The initial configuration is the initial cache.

2. If $x, y \in \mathcal{X}$, then $d(x, y)$ is the cost of changing the cache from x to y. Since we assume that it costs 1 to eject a page and bring in a new page, $d(x, y)$ is the cardinality of the set $x - y$.

3. $\mathcal{R}$ is simply the set of all pages. If a page r is requested, it means that the algorithm must ensure that r is in the cache at some point as it moves between configurations. Thus: For any $x, y \in \mathcal{X}$ and any $r \in \mathcal{R}$, we have

$$cost(x, r, y) = \begin{cases} 2 \text{ if } x = y,\ r \notin x \\ d(x, y) \text{ if } r \in x \text{ or } r \in y \\ d(x, y) + 1 \text{ otherwise} \end{cases}$$

We now define a C-knowledge state potential [BLR04] for a given knowledge state algorithm $\mathcal{A}$. Let $\Phi_{\mathcal{A}}$ be a real-valued function on knowledge states. Then we say that $\Phi_{\mathcal{A}}$ is a *C-knowledge state potential for* $\mathcal{A}$ if

1. $\Phi_{\mathcal{A}}(k) \geq 0$ for any k.

2. If $k = (\pi, \omega)$ is the current knowledge state and r is the next request, $\{k_i = (\pi_i, \omega_i)\}$ are the subsequents of that request, and $\{\lambda_i\}$ are the weights of the subsequents, let

$\Delta\Phi_{\mathcal{A}}(k,r) = \sum_{i=1}^{m} \lambda_i \Phi_{\mathcal{A}}(\pi_i, \omega_i) - \Phi_{\mathcal{A}}(\pi, \omega)$. Then

$$cost_{\mathcal{A}}(k,r) + \Delta\Phi_{\mathcal{A}}(k,r) \leq C \cdot offset_{\mathcal{A}}(k,r).$$

Theorem 5 *If a knowledge state algorithm $\mathcal{A}$ has a C-knowledge state potential, then $\mathcal{A}$ is C-competitive.*

For the paging problem, at the low end of memory requirements, we have memoryless algorithms. RAND is an example of the "memoryless" randomized online algorithm. For the k-cache problem, its competitiveness is k. RAND pursues the simple strategy that in case of a fault, it ejects a page at random . Based on the above concept, a memoryless algorithm is trackless for sure. But, a trackless algorithm is not necessarily memoryless. One typical example is LRU, which is trackless, but LRU has to have memory to remember the ordering of its cache pages by recency of use. Fiat, Karp, Luby, McGeach , Sleator and Young [FKL+91] showed that RMARK has competitive ratio $2H_k$ against an oblivious adversary. This algorithm is based on the same idea as RAND, but it marks each page as it is used, and always ejects an unmarked page when there is a fault. Once, all of the pages have been marked, it will erase all the marks. RMARK is trackless and uses $O(k)$ memory.

The memory of an online algorithm for the paging problem may also consist of "bookmarks". We use "bookmarks" to remember those pages which have been ejected and might be requested later. Bookmarks are cumbersome to keep. For example, in the case of the world-wide web, it is hard to keep track of the exact identity of a page.

Fiat, Karp, Luby, McGeach, Sleator and Young [FKL+91] proved that for any randomized paging algorithm, if the number of pages is greater than or equal to $k+1$, where k is the cache size, the competitive ratio of this algorithm against any oblivious adversary

is greater than or equal to H_k. PARTITION [KP94a] is the first randomized online algorithm for the paging algorithm to achieve the optimal competitiveness of H_k. PARTITION keeps track of everything, *i.e.*, it uses an unlimited number of bookmarks. But, we do not need to keep track of everything to achieve the optimal competitiveness. Achlioptas *et. al.* [ACN96] showed that EQUITABLE can achieve the competitiveness of H_k by maintaining a constant number of bookmarks. The constant is a rapidly growing function of k. In fact, EQUITABLE uses $O(k^2 \log k)$ bookmarks. It is an open question whether optimality can be achieved with $O(k)$ bookmarks.

We will address this issue here for $k = 2$ and $k = 3$. In the next two sections, for the 2-cache problem, we describe an algorithm K_2 with one bookmark; for the 3-cache problem, we construct the algorithm K_3. The Algorithm K_3 uses six knowledge states and 2 bookmarks. These algorithms demonstrate that the knowledge state approach simplifies memory requirements. K_2 is $H_2 = \frac{3}{2} - competitive$, while K_3 is $H_3 = \frac{11}{6} - competitive$. They both achieve the known lower bound of H_k.

4.2 A Knowledge State Algorithm for the 2-Cache Problem - K_2

We now define the knowledge state algorithm K_2 for the 2-cache problem. Each knowledge state of K_2 is supported by a set of cardinality at most 2.

Knowledge States of K_2. We will keep the minimum of the estimator at zero by choosing, at each step, the offset to be as large as possible. In this way, the potential will always be non-negative. Two knowledge states are said to be *equivalent*, if each can be obtained from the other by simply renaming the pages.

K_2 will always have only two knowledge states, up to equivalence, although there are

infinitely many pages. We show the two knowledge states as follows:

1. We define $A^{a,b} = (\alpha^{a,b}, [a,b])$ to be the knowledge state supported by the configuration $\{a,b\}$, where $\alpha^{a,b}(\{a,b\}) = 0$, and $[a,b]$ is the distribution concentrated on $\{a,b\}$. $\alpha^{a,b} = ab||$.

2. We define $B^{a,b,c} = (\beta^{a,b,c}, \frac{1}{2}[a,b] + \frac{1}{2}[a,c])$ to be the knowledge state supported by $\{\{a,b\},\{a,c\}\}$, where $\beta^{a,b,c}(\{a,b\}) = \beta^{a,b,c}\{a,c\} = 0$. $\beta^{a,b,c} = a|bc|$.

Here, please notice that $A^{a,b} = A^{b,a}$ and $B^{a,b,c} = B^{a,c,b}$ by symmetry.

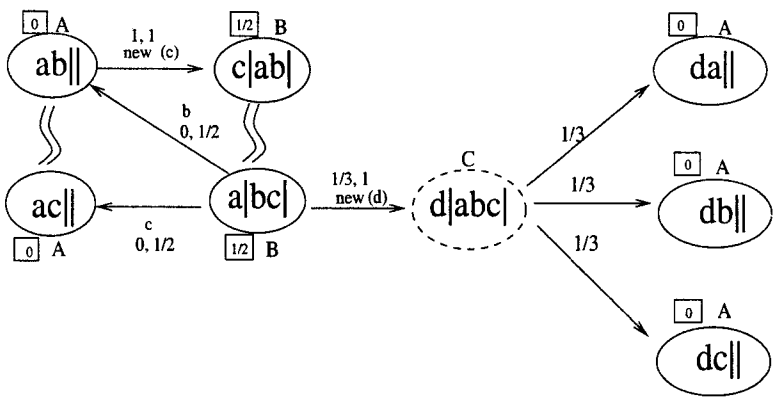

Figure 4.2: K_2 actions

K_2 Actions. K_2 has the actions as illustrated by Figure 4.2. In this figure, the number in the square is the potential at that state. On each arrow, there is a letter denoting the requested page. The word "new" means the request is to a new page. Also on each arrow, there is a pair of numbers, the first of which is the offset, and the second of which is the cost between two steps. The double wavy lines mean equivalence between two states. The dotted oval represents the *fleeting state*, meaning that the state C doesn't exist at the end of the step. It is there for explanation only.

We explain the actions of K_2 as follows:

- The initial knowledge state is $A^{a,b}$,
- When the current knowledge state is $A^{a,b}$, the new request is a, the new knowledge state is $A^{a,b}$, $\mathit{offset} = 0$, and $\mathit{cost} = 0$.
- When the current knowledge state is $A^{a,b}$, the new request is some page $c \notin \{a, b\}$, then the new knowledge state is $B^{c,a,b}$, $\mathit{offset} = 1$, and $\mathit{cost} = 1$.
- When the current knowledge state is $B^{a,b,c}$, the new request is a, then the new knowledge state is $B^{a,b,c}$, $\mathit{offset} = 0$, and $\mathit{cost} = 0$.
- When the current knowledge state is $B^{a,b,c}$, the new request is b, then the new knowledge state is $A^{b,a}$ (which is equivalent to $A^{a,b}$), $\mathit{offset} = 0$, and $\mathit{cost} = \frac{1}{2}$.
- When the current knowledge state is $B^{a,b,c}$, the new request is c, then the new knowledge state is $A^{c,a}$ (which is equivalent to $A^{a,c}$), $\mathit{offset} = 0$, and $\mathit{cost} = \frac{1}{2}$.
- When the current knowledge state is $B^{a,b,c}$, the new request is some page $d \notin \{a, b, c\}$, then there are three subsequents $A^{d,a}$, $A^{d,b}$, and $A^{d,c}$. These three subsequents have the uniform distribution that means each subsequent is chosen with probability $\frac{1}{3}$. $\mathit{offset} = \frac{1}{3}$.

Theorem 6 *K_2 is $\frac{3}{2}$-competitive.*

Proof: We define $\Phi(A^{a,b}) = 0$ and $\Phi(B^{a,b,c}) = \frac{1}{2}$. Φ is a $\frac{3}{2}$-potential for K_2. In order to prove this theorem, let K be the current knowledge state and r the new request, we will show that:

$$cost + \Delta\Phi \leq \tfrac{3}{2} offset \tag{4.1}$$

holds in every case.

We first note that, for distinct pages a, b, c, d: (Recall that $\alpha^{a,b} = \alpha^{b,a}$ and $\beta^{a,b,c} = \beta^{a,c,b}$.)

$$\begin{aligned}
\alpha^{a,b}\wedge a &= \alpha^{a,b} \\
\alpha^{a,b}\wedge b &= \alpha^{a,b} \\
\alpha^{a,b}\wedge c &= \beta^{c,a,b} + 1 \\
\beta^{a,b,c}\wedge a &= \beta^{a,b,c} \\
\beta^{a,b,c}\wedge b &= \alpha^{a,b} \\
\beta^{a,b,c}\wedge c &= \alpha^{a,c} \\
\beta^{a,b,c}\wedge d &\geq \tfrac{1}{3}\alpha^{a,d} + \tfrac{1}{3}\alpha^{b,d} + \tfrac{1}{3}\alpha^{c,d} + \tfrac{1}{3}
\end{aligned}$$

The last inequality need only be verified for the support set of $\beta^{a,b,c}\wedge d$ which are the configurations in $\{\{a,d\},\{b,d\},\{b,d\}\}$.

Figure 4.3 shows that Inequality 4.1 is proved.

In Figure 4.3, we have two set of numbers which represent two different kind of regular moves for K_2. The arrows in this figure illustrate the directions of the changes. For example, the first arrow on the left hand indicates that, from A^{ab} to B^{cab}, when requesting c, the cost is one. The little number that resides in the circle is the distribution on the *support set.* The little number that resides in the square is the potential for that particular state.

To prove inequality

$$cost + \Delta\Phi \leq \tfrac{3}{2} offset$$

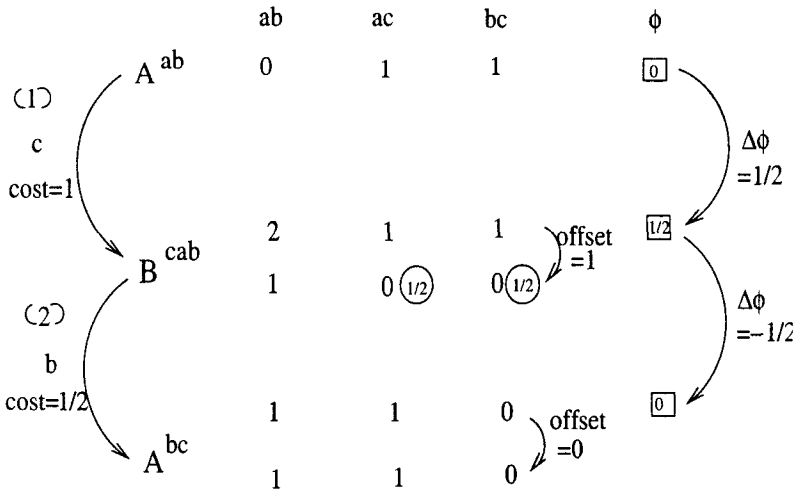

Figure 4.3: Regular moves for K_2

Expand the above inequality, we get:

(1) $1 + 1/2 \leq 3/2 \cdot 1$

(2) $1/2 + (-1/2) \leq 3/2 \cdot 0$

Therefore, the proof for regular moves is done.

Figure 4.4 proves that in the Las Vegas steps for the K_2 actions, the offset is $1/3$ and $\Delta\Phi$ is $-1/2$

In Figure 4.4, the arrows in this figure illustrate the directions of the changes. For example, the first arrow on the left hand side indicates that, when updating a, the estimators are updated. The third, fourth and fifth rows are estimators for three subsequents. These subsequents are uniformly distributed. The last row indicates the average estimators of the above three rows. The difference of the second row and the last row is the *offset* we are looking for. The little number that resides in the circle is the distribution of the *support set.* The little number that resides in the square is the potential for that particular state.

Figure 4.5 proves that the minimum transportation cost in the Las Vegas steps for the

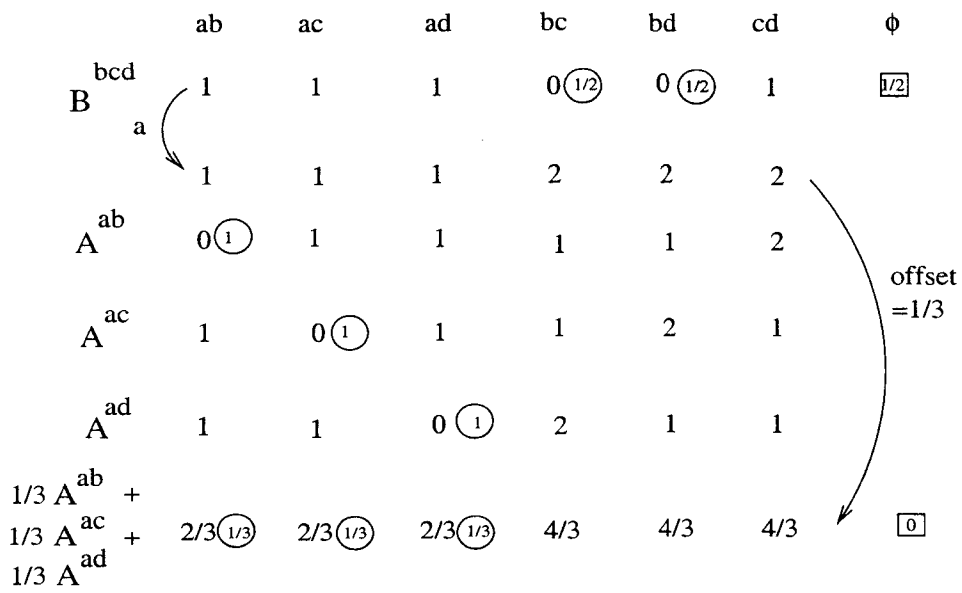

Figure 4.4: Las Vegas Step for K_2

K_2 actions is:

$$cost = 1 \cdot 2/6 + 1 \cdot 1/6 + 1 \cdot 1/6 + 1 \cdot 2/6 = 1$$

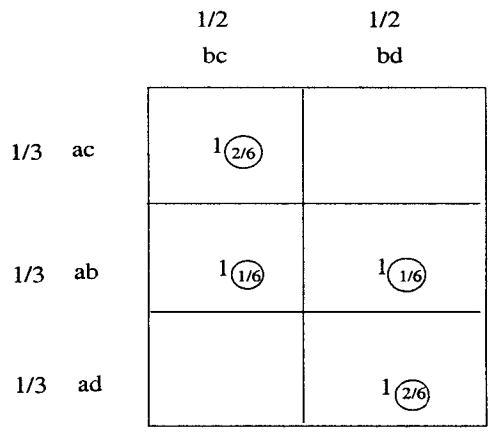

Figure 4.5: Las Vegas Step table for k_2

In Figure 4.5, the little numbers that reside in the circles are obtained by the following technique:

- The sum of the numbers in the same column should be equal to the number marked outside of that column;
- The sum of the numbers in the same row should be equal to the number marked outside of that row.

The numbers outside of the table indicate the distributions of the subsequents. The other numbers inside of the table indicate the costs.

We want to prove

$$cost + \Delta\Phi \leq \tfrac{3}{2} offset$$

Expanding the above inequality, we get:

$$1 + (-1/2) \leq 3/2 \cdot 1/3$$

which is correct. Therefore, the Las Vegas step actions in K_2 are proved.

Now, we prove each case:

Case $K = A^{a,b}$ and $r = a$ or $r = b$:

$cost = 0$, $offset = 0$, and $\Delta\Phi = 0$. Therefore, 4.1 is obvious.

Case $K = B^{a,b,c}$ and $r = a$:

$cost = 0$, $offset = 0$, and $\Delta\Phi = 0$. Therefore, 4.1 is obvious.

Case $K = A^{a,b}$ and $r = c \notin \{a, b\}$:

$offset = 1$, $cost = 1$ and $\Delta\Phi = \frac{1}{2}$. Therefore, 4.1 is proved.

Case $K = B^{a,b,c}$ and $r = b$ or $r = c$:

Without loss of generality, $r = b$. Because $\beta^{a,b,c} \wedge b = \alpha^{b,a}$, $offset = 0$. $cost = \frac{1}{2}$ and $\Delta\Phi = -\frac{1}{2}$, therefore 4.1 is proved.

Case $K = B^{a,b,c}$ and $r = d \notin \{a, b, c\}$:

Because $\beta^{a,b,c} \wedge d \geq \frac{1}{3}\alpha^{a,d} + \frac{1}{3}\alpha^{b,d} + \frac{1}{3}\alpha^{c,d} + \frac{1}{3}$, $offset = \frac{1}{3}$. $cost = 1$ and $\Delta\Phi = -\frac{1}{2}$, 4.1 is proved.

□

Table 4.1 summarizes all actions and all of our analysis of K_2 (except for the simple actions.)

k	r	$\{k_i\}$	$\{\lambda_i\}$	$cost$	$offset$	$\Phi(k)$	$E(\Phi(k'))$	$\Delta\Phi$	$\frac{C \cdot offset}{-\Delta\Phi - cost}$
$A^{a,b}$	c	$B^{c,a,b}$	1	1	1	0	$\frac{1}{2}$	$\frac{1}{2}$	0
$B^{a,b,c}$	b	$A^{a,b}$	1	$\frac{1}{2}$	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	0
$B^{a,b,c}$	d	$A^{d,a}$ $A^{d,b}$ $A^{d,c}$	$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{2}$	0	$\frac{1}{2}$	0

Table 4.1: Summary for K_2

4.3 A Knowledge State Algorithm for the 3-Cache Problem - K_3

K_3 is an optimally competitive randomized algorithm for the 3-cache problem which is H_3-competitive. We know that $H_3 = \frac{11}{6}$. Before we give details, we give a bird's eye view of the algorithm in table form (Table 4.2) as at the end of the previous section. K_3 has six knowledge states, up to equivalence. The number of pages contained in a support configuration (we call these *active pages*) is never more than five. As before, the table shows the potential which proves its competitiveness.

Knowledge states of K_3.

1. We define $A^{a,b,c} = (abc||, [abc])$ for any three pages a, b, c.

2. We define $B^{a,b,c,d} = \left(a|bcd||, \frac{1}{3}[abc] + \frac{1}{3}[abd] + \frac{1}{3}[acd]\right)$ for any four pages a, b, c, d.

3. We define $C^{a,b,c,d} = \left(ab||cd|, \frac{1}{2}[abc] + \frac{1}{2}[abd]\right)$ for any four pages a, b, c, d.

k	r	$\{k_i\}$	$\{\lambda_i\}$	$cost$	$offset$	$\Phi(k)$	$E(\Phi(k'))$	$\Delta\Phi$	$\frac{C \cdot offset}{-\Delta\Phi - cost}$
$A^{a,b,c}$	d	$B^{d,a,b,c}$	1	1	1	0	$\frac{5}{6}$	$\frac{5}{6}$	0
$B^{a,b,c,d}$	b	$C^{a,b,c,d}$	1	$\frac{1}{3}$	0	$\frac{5}{6}$	$\frac{1}{2}$	$-\frac{1}{3}$	0
$C^{a,b,c,d}$	c	$A^{a,b,c}$	1	$\frac{1}{2}$	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	0
$B^{a,b,c,d}$	e	$D^{e,a,b,c,d}$	1	1	1	$\frac{5}{6}$	$\frac{3}{2}$	$\frac{2}{3}$	$\frac{1}{6}$
$C^{a,b,c,d}$	e	$F^{e,a,b,c,dS}$	1	1	1	$\frac{1}{2}$	$\frac{5}{4}$	$\frac{3}{4}$	$\frac{1}{12}$
$D^{a,b,c,d,e}$	b	$E^{a,b,c,d,e}$	1	$\frac{1}{2}$	0	$\frac{3}{2}$	1	$-\frac{1}{2}$	0
$F^{a,b,c,d,e}$	b	$E^{a,b,c,d,e}$	1	$\frac{1}{4}$	0	$\frac{5}{4}$	1	$-\frac{1}{4}$	0
$F^{a,b,c,d,e}$	d	$C^{a,d,b,c}$	1	$\frac{3}{4}$	0	$\frac{5}{4}$	$\frac{1}{2}$	$-\frac{3}{4}$	0
$E^{a,b,c,d,e}$	c	$A^{a,b,c}$	1	$\frac{1}{2}$	0	1	0	-1	$\frac{1}{2}$
$E^{a,b,c,d,e}$	d	$A^{a,b,d}$	1	$\frac{3}{4}$	0	1	0	-1	$\frac{1}{4}$
$D^{a,b,c,d,e}$	f	$A^{f,a,b}$ $A^{f,a,c}$ $A^{f,a,d}$ $A^{f,a,e}$ $A^{f,b,c}$ $A^{f,b,d}$ $A^{f,b,e}$ $A^{f,c,d}$ $A^{f,c,e}$ $A^{f,d,e}$	$\frac{1}{10}$ $\frac{1}{10}$ $\frac{1}{10}$ $\frac{1}{10}$ $\frac{1}{10}$ $\frac{1}{10}$ $\frac{1}{10}$ $\frac{1}{10}$ $\frac{1}{10}$ $\frac{1}{10}$	1	$-\frac{1}{5}$	$\frac{3}{2}$	0	$-\frac{3}{2}$	$\frac{101}{180}$
$E^{a,b,c,d,e}$	f	$A^{f,a,b}$	1	1	0	1	0	-1	0
$F^{a,b,c,d,e}$	f	$C^{f,a,b,c}$ $C^{f,b,a,c}$ $C^{f,c,a,b}$ $C^{f,a,d,e}$ $C^{f,b,d,e}$ $C^{f,c,d,e}$	$\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{6}$	1	$\frac{1}{6}$	$\frac{5}{4}$	$\frac{1}{2}$	$-\frac{3}{4}$	$\frac{1}{18}$

Table 4.2: Summary for K_3

4. We define $D^{a,b,c,d,e} = \left(a|bcde||, \frac{1}{6}[abc] + \frac{1}{6}[abd] + \frac{1}{6}[abe] + \frac{1}{6}[acd] + \frac{1}{6}[ace] + \frac{1}{6}[ade]\right)$ for any five pages a, b, c, d, e.

5. We define $E^{a,b,c,d,e} = \left(ab||cde|, \frac{1}{2}[abc] + \frac{1}{4}[abd] + \frac{1}{4}[abe]\right)$ for any five pages a, b, c, d, e.

6. We define $F^{a,b,c,d,e} = \left(a|bc|de|, \frac{1}{2}[abc] + \frac{1}{8}[abd] + \frac{1}{8}[abe] + \frac{1}{8}[acd] + \frac{1}{8}[ace]\right)$ for any five pages a, b, c, d, e.

Note that there are many symmetries, for example $A^{a,b,c} = A^{b,a,c} = A^{a,c,b}$.

K_3 Actions. Let K be the old knowledge state and r the request. K_3 has actions as illustrated by Figure 4.6.

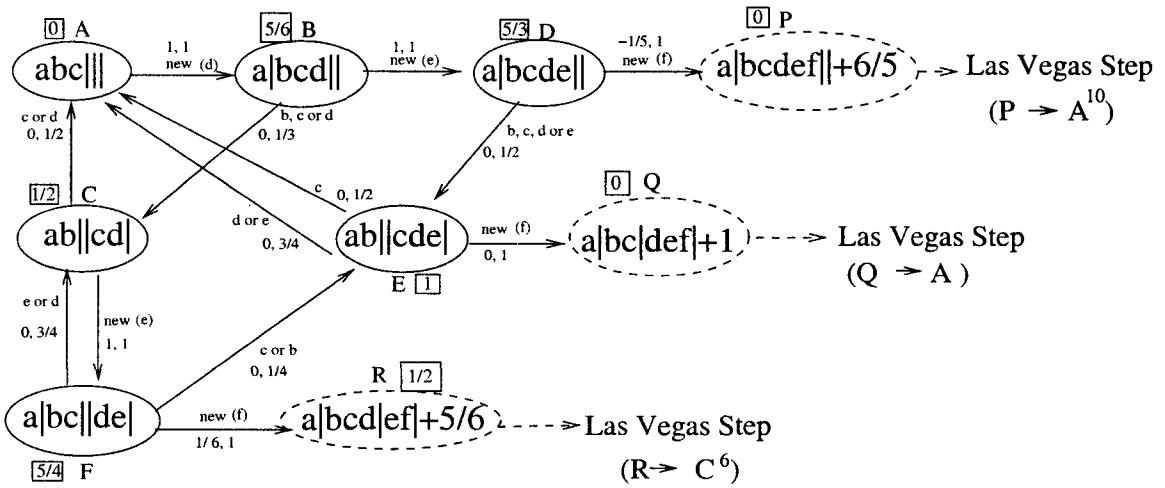

Figure 4.6: K_3 actions

Figure 4.7 illustrates the Las Vegas Step from D. From D, K_3 passes through the fleeting state P, and becomes one of ten As.

Figure 4.8 illustrates the Las Vegas Step from E. From E, K_3 passes through the fleeting state Q, and becomes one A.

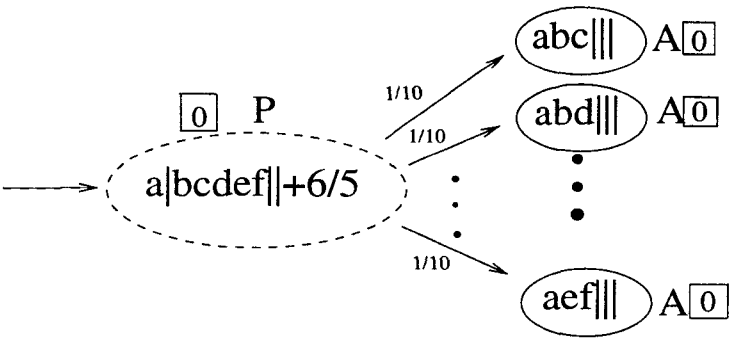

Figure 4.7: Las Vegas moves from P to A

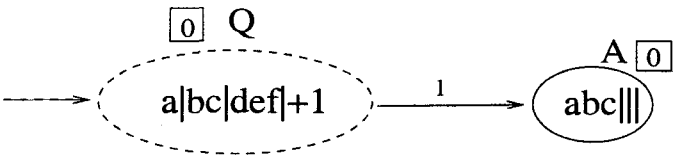

Figure 4.8: Las Vegas moves from Q to A

Figure 4.9 illustrates the Las Vegas Step from F. From F, K_3 passes through the fleeting state R, and becomes one of six Cs.

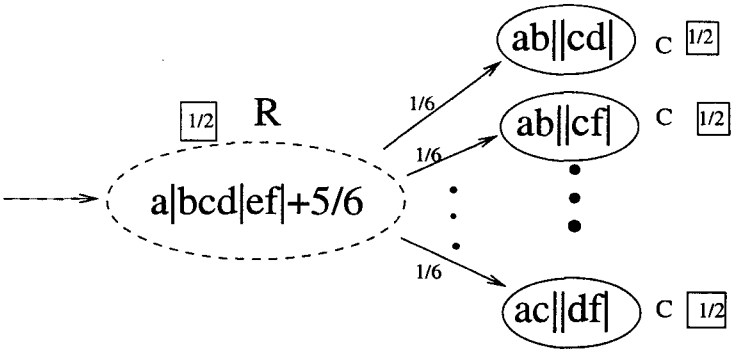

Figure 4.9: Las Vegas moves from R to C

In these figures, the number resides in the square is the potential at that state. On each arrow, there is letter(s) denoting the requested page. The word "new" means the request is a to new page. Also on each arrow, there is a pair of numbers, the first of which is the

offset, and the second of which is the cost between two steps. The dotted ovals represent the *fleeting states*, meaning they don't exist at the end of the step. They are there only for explanation.

In Figure 4.6, We use generic names for the knowledge states. Each knowledge state is denoted by an equivalent knowledge state in which the page names start with "a" and are in alphabetical order. Otherwise, the figure will be too complex to draw. In this way, equivalent knowledge states will have same bar notation.

We explain the actions of K_3 in detail as follows:

1. When $K = A^{a,b,c}$ and $r = a, b, c$, or $K = B^{a,b,c,d}$ and $r = a$, or $K = C^{a,b,c,d}$ and $r = a, b$, or $K = D^{a,b,c,d,e}$ and $r = a$, or $K = E^{a,b,c,d,e}$ and $r = a, b$, or $K = F^{a,b,c,d,e}$ and $r = a$, the new knowledge state is unchanged, the cost is zero, and the offset is zero.

2. When $K = A^{b,c,d}$ and $r = a$, the new knowledge state is $B^{a,b,c,d}$, the cost is 1, and the offset is 1.

3. When $K = B^{a,b,c,d}$ and $r = b$, the new knowledge state is $C^{a,b,c,d}$, the cost is $\frac{1}{3}$, and the offset is zero.

4. When $K = B^{b,c,d,e}$ and $r = a$, the new knowledge state is $D^{a,b,c,d,e}$, the cost is 1, and the offset is 1.

5. When $K = C^{a,b,c,d}$ and $r = c$, the new knowledge state is $A^{a,b,c}$, the cost is $\frac{1}{2}$, and the offset is zero.

6. When $K = D^{a,b,c,d,e}$ and $r = b$, the new knowledge state is $E^{a,b,c,d,e}$, the cost is $\frac{1}{2}$, and the offset is zero.

7. When $K = E^{a,b,c,d,e}$ and $r = c$, the new knowledge state is $A^{a,b,c}$, the cost is $\frac{1}{2}$, and the offset is zero.

8. When $K = E^{a,b,c,d,e}$ and $r = d$, the new knowledge state is $A^{a,b,d}$, the cost is $\frac{3}{4}$, and the offset is zero.

9. When $K = F^{a,b,c,d,e}$ and $r = b$, the new knowledge state is $E^{a,b,c,d,e}$, the cost is $\frac{1}{4}$, and the offset is zero.

10. When $K = F^{a,b,c,d,e}$ and $r = d$, the new knowledge state is $C^{a,d,b,c}$, the cost is $\frac{3}{4}$, and the offset is zero.

11. When $K = D^{b,c,d,e,f}$ and $r = a$, the new knowledge is selected among the ten knowledge states $A^{a,b,c}$, $A^{a,b,d}$, $A^{a,b,e}$, $A^{a,b,f}$, $A^{a,c,d}$, $A^{a,c,e}$, $A^{a,c,f}$, $A^{a,d,e}$, $A^{a,d,f}$, and $A^{a,e,f}$ uniformly. The cost is 1, and the offset is $-\frac{1}{6}$.

12. When $K = E^{b,c,d,e,f}$ and $r = a$, the new knowledge state is $A^{a,b,c}$. The cost is 1, and the offset is 0.

13. When $K = F^{b,c,d,e,f}$ and $r = a$, the new knowledge is selected among the six knowledge states $C^{a,b,c,d}$, $C^{a,c,b,d}$, $C^{a,d,b,c}$, $C^{a,b,e,f}$, $C^{a,c,e,f}$, and $C^{a,d,e,f}$ uniformly. The cost is 1, and the offset is $\frac{1}{6}$.

K_3 Potentials. We define a potential Φ on the knowledge states of K_3 as follows:

1. $\Phi(A^{a,b,c}) = 0$.

2. $\Phi(B^{a,b,c,d}) = \frac{5}{6}$.

3. $\Phi(C^{a,b,c,d}) = \frac{1}{2}$.

4. $\Phi(D^{a,b,c,d,e}) = \frac{5}{3}$.

5. $\Phi(E^{a,b,c,d,e}) = 1$.

6. $\Phi(F^{a,b,c,d,e}) = \frac{5}{4}$.

Verifications for K_3. Figure 4.10 proves that the minimum transportation cost for the Las Vegas Step from state D to state A is:

$$cost = 12 \cdot 1/30 + 6 \cdot 3/30 = 1$$

		$\frac{1}{6}$ bcd	$\frac{1}{6}$ bce	$\frac{1}{6}$ bcf	$\frac{1}{6}$ bde	$\frac{1}{6}$ bdf	$\frac{1}{6}$ bef
$\frac{1}{10}$	abc	1 1/30	1 1/30	1 1/30			
$\frac{1}{10}$	abd	1 1/30			1 1/30	1 1/30	
$\frac{1}{10}$	abe		1 1/30		1 1/30		1 1/30
$\frac{1}{10}$	abf			1 1/30		1 1/30	1 1/30
$\frac{1}{10}$	acd	1 3/30					
$\frac{1}{10}$	ace		1 3/30				
$\frac{1}{10}$	acf			1 3/30			
$\frac{1}{10}$	ade				1 3/30		
$\frac{1}{10}$	adf					1 3/30	
$\frac{1}{10}$	aef						1 3/30

Figure 4.10: Cost D to A

In Figure 4.10, the little numbers that reside in the circles are explained as follows:

- The sum of the numbers in the same column should be equal to the number marked outside of that column;

- The sum of the numbers in the same row should be equal to the number marked outside of that row.

The numbers outside of the table indicate the distributions of the subsequents. The other numbers inside of the table indicate the costs.

Figure 4.11 proves that the offset for the Las Vegas step from state D to state A is $-1/5$.

b|cdef||	a	abc	abd	abe	abf	acd	ace	acf	ade	adf	aef	
a|bcdef||		1	1	1	1	1	1	1	1	1	1	
abc|||		0	1	1	1	1	1	1	2	2	2	
abd|||		1	0	1	1	1	2	2	1	1	2	
abe|||		1	1	0	1	2	1	2	1	2	1	
abf|||		1	1	1	0	2	2	1	2	1	1	offset =−1/5
acd|||		1	1	2	2	0	1	1	1	1	2	
ace|||		1	2	1	2	1	0	1	1	2	1	
acf|||		1	2	2	1	1	1	0	2	1	1	
ade|||		2	1	1	2	1	1	2	0	1	1	
adf|||		2	1	2	1	1	2	1	1	0	1	
aef|||		2	2	1	1	2	1	1	1	1	0	
		$\frac{6}{5}$	$\frac{6}{5}$	$\frac{6}{5}$	$\frac{6}{5}$	$\frac{6}{5}$	$\frac{6}{5}$	$\frac{6}{5}$	$\frac{6}{5}$	$\frac{6}{5}$	$\frac{6}{5}$	

Figure 4.11: Offset D to A

In Figure 4.11, the arrows in this figure illustrate the directions of the changes. For example, the first arrow on the left hand side indicates that, when updating a, we obtain the first row which indicates the estimators after updating. The other rows except for the last one are estimators of subsequents. These subsequents are uniformly distributed. The last row indicates the average estimators of the above three rows. The difference of the first and the last rows is the *offset* we are looking for.

We want to prove

$$cost + \Delta\Phi \leq \frac{11}{6} offset$$

Expanding the above inequality, we get:

$$1 + (0 - 5/3) \leq 11/6 \cdot (-1/5)$$

which is correct. Therefore, this Las Vegas step is proved.

Figure 4.12 proves that the minimum transportation cost for the Las Vegas Step from state E to state A is:

$$cost = 1 \cdot (1/2 + 1/4 + 1/4) = 1$$

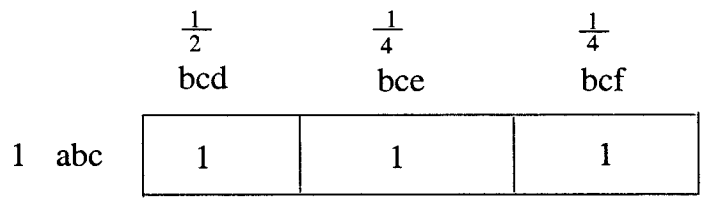

Figure 4.12: Cost E to A

The explanation of this figure is same as that for Figure 4.10.

Figure 4.13 proves that the offset for the Las Vegas step from state E to state A is 0.

The explanation of this figure is almost same as that for Figure 4.11. But, there is only one possible state in this case.

We want to prove

$$cost + \Delta\Phi \leq \frac{11}{6} offset$$

Expanding the above inequality, we get:

$$1 + (0 - 1) \leq 11/6 \cdot 0$$

	abc	abd	abe	abf	acd	ace	acf	bcd	bce	bcf	
w=bc||def|	1	1	1	1	1	1	1	0	0	0	
w^a=a|bc|def|+1	1	1	1	1	1	1	1	2	2	2	offset =0
abc|||	0	1	1	1	1	1	1	1	1	1	

Figure 4.13: Offset E to A

which is correct. Therefore, this Las Vegas step is proved.

Figure 4.14 proves that the minimum transportation cost for the Las Vegas Step from state F to state C is:

$$cost = 3 \cdot 1/6 + 4 \cdot 1/24 + 4 \cdot 1/12 = 1$$

		$\frac{1}{2}$ bcd	$\frac{1}{8}$ bce	$\frac{1}{8}$ bcf	$\frac{1}{8}$ bde	$\frac{1}{8}$ bdf
$\frac{1}{6}$	abc	1 1/6				
$\frac{1}{6}$	abd	1 1/6				
$\frac{1}{12}$	abe		1 1/24		1 1/24	
$\frac{1}{12}$	abf			1 1/24		1 1/24
$\frac{1}{6}$	acd	1 1/6				
$\frac{1}{12}$	ace		1 1/12			
$\frac{1}{12}$	acf			1 1/12		
$\frac{1}{12}$	ade				1 1/12	
$\frac{1}{12}$	adf					1 1/12

Figure 4.14: Cost F to C

The explanation of this figure is same as that for Figure 4.10.

Figure 4.15 proves that the offset for the Las Vegas step from state F to state C is $1/6$.

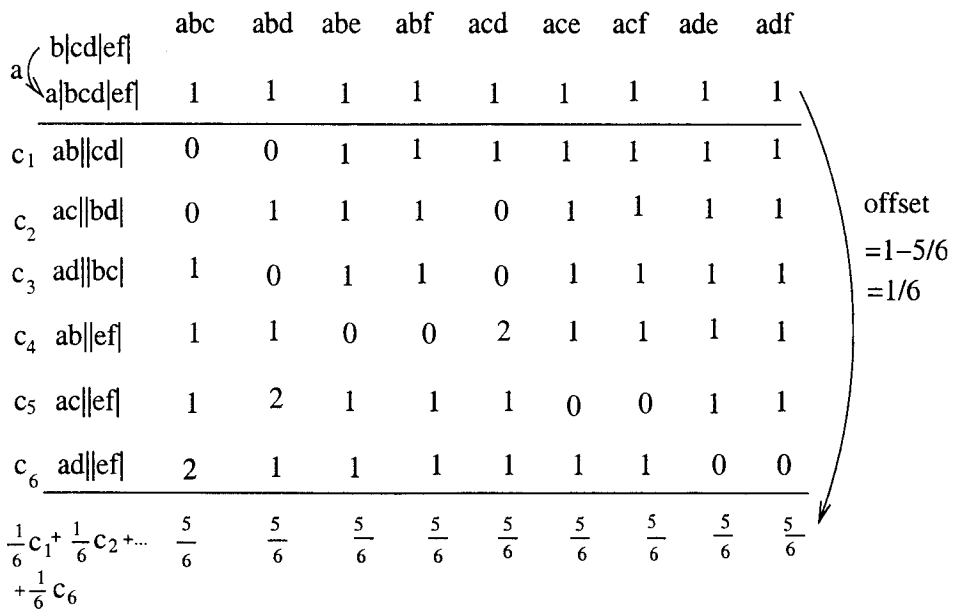

Figure 4.15: Offset F to C

The explanation of this figure is same as that for Figure 4.11.

We want to prove

$$cost + \Delta\Phi \leq \frac{11}{6} offset$$

Expanding the above inequality, we get:

$$1 + (1/2 - 5/4) \leq 11/6 \cdot (1/6)$$

which is correct. Therefore, this Las Vegas step is proved.

BIBLIOGRAPHY

[ACN96] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Proc. 4th European Symp. on Algorithms (ESA), Lecture Notes in Computer Science*, 1136:419–430, 1996.

[BG00] Yair Bartal and Edward Grove. The harmonic k-server algorithm is competitive. *Journal of the ACM*, pages 47(1):1–15, 2000.

[BLR04] W. Bein, L. L. Larmore, and R. Resichuk. Knowledge state algorithms: Randomization with limited information. *to appear*, 2004.

[CDRS93] D. Coppersmith, Peter Dolye, P. Raghavan, and M. Snir. Random walks on weighted graphs and apllications to on-line algorithms. *Journal of the ACM*, pages 40: 421–453, 1993.

[CL91] M. Chrobak and L. L. Larmore. On fast algorithms for two servers. *Jounal of Algorithms*, pages 12:607–614, 1991.

[CL92a] M. Chrobak and L. L. Larmore. Harmonic is three-competitive for two servers. *Theoretical Computer Science*, 98:339–346, 1992.

[CL92b] M. Chrobak and L. L. Larmore. The server problem and on-line games. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 7:11–64, 1992.

[DK93] X. Deng and E. Koutsoupias. Competitive implementations of parallel programs. *Proc.4th ACM-SIAM Symposium on Discrete Algorithms*, pages 455–461, 1993.

[FKL+91] A. Fiat, R. Karp, M. Luby, L. A. McGeoch, D. Sleator, and N. E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12:685–699, 1991.

[FRR90] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k-server algorithms. *Proc. 22nd IEEE Symposium on Foundations of Computer Science*, pages 454–463, 1990.

[IR91] S. Irani and R. Rubinfeld. A competitive 2-server algorithm. *Information Processing letters*, pages 39:85–91, 1991.

[Kle94] J. Kleinberg. A lower bound for two-server balancing algorithms. *Information Processing letters*, 1994.

[KP94a] E. Koutsoupias and C. Papadimitriou. Beyond competitive analysis. *Proc.35th IEEE Symp. Foundations of Computer Science(FOCS)*, pages 394–400, 1994.

[KP94b] E. Koutsoupias and C. Papadimitriou. On the k-server conjecture. *Proc.25th Symposium on Theory of Computing*, pages 507–511, 1994.

[MMS90] M. Manasse, L. A. McGeoch, and D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230, 1990.

[RS94] Prabhakar Raghavan and Marc Snir. Memory versus randomization in on-line algorithms. *IBM Journal on Research and Development*, page 38, 1994.

VITA

Graduate College
University of Nevada, Las Vegas

Qin Zhang

Home Address:
871 Cline Cellars Ave.
Las Vegas, NV 89123

Degrees:
Bachelor of Science, Mathematics, 1992
Shanghai Teacher's University, P.R.China

Thesis Title: The Randomized Server Problem

Thesis Examination Committee:
Chairperson, Dr. Wolfgang W. Bein, Ph.D.
Committee Member, Dr. Lawrence L. Larmore, Ph.D.
Committee Member, Dr. Kazem Taghva, Ph.D.
Graduate Faculty Representative, Dr. Shashi Nambisan, Ph.D.