Computational Game Theory **Spring Semester, 2003/4**

Lecture 9: Dynamics in Load Balancing

Lecturer: Yishay Mansour *Scribe: Anat Axelrod, Eran Werner*

9.1 Lecture Overview

In this lecture we consider dynamics of a load balancing system. We study the number of steps required to reach pure Nash equilibrium in a system of "selfish" and greedy players (jobs). We are interested in the convergence time to Nash equilibrium, rather than the quality of the allocation. We consider a variety of load balancing models including identical, restricted, related and unrelated machines. We compare different scheduling strategies such as allowing the largest job to move first (or smallest weight first) and compare their effect on the convergence time. Note that the discussion can be easily extended to the domain of routing problems.
The lecture is divided into two major sections. In the first we deal with migration policies that allow a single user to migrate at a time. In the second part we discuss the case where users migrate concurrently.
For the former part we consider two settings:

1. An empty system where jobs are added gradually. We will show an upper bound of $O(n)$ for this case using the Max Weight First policy.

2. A system at an arbitrary initial state. Here we begin by showing that any scheduler will converge, even in the unrelated and restricted assignment model. Afterwards, we give special treatment to the case of identical machines where we show both an upper bound of $O(n)$ using Max Weight First and a lower bound of $\Omega(n^2)$, on the other hand, using Min Weight First.

When users are allowed to migrate concurrently from overloaded to underloaded machines, we present two algorithms for the special setting of two identical machines and we conclude by showing a general algorithm for multiple identical machines.

9.2 The Model - Formal Definition

In our load balancing scenario there are m parallel machines and n independent jobs. At each step each job is assigned to exactly one machine.

- S_i - The speed of machine M_i. $S_i \in [1, S_{max}]$ where the speeds are normalized s.t. $S_{min} = 1$ and the maximum speed is S_{max}. When considering the identical or unrelated machines models, $S_i = 1$ for $1 \leq i \leq m$.

- W_j - The weight of job j. We will mainly deal with the related and identical machine models. When generalizing to the unrelated machine models we denote $w_{i,j}$ as the weight of job j when assigned on machine M_i.

- $B_i(t)$ - The set of jobs on machine M_i at time t.

- $L_i(t)$ - The load of machine M_i at time t. This is the sum of the weights of the jobs that chose M_i. $L_i(t) = \sum_{j \in B_i(t)} W_j$.

- $T_i(t)$ - The normalized load on machine M_i at time t, obtained by dividing the load of the machine with its speed. $T_i(t) = \frac{L_i(t)}{S_i}$

- $U(t)$ - The set of jobs that may decrease their experienced load at time t by migrating to another machine. A job wishes to migrate if and only if its cost (load) will strictly reduce after the migration. Note that the users (jobs) are considered to be myopic i.e. at each step they wish only to improve their state regardless of future consequences. We examine arbitrary improvement steps, and when the job selects a machine which minimizes its cost it is said to have played the best response policy.

- ***The Scheduler*** - Before migrating between machines a job needs to receive a grant from the centralized scheduler. The scheduler does not influence the selection of the target machine but merely controls the order in which the jobs migrate.

- ***Scheduling Strategies*** - The input at time t is the set of jobs $U(t)$ and the output is a job $j \in U(t)$ which is allowed to migrate at time t. When only one job is allowed to migrate at a time it is easy to define some natural selecting strategies, for example:

  - Max Weight First
  - Min Weight First
  - Arbitrary Job Next

- ***Pure Nash Equilibrium*** - A state in which no job can benefit from unilaterally migrating to another machine. For every user j on machine M_i it holds that $\forall k \;\; T_i \leq T_k + w_{k,j}$.

9.2.1 Dynamics of Migration

We consider two models in which jobs are allowed to migrate in order to improve their costs. First, we examine a scenario where at each step a single user (job), selected by the scheduler from the group of "unsatisfied" jobs, is allowed to move. In the latter model this group of jobs is allowed to migrate simultaneously.

9.3 Single Migrator

We will divide our discussion into two settings:
In 9.3.1 we start from an empty system and gradually add one job at a time.
In 9.3.2 we consider starting from an arbitrary state, then applying the scheduling policy until the system converges to equilibrium.

9.3.1 Starting from Scratch

Let the jobs enter the system in descending sorted order of weights:
$w_1 \geq w_2 \geq ... \geq w_n$. When entering the system, each job immediately selects its best response.

Claim 9.1 *The system will remain in Nash equilibrium at each step.*

Proof. By induction on the number of steps.
Trivially, the system is in Nash for the initial step of an empty system.
Assume that it is also in Nash at step $j-1$, immediately before w_j is added.
w.l.o.g job j selects machine M_1.
Obviously, all jobs on machines other than M_1 remain in equilibrium. It is enough to show that all jobs on M_1 are in equilibrium as well. Assume by contradiction that there exists a job k on M_1 which wishes to migrate to machine M_2. Then, $T_1 > T_2 + \frac{w_k}{S_2}$. Since $w_j \leq w_k$ it is also true that $T_1 > T_2 + \frac{w_j}{S_2}$, contradicting our assumption that machine M_1 was the best response for (and selected by) job j. □

Food For Thought 1 *How would you show an existence of a pure Nash equilibrium in the unrelated machines model?*

9.3.2 Starting from Something

Now the system is initially in an arbitrary state and we seek to reach equilibrium by a sequence of steps in which only one job, selected by the scheduler, can move. This model is called **E**lementary **S**tepwise **S**ystem (ESS).

Unrelated Machines

First we will show that even for the general model of unrelated machines with restricted assignments, convergence to equilibrium is promised. Furthermore, we establish this result without relying on best response moves but only general improvement steps. We obtain this result by identifying a potential function and showing that the potential strictly decreases after each improvement step. Consider the sorted vector of machine loads. We show that defining a lexicographic order on those sorted vectors provides us with a potential function. Using lexicographic order, vectors are compared by their first unequal component. One vector is called lexicographically larger than the other if its first non equal component is larger than its corresponding counterpart in the second vector.

Claim 9.2 *The sorted lexicographic order of the machine loads decreases when a job migrates via improvement.*

Proof. Improvement influences only two components of the sorted vector: one corresponding to the load on the machine that the job has left while the other to the machine the job has joined. The load on the machine the job joined cannot be higher than the load it experienced on the machine it had left (otherwise would it have moved?). Formally, if the job migrated from machine M_i to machine M_j, we have $L_i(t) > L_j(t+1)$. Additionally, $L_i(t) > L_i(t+1)$ since the job had left. Thus, if $L_i(t)$ was the k^{th} component in the sorted vector of loads then the respective component in the new vector cannot be larger and the new vector is considered lexicographically smaller. □

Since we have shown that improvement always decreases the potential, this gives us an upper bound on the convergence time which is equal to the number of different sorted loads vectors (trivially bounded by the number of configurations) - m^n.

Corollary 9.3 *For any ESS strategy, the system of unrelated machines with restricted assignments reaches Nash equilibrium in at most m^n steps.*

This result was obtained for any scheduler. We now wish to investigate how specific scheduling strategies may give better bounds. Unfortunately, we can provide results only in more specific models.

Identical Machines - Upper Bound

In the identical machines model with unrestricted assignments we use the Max Weight First (MWF) scheduling strategy to establish an upper bound of O(n) on the convergence time. The scheduler always selects a job $J(t) = \arg\max_{j \in U(t)}\{w_j\}$, i.e. the largest unsatisfied job. We assume that the jobs play their best response.

Claim 9.4 *Suppose that a job J has migrated to machine M which is its best response at time t_0. If J wishes to migrate again at a later time $t_2 > t_0$ then another job J' such that $w_{J'} > w_J$ must have joined machine M at time t_1, $t_0 < t_1 \leq t_2$.*

Proof. Since J wishes to migrate, one of two things must have happened at some earlier time t_1:

1. Job J' has moved from machine $M_{J'}$ to a different machine $M' \neq M$.
2. Job J' has joined job J at machine M.

1. Obviously, Job J does not wish to deviate to M' since it did not wish to do so prior to the move of J', and now the load on M' has only increased.
   Fortunately, job J has also no desire to move to $M_{J'}$ even though its load decreased. The reason for this is:
   $T_{M_{J'}}(t_1 - 1) > T_{M'}(t_1 - 1) + w_{J'} = T_{M'}(t_1)$ (J' shifted from $M_{J'}$ to M').
   ⇓

$T_{M_{J'}}(t_1) = T_{M_{J'}}(t_1 - 1) - w_{J'} > T_{M'}(t_1 - 1)$.
The resulting load on $M_{J'}$ is higher than the load that was on M' at time $t_1 - 1$. Since M' was not a best response for J at that time $M_{J'}$ certainly isn't now.

2. In this case we want to show that J will want to leave M only if $w_{J'} > w_J$. Assume by contradiction that $w_{J'} \leq w_J$. Let M' be the machine that J wishes to move to. Thus, $T_M(t_1) = T_M(t_1 - 1) + w_{J'} > T_{M'}(t_1 - 1) + w_J \geq T_{M'}(t_1 - 1) + w_{J'}$ Hence, $T_M(t_1) \geq T_{M'}(t_1 - 1) + w_{J'}$ contradicting the assumption that J' played his best response at time t_1.

□

Theorem 9.5 *The Max Weight Job strategy with best response policy, for a system of identical machines with unrestricted assignments, reaches Nash equilibrium in at most n steps.*

Proof. By claim 9.4 once a job has migrated to a new machine, it will never leave it unless a larger job arrives. Since the jobs arrive in descending weight order (MWF) only smaller jobs may arrive at subsequent steps. Therefore each job stabilizes after its first migration, and the theorem follows. □

Identical Machines - Lower Bound

We now present a lower bound using the Min Weight strategy. We demonstrate a setting with two identical machines and n jobs. Later, we explain the idea for generalizing the result to m machines.
Consider the following scenario. There are $\frac{n}{2}$ classes of jobs $C_1, ..., C_{\frac{n}{2}}$. Each class C_k contains exactly two jobs with weights $w_k = 3^{k-1}$. Notice that a job in C_k has weight equal to the total weight of all of the jobs in the first $k-1$ classes plus 1. Initially all jobs are located at the first machine. The scheduling process is divided into phases. At each phase k all jobs from classes C_1 to C_k except one job from C_k move from one machine to another. Thus, the duration of phase k is $2k - 1$. The scheduling consists of phases $\frac{n}{2}, ..., 1$, since after each phase the two jobs of the heaviest class are set on two different machines and the scheduling can be regarded as continuing recursively with one less class. At equilibrium each machine will occupy exactly one job from each class. The convergence time is given by the recursive formula:

$$f(r) = f(r-1) + 2r - 1$$

for $r = \frac{n}{2}$ and this is clearly $\Omega(n^2)$.
For the general lower bound we have $m = k + 1$ machines and n jobs. We divide the jobs into k weight classes of size $\frac{n}{k}$. The weight of a job in class j is chosen such that it is larger than the sum of all the job weights of earlier classes: if we denote by w_j the weight of a job $\in C_j$ then, $w_j > \sum_{i=1}^{j-1} \frac{n}{k} w_i$. Initially all jobs from class i are assigned to machine i and machine 0 is empty. Applying the Min weight scheduling consists of phases such that before a job from class k is granted to move all the lighter

jobs are equally distributed between machines $[0, k-1]$. However, as soon as the first heavy job is allowed to move it causes all of the lighter jobs to shift among the other machines, not containing a heavy weight job, and this process continues recursively, as the scheduling is from the lightest job first. This example gives a lower bound of $\Omega(\frac{(\frac{n}{k})^k}{2(k!)})$

Food For Thought 2 *Could you think of a natural scheduling strategy that would quickly converge to equilibrium in the related (or maybe even unrelated) model ?*

9.4 Concurrent Jobs Migration

In this section we examine the setting of n identical users i.e. $\forall j \quad w_j = 1$. For simplicity we also deal with identical machines i.e. $\forall i \quad S_i = 1$. With contrast to the previous model, several users can now move concurrently. Consequently, equilibrium is no longer promised, as the system may oscillate. For example, consider a two machine setting, where all jobs are initially placed on one of the machines. Since they all want to migrate at once they will find themselves all in a similar situation but on the second machine. We overcome this problem by introducing randomization into the decision if a job should migrate.
The system is at Nash equilibrium at time t if for every pair of machines M_i, M_j it holds that $L_i(t) \leq L_j(t) + 1$.

We will divide our discussion into two:
In 9.4.1 we start from the specific case of two identical machines.
In 9.4.2 we extend our discussion and deal with multiple identical machines.

9.4.1 Two Identical Machines

The Balance Algorithm

First we consider the case of only two identical machines M_1, M_2 with load functions $L_1(t), L_2(t)$ respectively. We use two variables $over, under \in \{M_1, M_2\}$ to identify at each step which is the more/less loaded machine i.e. its load is above/under the average load ($\frac{n}{m}$ or $\frac{n}{2}$ in this case) in the system. Obviously, $L_{over}(t) \geq L_{under}(t)$. Let $d_t = |L_1(t) - L_2(t)|$.
The BALANCE algorithm moves every job $j \in B_{over}(t)$ to the other machine with probability $\frac{d_t}{2L_{over}(t)}$. The idea is to achieve an equal expected load on both machines at the end of each step: $E[L_1(t+1)|L_1(t), L_2(t)] = \frac{n}{2}$. So, each job on *over* moves with probability $\frac{d_t}{2L_{over}(t)}$ and we get $L_{over}(t) \cdot \frac{dt}{2L_{over}(t)} = \frac{dt}{2}$ expected movements, as desired.
We would like to show that after expected $O(\log \log n)$ steps the system will reach an equilibrium. For the proof we use the following lemma.

Lemma 9.4.1 ***(Chernoff)*** *Let $z_1, ..., z_n$ be independent random binary variables and $Z = \sum_i z_i$, where $p = \sum_{i=1}^{n} \frac{E[z_i]}{n}$ and $\widehat{p} = \frac{1}{n}\sum_{i=1}^{n} z_i$. Then,*

$$(1) \quad P[p \leq \widehat{p} + \sqrt{\tfrac{2p\ln(1/\delta)}{n}}] \geq 1-\delta \qquad p \in [0,1]$$

$$(2) \quad P[p \geq \widehat{p} - \sqrt{\tfrac{3p\ln(1/\delta)}{n}}] \geq 1-\delta \qquad p \in [\tfrac{\ln(1/\delta)}{3n}, 1]$$

$$(3) \quad P[p \geq \widehat{p} - \tfrac{2\ln(1/\delta)}{n}] \geq 1-\delta \qquad p \in [0, \tfrac{\ln(1/\delta)}{3n}]$$

Theorem 9.6 *The BALANCE algorithm terminates within expected $O(\ln\ln n)$ steps.*

Proof. Let k be an upper bound on the number of steps of the BALANCE algorithm until it reaches Nash equilibrium. We divide our proof into two stages:

1. $dt > 3\ln(\frac{1}{\delta'})$
2. $dt \leq 3\ln(\frac{1}{\delta'})$

where $\delta' = \frac{\delta}{2k}$ and δ indicates the probability of the algorithm failing to end within k time steps.

First we show that the first stage terminates with probability $1 - \frac{\delta}{2}$ within $O(\ln\ln n)$ steps. By lemma 9.4.1 (2) for every step $t \leq k$, $d_{t+1} \leq \sqrt{3d_t\ln(\frac{1}{\delta'})}$ with probability $1 - \frac{\delta'}{2}$: represent each job on *over* as a binary random variable with $P(1) = \frac{d_t}{2L_{over}(t)}$. Thus, $p = \sum_{i=1}^{n} \frac{E[z_i]}{n} = \frac{d_t}{2L_{over}}$ and $\widehat{p} = \frac{1}{n}\sum_{i=1}^{n} z_i$ is the ratio of jobs that were actually moved in this step.

Since $d_1 \leq n$ we get

$$\begin{aligned}
d_{t+1} &\leq \sqrt{3d_t\ln(\tfrac{1}{\delta'})} \\
&= \sqrt{d_t}\cdot\sqrt{3\ln(\tfrac{1}{\delta'})} \\
&\leq \sqrt{\sqrt{3d_{t-1}\ln(\tfrac{1}{\delta'})}}\cdot\sqrt{3\ln(\tfrac{1}{\delta'})} \\
&= \sqrt{\sqrt{d_{t-1}}}\cdot\sqrt{\sqrt{3\ln(\tfrac{1}{\delta'})}}\cdot\sqrt{3\ln(\tfrac{1}{\delta'})} \\
&\vdots \\
&\leq n^{\frac{1}{2^{t+1}}}\cdot 3\ln(\tfrac{1}{\delta'})
\end{aligned}$$

For $t = O(\ln\ln n)$, $n^{\frac{1}{2^t}} = O(1)$ and therefore, $d_t \leq 3\ln(\frac{1}{\delta'})$ and the first stage terminates within $O(\ln\ln(n))$ steps. This situation remains with high probability until the k^{th} step.

In the second stage exactly $\frac{d_t}{2}$ jobs will be displaced in one step with probability $O(\frac{1}{\sqrt{\log(k)}})$, so the expected number of steps will be $O(\sqrt{\log k})$.
Summing the two stages we have $k = O(\ln \ln n + \sqrt{\log k}) = O(\ln \ln n)$. □

Unfortunately, the BALANCE algorithm does not assure that players will indeed act according to their best response at each step. For example, consider the setting in figure 9.1 at time t. What is the point of view of a single user on *over* (M_1)? Each user on M_1 will migrate to M_2 with probability $\frac{d_t}{2L_{over}(t)} = \frac{200}{800} = \frac{1}{4}$. Hence, for a single user on M_1:

- The expected load on M_1 without him is $(400-1) - \frac{1}{4}(400-1) = 300 - \frac{3}{4}$.
- The expected load on M_2 is $200 + \frac{1}{4}(400-1) = 300 - \frac{1}{4}$.

and the user prefers to remain on M_1. This provides an incentive for users to "cheat" and deviate from the joint strategy.
We now try to establish an algorithm which prevents such behavior.

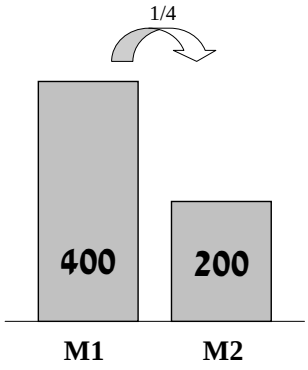

Figure 9.1:

The NashTwo Algorithm

Consider any stage before reaching Nash equilibrium where $L_1 = k+d$ and $L_2 = k-d$ i.e. $d_t = 2d$. We examine w.l.o.g a single user on L_1. We want to define the probability of migration p such that a single user will see an expected identical load on both machines (when excluding himself):
$(L_1 - 1)(1-p) = L_2 + (L_1 - 1)p$
$\Updownarrow$
$p = \frac{(L_1 - L_2) - 1}{2(L_1 - 1)} = \frac{d_t - 1}{2(L_1 - 1)}$

The NashTwo algorithm moves jobs from *over* to *under* with probability p.

Lemma 9.4.2 *At every stage Algorithm NashTwo is a Nash migration policy.*

Proof. Again we compare the loads on the two machines when excluding one user on the overloaded machine.
$L_1 = (k+d-1)(1-\frac{2d-1}{2(k+d)-2}) = n - \frac{1}{2}$
$L_2 = k - d + (k+d-1)(1-\frac{2d-1}{2(k+d)-2}) = n - \frac{1}{2}$
Therefore, each user on the overloaded machine cannot gain by deviating from the joint strategy. Users on the underloaded machine, like before, can obviously only lose by rerouting to the overloaded machine. □

It can be shown (in a different scribe...) that NashTwo converges at a similar rate as that of BALANCE.

9.4.2 Multiple Identical Machines

In the final subsection we extend the results to multiple identical machines. We seek a goal of dividing the load equally among the machines, i.e. to achieve a load of $\overline{L} = \frac{n}{m}$ which we assume to be an integer value. For each machine M_i, define $d_i(t) = L_i(t) - \overline{L}$, the difference between the current and the average load on machine M_i. Next, for each time t partition the machines into two disjoint sets,

- $Under(t) = \{M_i | d_i(t) < 0\}$
- $Over(t) = \{M_i | d_i(t) \geq 0\}$

We define $d_t = \sum_{i \in Over(t)} d_i(t)$.
Using these definitions the MultipleBALANCE algorithm proceeds as follows. Each user j on machine $M_i \in Over(t)$ determines whether it should move with probability $\frac{d_i(t)}{L_i(t)}$, and if so, selects its target underloaded machine $M_k \in Under(t)$ with probability $\frac{|d_k(t)|}{d_t}$.

Lemma 9.4.3 *(without proof) Let $Z_1, ..., Z_n$ be an i.i.d random binary variables with $P(Z_i = 1) = p$ and $Z = \sum_i Z_i$. Then, $P(Z = \lceil pn \rceil) \geq \frac{1}{e}\frac{1}{\sqrt{2\pi\lceil pn \rceil}}$. If $pn = q$ is an integer then, $P(Z = q) \geq \frac{1}{\sqrt{2\pi q}}$.*

Theorem 9.7 *The MultipleBALANCE algorithm convergence within expected $O(\log\log n + \log^2 m)$ steps.*

Proof. Let k be an upper bound on the number of steps of the MultipleBALANCE algorithm until it reaches Nash equilibrium. We divide our proof into two stages:
1. While there exists a machine M_i such that $d_i(t) > 3\ln(\frac{1}{\delta'})$
2. $\forall i$, $d_i(t) \leq 3\ln(\frac{1}{\delta'})$
where $\delta' = \frac{\delta}{3mk}$ and δ as before.
Similarly to the proof of the convergence time of the BALANCE algorithm, we get that for any machine M_i and time t, $d_i(t+1) \leq \sqrt{3d_k(t)\ln(\frac{1}{\delta'})}$. As before, using the fact that $d_i(1) \leq n$ and the mentioned computations will provide us the bound of $O(\ln\ln n)$ on the expected number of steps in the first stage.

During all of the second phase $\forall i$ it holds that $d_i(t) \leq 3\ln(\frac{1}{\delta'})$. Denote the number of unbalanced machines at time t as $Unbalanced(t)$. Note that $Unbalanced(t)$ can only decrease over time since once a machine is balanced it won't be changed.
We examine substages of the second phase. First we deal with the system when $Unbalanced(t) \geq \beta \log^{1.5}(\frac{1}{\delta'})$ with $\beta > 0$ a constant which will be specified from the proof. By lemma 9.4.3, since $d_i(t) \leq 3\ln(\frac{1}{\delta'})$, each unbalanced machine terminates in one step with probability $q = \Theta(\frac{1}{\sqrt{\ln(\frac{1}{\delta'})}})$. According to the standard properties of a binomial distribution, the expected number of machines in $Under(t)$ is at most $\frac{1}{2}Unbalanced(t-1)$. Since machines become balanced with probability $q < 0.01$, the expected number of machines in $Over(t)$ is $E[|Over(t)|] \geq 0.49Unbalanced(t-1)$. Let $O_i(t)$ be a variable indicating whether machine $M_i \in Over(t)$, thus we can write that $|Over(t)| = \sum_{i=1}^{m} O_i(t)$. The variables $O_i(t)$ are negatively associated and therefore we can apply the Chernoff bound on the number of overloaded machines as follows.

$$\begin{aligned} |Over(t)| &\geq 0.49Unbalanced(t-1) - \sqrt{3Unbalanced(t-1)\ln\tfrac{1}{\delta'}} \\ &\geq 0.48Unbalanced(t-1) \end{aligned}$$

for $Unbalanced(t-1) \geq \gamma(\log(\frac{1}{\delta'}))$, for some constant $\gamma > 0$ (with probability $1-\delta'$). The expected number of machines that become balanced at time t is at least $q|Over(t-1)|$. Note that each overloaded machine ($\in Over(t-1)$) becomes balanced with probability q independently and therefore we can apply the Chernoff bound on the number of new balanced machines as follows.

$$\begin{aligned} Unb(t) - Unb(t-1) &\geq 0.48Unb(t-1) - \sqrt{3qUnb(t-1)\log\tfrac{1}{\delta'}} \\ &\geq 0.47qUnb(t-1) \end{aligned}$$

for $Unbalanced(t-1) \geq \beta(\log^{1.5}(1/\delta'))$, for sufficiently large constant β (with probability $1-\delta'$). Consequently, after $O(\frac{\log m}{q}) = O(\log m\sqrt{\log(1/\delta')})$ the first substage terminates. Now, when $Unbalanced(t) < \beta\log^{1.5}(1/\delta')$, the number of unbalanced machines is only $O(\log^{1.5}(1/\delta'))$. Recall the second stage in the proof of the convergence time of the BALANCE algorithm. In a similar manner, it implies that after $O(\log^{1.5}(1/\delta')\cdot\sqrt{\log(1/\delta')}) = O(\log^2(1/\delta'))$ the remaining machines will be balanced. To conclude, we sum up our results to get,

$$\begin{aligned} k &= O(\log\log n + \log m\sqrt{\log(1/\delta')} + \log^2(1/\delta')) \\ &= O(\log\log n + \log m\sqrt{\log(mk/\delta)} + \log^2(mk/\delta)) \\ &= O(\log\log n + \log^2(m/\delta)) \end{aligned}$$

□

Note 9.8 *The component* $\log^2$ *can be improved to* $\log^{1.5}$*. Also, we can get a bound on the expected time of* $O(\log\log n + \log^{1.5} m)$ *(without dependency on* δ*).*

9.5 The End

Nothing lasts forever.