Probabilistic Recurrence Relations for Parallel Divide-and-Conquer Algorithms

Marek Karpinski [1]

Wolf Zimmermann [2]

TR-91-067

December, 1991

Abstract

We study two probabilistic recurrence relations that arise frequently in the analysis of parallel and sequential divide-and-conquer algorithms (cf. [Karp 91]). Suppose a problem of size x has to be solved. In order to solve it we divide it into subproblems of size $h_1(x), \ldots, h_k(x)$ and these subproblems are solved recursively. We assume that $size(h_i(z))$ are random variables. This occurs if either the *break up* step is randomized or the instances to be solved are drawn from a probability distribution. The running time $T(z)$ of a parallel algorithm is therefore determined by the maximum of the running times $T(h_i(z))$ of the subproblems while the sequential algorithm is determined by the sum of the running times of the subproblems. We give a method for estimating (*tight*) *upper bounds* on the probability distribution of $T(x)$ for these two kinds of recurrence relations, answering the open questions in [Karp 91].

[1] Dept. of Computer Science, University of Bonn, 5300 Bonn 1, and the International Computer Science Institute, Berkeley, California. Supported in part by Leibniz Center for Research in Computer Science, by the DFG, Grant KA 673/4-1 and by the SERC Grant GR-E 68297

[2] Dept. of Computer Science, University of Karlsruhe, and International Computer Science Institute, Berkeley, California.

1 Introduction

Two classes of probabilistic recurrence relations occur frequently in the analysis of divide-and-conquer algorithms cf. [Ka 91]. A problem instance z of size x is divided into subproblems $h_1(z), \ldots, h_k(z)$. On a sequential computer the subproblems has to be solved one after another. Therefore the running time of a divide- and-conquer algorithm is the solution of a recurrence of the form

$$T(z) = a(z) + T(h_1(z)) + \cdots + T(h_k(z)) \tag{1}$$

where $a(z)$ is the time required by the break up of the instance z into the subproblems and the use of the solutions of the subproblems to the solution of the original problem. On a parallel computer all the subproblems can be solved at the same time on different processors. Therefore the running time is the solution of a recurrence of the form

$$T(z) = a(z) + \max(T(h_1(z)), \ldots, T(h_k(z))) \tag{2}$$

We consider the case where the $h_i(z)$ are random variables. In this case the running time $T(z)$ is also a random variable, and we estimate bounds on its probability distribution for both cases, (1) and (2). This work is an extension of Karp's results [Ka 91] and solves the open questions of [Ka 91]. Throughout the paper we use the following notations and assumptions:

- $a(z)$ is a function on the size of z and does not depend on the distribution of z. We will denote this fact by writing $a(x)$ instead of $a(z)$[3]
- The $size(h_i(z))$ are random variables satisfying $0 \leq size(h_1(z)) + \cdots + size(h_k(z)) \leq size(z)$.

[3]We always use x to denote $size(z)$.

- $E[size(h_i(z))] \leq m_i(size(z)) \leq size(z)$ for all $1 \leq i \leq k$.
- $m_i(x)/x$ is non-decreasing.

These assumptions are quite general. In particular, no specific assumption on the distribution of $(h_1(x), \ldots, h_k(x))$ is necessary. These weak assumptions make our results practical, because the distribution of $(h_1(x), \ldots, h_k(x))$ is usually unknown.

In section 2 we consider probabilistic recurrences of type (2), and in section 3, probabilistic recurrences of type (1). This paper has similar aims as [Ka 91]. Among others, we provide a *cook book* methodology for analyzing divide-and-conquer algorithms in some general situations. Observe that from the estimate of the probability distributions for $T(x)$ an upper bound for the expected value of $T(x)$ can be obtained directly.

2 Probabilistic Recurrence Relations with Maxima

We consider first the probabilistic recurrence relations of the form (2), which usually occur in the time analysis of parallel divide-and-conquer algorithms. These kind of recurrences are also useful in the analysis of space complexity of sequential algorithms. Let $u(x)$ be the least positive solution of the (deterministic) recurrence relation $T(x) = a(x) + \max(T(m_1(x)), \ldots, T(m_k(x)))$. This solution is usually determined by small instances of the problem which can be solved trivially (for example instances of size 1 or 0).

The main result is:

Theorem 1 *Let $a(x)$ be continuous, non-decreasing, and strictly increasing on $\{x|a(x) > 0\}$. Also let the $m_i(x)$ be strictly increasing. Then, for every instance z of size x and every positive integer n, we have*

$$\Pr[T(z) \geq u(x) + n\ a(x)] \leq \left(\frac{m_1(x) + \cdots + m_k(x)}{x}\right)^n$$

As an immediate corollary, we get under the same conditions as Theorem 1 the following upper bound for the expected value of $T(z)$:

Corollary 2

$$E(T(z)) \leq u(x) + ((\log(x) - \log(m_1(x) + \cdots + m_k(x))) \cdot a(x))^{-1}$$

Proof: Use the formula $E[Y] = \int_0^\infty \Pr[Y \geq y]\mathrm{d}y$ for non-negative random variables Y. □

The proof of Theorem 1 follows a similar line as in [Ka 91]. We shall also use the following Lemma:

Lemma 3 (R. Karp [Ka 91]) *Let X be a random variable with range $[0, x]$ for some x. Let f be a non-negative real-valued integrable function such that there is $b > 0$ with:*

- *$f(x)/x$ is non-decreasing on $[0, b]$.*
- *For all $y \geq b$: $f(y) = 1$.*
- *$E[X] \leq \min(b, x)$*

Then $E[f(X)] \leq \dfrac{E[X]\ f(\min(b, x))}{\min(b, x)}$.

We are going to prove the following stronger version of Theorem 1:

Theorem 4 *For all $r > 0$ and all instances z define $s_r(z) = \Pr[T(z) \geq r]$. Define a function $d_r(x)$ as follows:*

$$d_r(x) = \begin{cases} \left(\frac{m_1(x)+\cdots+m_k(x)}{x}\right)^{\lceil \frac{r-u(x)}{a(x)} \rceil} \frac{x}{u^{-1}(r-a(x)\ \lceil (r-u(x))/a(x) \rceil)} & \text{if } u(x) < r \\ 1 & \text{otherwise} \end{cases}$$

Then $s_r(z) \leq d_r(\mathrm{size}(z))$.

Proof: The outline of this proof is similar to that of Theorem 7 in [Ka 91]. The proof is by induction using as hypothesis the claim in 5. In order to prove the induction step, we apply another induction.

It holds by basic laws of probability theory:

$$\begin{aligned} s_r(z) &= \Pr[\max_{1 \leq i \leq k} T(h_i(z)) \geq r - a(x)] \\ &\leq \sum_{i=1}^{k} \Pr[T(h_i(z)) \geq r - a(x)] \\ &= \sum_{i=1}^{k} E[s_{r-a(x)}(h_i(z))] \end{aligned}$$

Define a sequence $\{s_r^j\}$ as follows:

$$\begin{aligned} s_r^0(z) &= \begin{cases} 1 & \text{if } r \leq u(size(z)) \\ 0 & \text{otherwise} \end{cases} \\ s_r^{j+1}(z) &= \sum_{i=1}^{k} E[s_{r-a(x)}^j(h_i(z))] \end{aligned}$$

Then $s_r(z) = \sup_j s_r^j(z)$. We show now by induction, that for all j we have $s_r^j(z) \leq d_r(size(z))$. Observe first, that $d_r(x)$ is integrable and $d_r(x)/x$ is non-decreasing, because $u^{-1}(x)$, $a(x)$, and $m_i(x)/x$ are non-decreasing.

BASE CASE: $j = 0$. Straightforward.

INDUCTIVE CASE: It holds

$$\begin{aligned} s_r^{j+1}(z) &= \sum_{i=1}^{k} E[s^j_{r-a(x)}(h_i(z))] \\ &\leq \{\text{Induction hypothesis }\} \\ &\quad \sum_{i=1}^{k} E[d_{r-a(x)}(size(h_i(z)))] \end{aligned}$$

Three cases must be distinguished depending on the value of $u(x)$:

1ST CASE: $u(size(z)) > r$. Then immediatly $d_r(size(z)) = 1$ and the claim becomes trivial.

2ND CASE: $r - a(size(z)) \leq u(size(z)) < r$. Then by Lemma 3 and the induction hypothesis:

$$E[d_{r-a(x)}(size(h_i(z)))] \leq \frac{E[size(h_i(z))]}{u^{-1}(r - a(x)\ \lceil (r - u(x))/a(x) \rceil)}$$

Observe that $\lceil (r - u(x))/a(x)) \rceil = 1$ in this case. Therefore

$$s_r^{j+1}(z) \leq \frac{m_1(x) + \cdots + m_k(x)}{u^{-1}(r - a(x))} = d_r(size(z))$$

3RD CASE: $u(size(z)) < r - a(size(z))$. Then by Lemma 3 and the induction hypothesis of the induction over j:

$$E[d_{r-a(x)}(size(h_i(z)))] \leq \frac{m_i(x)}{x}\ d_{r-a(x)}(x)$$

Thus:

$$s_r^{j+1}(z) \leq \frac{m_1(x) + \cdots + m_k(x)}{x} d_{r-a(x)}(x) = d_r(size(z))$$

and our Theorem 4 follows. □

3 Probabilistic Recurrence Relations with Sums

Here, we consider recurrence relations of the form

$$T(z) = a(size(z)) + \sum_{i=1}^{k} T(h_i(z))$$

where the h_i satisfy the same property as in the previous section. We assume that $k \geq 2$. For $k = 1$ the results of [Ka 91] can be applied.

This kind of recurrence relations occurs for example in the analysis of sequential divide-and-conquer algorithms. The result has a more complicated form than the previous one although the proof follows almost the same line.

Let $u(x)$ be the smallest positive solution of the (deterministic) recurrence

$$T(x) = a(x) + \sum_{i=1}^{k} T(m_i(x))$$

Then the following (tight) estimate can be obtained:

Theorem 5 *For all $r > 0$ and all instances z define $s_r(z) = \Pr[T(z) \geq r]$. Define a function $d_r(x)$ as follows:*

$$d_r(x) = \begin{cases} \left(\frac{m_1(x)+\cdots+m_k(x)}{x}\right)^{f_k(r,x)} \frac{x}{u^{-1}(k^{-f_k(r,x)}\ (r+\frac{1}{k-1}\ a(x))-\frac{1}{k-1}\ a(x))} & \text{if } u(x) < r \\ 1 & \text{otherwise} \end{cases}$$

where

$$f_k(r,x) = \left\lceil \log_k \left(\frac{r + \frac{1}{k-1}\ a(x)}{u(x) + \frac{1}{k-1}\ a(x)} \right) \right\rceil$$

Then $s_r(z) \leq d_r(\text{size}(z))$.

Proof: The proof is by induction on r and uses as induction hypothesis the claim of Theorem 5. By basic laws of probability we get:

$$s_r(z) \quad = \quad \Pr[T(z) \geq r]$$

$$
\begin{aligned}
&= \Pr\left[\sum_{i=1}^{k} T(h_i(z)) \geq r - a(x)\right] \\
&\leq \Pr[\max_{i=1}^{k} T(h_i(z)) \geq k^{-1}\ (r - a(x))] \\
&\leq \sum_{i=1}^{k} \Pr[T(h_i(z)) \geq k^{-1}\ (r - a(x))] \\
&= \sum_{i=1}^{k} \mathrm{E}[s_{k^{-1}\ (r-a(x))}(h_i(z))]
\end{aligned}
$$

The application of the inequality

$$
\sum_{i=1}^{k} T(h_i(z)) \leq k\ \max_{i=1}^{k} T(h_i(z))
$$

in the above estimates leads to a similar formula as in the proof of Theorem 4. Now we proceed as in the proof of Theorem 4 and define a sequence $\{s_r^j(z)\}$ as follows:

$$
\begin{aligned}
s_r^0 &= \begin{cases} 1 & \text{if } r \leq u(x) \\ 0 & \text{otherwise} \end{cases} \\
s_r^{j+1}(z) &= \sum_{i=1}^{k} \mathrm{E}[s^j_{k^{-1}\ (r-a(x))}(h_i(z))]
\end{aligned}
$$

Clearly, with these definitions it is $s_r(z) = \sup_j s_r^j(z)$ and it is therefore sufficient to prove that for all j and r it is $s_r^j(z) \leq d_r(size(z))$. This is done by induction on j: Base Case: $j = 0$. Straightforward.

Inductive Case: It is

$$
\begin{aligned}
s_r^{j+1}(z) &= \sum_{i=1}^{k} \mathrm{E}[s^j_{k^{-1}\ (r-a(x))}(h_i(z))] \\
&\leq \{\text{InductionHypothesis}\} \\
&\quad \sum_{i=1}^{k} \mathrm{E}[d_{k^{-1}\ (r-a(x))}(size(h_i(z)))]
\end{aligned}
$$

Now we apply Lemma 3. Observe that the function $d_r(x)$ satisfies the preconditions of Lemma 3. Then three cases, depending on the value of $u(x)$ have to be distinguished:

1ST CASE: $u(x) \geq r$. Then $d_r(x) = 1$ and the claim is trivial.

2ND CASE: $k^{-1}\ (r - a(x)) \leq u(x) < r$. Then

$$0 < \left\lceil \log_k \frac{r + \frac{1}{k-1}\ a(x)}{u(x) + \frac{1}{k-1}\ a(x)} \right\rceil \leq \left\lceil \log_k \frac{k\ u(x) + \frac{k}{k-1}\ a(x)}{u(x) + \frac{1}{k-1}\ a(x)} \right\rceil = 1$$

Hence $f_k(r, x) = 1$. The application of Lemma 3 yields then:

$$\mathrm{E}[d^j_{k^{-1}\ (r-a(x))}(size(h_i(z)))] \leq \frac{\mathrm{E}[size(h_i(z))]}{u^{-1}(k^{-1}\ (r - a(x))})$$

and therefore

$$s^{j+1}_r(z) \leq \frac{m_1(x) + \cdots + m_k(x)}{x}\ \frac{x}{u^{-1}(k^{-1}\ (r - a(x))})$$

The RHS of this inequality is equal to $d_r(z)$ because $f_k(r, x) = 1$. 3RD CASE: $u(x) < k^{-1}\ (r - a(x))$. Then by Lemma 3:

$$\mathrm{E}[d_{k^{-1}\ (r-a(x))}(size(h_i(z)))] \leq \frac{m_i(x)}{x} d_{k^{-1}\ (r-a(x))}(x)$$

Hence:

$$s^{j+1}_r(z) \leq \left(\frac{m_1(x) + \cdots + m_k(x)}{x}\right)^{f_k(k^{-1}(r-a(x)),x)+1} \frac{x}{u^{-1}(k^{-f_k(k^{-1}(r-a(x)),x)}(k^{-1}(r - a(x)) + \frac{1}{k-1}\ a(x)) - \frac{1}{k-1}\ a(x))}$$

It is

$$\begin{aligned} f_k(k^{-1}(r - a(x)), x) + 1 &= \left\lceil \log_k \left(k\ \frac{k^{-1}(r - a(x)) + \frac{1}{k-1}\ a(x)}{u(x) + \frac{1}{k-1}\ a(x)} \right) \right\rceil \\ &= \left\lceil \log_k \left(\frac{r + \frac{1}{k-1}\ a(x)}{u(x) + \frac{1}{k-1}\ a(x)} \right) \right\rceil \\ &= f_k(r, x) \end{aligned}$$

Substituting this result in the above estimate yields

$$s^{j+1}_r(z) \leq \left(\frac{m_1(x) + \cdots + m_k(x)}{x}\right)^{f_k(r,x)} \frac{x}{u^{-1}(k^{-f_k(r,x)}\ (r + \frac{1}{k-1}\ a(x)) - \frac{1}{k-1}\ a(x))} = d_r(x)$$

□

4 Applications

4.1 Running Time of Parallel Quicksort

We consider the following version of a randomized Quicksort. In order to sort a list z of length $x = size(z)$ the Pivot element is chosen randomly and the two sublists are sorted recursively in parallel. The divide step requires $\log_2 size(z)$ steps. Thus the probabilistic recurrence relation is

$$T(z) = \log_2 x + \max(T(h_1(z)), T(h_2(z))$$

If the distribution is uniform, we have $E[h_1(z)] = E[h_2(z)] = (size(z) - 1)/2$. Thus the preconditions of Theorem 1 are satisfied, and we have $u(x) = 1/2\ \log_2(x)\ (\log_2(x) + 1)$. Then

$$\Pr[T(z) \geq w + 1/2\ \log_2(x)\ (\log_2(x) + 1)] \leq \left(1 - \frac{1}{size(z)}\right)^w$$

and by corollary 2

$$\begin{aligned} E[T(z)] &\leq 1/2 \log_2(size(z))\ (\log_2(size(z)) + 1) + \left(\log \frac{size(z)}{size(z) - 1} \cdot \log size(z)\right)^{-1} \\ &\leq 1/2 \log_2(size(z))\ (\log_2(size(z)) + 1) + 3 \end{aligned}$$

More generally, if $\mathrm{E}[size(h_1(z))] = n/c$ for a $c > 1$ then it follows from Theorem 1 that $\mathrm{E}[T(z)] = O(\log^2(x))$.

4.2 Stack Size of Sequential Quicksort

Here we are interested in the maximal number of activation records[4] in the stack during the execution of quicksort. The algorithm is the following:

[4]For the definition see,e.g., the section on runtime environments in [ASU 86]

```
PROC quicksort(i,j:integer)
-- sorts elements A[i],...,A[j] of an external array A
  IF i<j THEN
  choose x in {i,..,j} randomly;
  pivot := A[x];
  l := i;
  r := j;
  REPEAT
    swap(A[l],A[r]);
    WHILE A[l] < pivot DO l := l + 1;
    WHILE A[r] < pivot DO r := r - 1;
  UNTIL l>r;
  quicksort(i,l-1);
  quicksort(l,j);
END quicksort
```

Let $S(z)$ be the maximal stack size during the execution of Quicksort on list z. Then $S(h_1(z))$ and $S(h_2(z))$ is the stack size of the two recursive calls. One activation record is needed for the call of *quicksort*(z). Then the stack size is given by the probabilistic recurrence relation

$$S(z) = 1 + \max(S_1(z), S_2(z))$$

If x is chosen uniformly then $\mathrm{E}[size(h_1(z))] = \mathrm{E}[size(h_2(z)] = (size(z) - 1)/2$. If $n = size(z)$ then $u(n) = \log_2 n$ is the least positive solution of $S(n) = 1 + S(n/2)$. The application of Theorem 1 yields

$$\Pr[S(z) \geq w + \log_2 n] \leq \left(1 - \frac{1}{n}\right)^w$$

and by corollary 2

$$\begin{aligned} E[S(z)] &\leq \log_2 n + \left(\log \frac{n}{n-1}\right)^{-1} \\ &\leq \log_2 n + 2 \end{aligned}$$

In general if $\mathrm{E}[h_1(z)] = n/c$ for a $c > 1$, it can be concluded from Theorem 1 that $E[S(z)] = O(\log n)$.

4.3 Average Running Time for the Parallel Testing of Equality on Binary Trees

Testing in parallel whether two binary trees are equal can be done as follows: if the first tree is empty then the result is true iff the second tree is also empty. Otherwise test for the left subtrees and right subtrees in parallel whether they are equal. The result is true iff both recursive calls return true. If z is the first tree, then $h_1(z)$ is the left subtree and $h_2(z)$ is the right subtree. If the input is uniformly distributed, then $\mathrm{E}[size(h_i(z))] = (n-1)/2$ where $n = size(z)$. The probabilistic recurrence relation for the running time is

$$T(z) = c + \max(T(h_1(z)), T(h_2(z)))$$

where c is the time needed for composing the two solutions. As in the previous subsection we obtain with Theorem 1 and corollary 2:

$$\Pr[T(z) \geq w + c\ \log_2 n] \leq \left(1 - \frac{1}{n}\right)^w$$

$$E[T(z)] \leq c \cdot \log_2 n + 2\ c^{-1}$$

Similarly if the distribution is non-uniform and just satisfies $\mathrm{E}[size(h_1(z))] = n/\gamma$ where $\gamma > 1$ with corollary 2 it is still possible to conclude $E[T(z)] = O(\log n)$.

The work of Martinez [Ma 91] shows that this bound is tight. Here a probability distribution is given where the average complexity is $\Theta(\log n)$.

4.4 Automatic Complexity Analysis

In recent years automatic complexity analysis systems are developed. The most important of these methods and systems are in [FSZ 88, HC 88, Mé 75, FSZ 91, Zi 90]. In [FSZ 88, FSZ 91] only uniform input distributions are considered. The method of Flajolet [FSZ 88, FSZ 91] is unable to deal with function composition because the output distribution of a function (or procedure) is usually non-uniform. However an application of Theorems 1 and 5 would allow to estimate the output distribution because of the very general assumptions in the preconditions of these Theorems.

In [Zi 90] it is shown how (deterministic and probabilistic) recurrence relations can be obtained automatically from a program, but they are solved only under very specific assumptions. The above results can be used to improve these methods substantially and make it more generally applicable (because of the very general assumptions on the probability distributions). These results also enable the extension of the above methods to analyze automatically the expected worst case of randomized algorithms.

5 Summary and Further Research

It would be interesting to improve the above results (and the results of [Ka 91]) in the case the upper bounds of the higher moments are known. Currently, we are

able to give estimates for the probability distribution of the running time of parallel divide-and-conquer algorithms. If the number of processors is restricted, then the running time leads to a probabilistic recurrence relation of the form

$$T(z) = a(x) + \max(T(h_1(x)), T(h_2(x))) + \max(T(h_3(x)), T(h_4(x)))$$

These kind of recurrences occur for example in the analysis of a convex hull algorithm (cf., e.g. [Ak 89]). These recurrences require a combination of Theorems 1 and 5.

Acknowledgements. We thank Peter Bürgisser, Gerhard Goos, Dick Karp, and Raimund Seidel for a number of interesting discussions.

References

[ASU 86] Aho, A.V., Sethi, R., and Ullman, J.D. *Compilers: Principles, Techniques, and Tools* Addison-Wesley 1986

[Ak 89] Akl, S. *The Design and Analysis of Parallel Algorithms* Prentice-Hall 1989

[FSZ 88] Flajolet, P., Salvy, B., and Zimmermann, P. *Lambda Upsilon Omega: An Assistant Algorithms Analyzer*, in: The Proceedings of 6th International Conference on Applied Algebra, Algebraic Algorithms and Error Correcting Codes, LNCS 357, pp 201–212, 1988

[FSZ 91] Flajolet, P., Salvy, B. and Zimmermann, P. *Average Case Analysis of Algorithms*, Theoretical Computer Science (79)1, pp. 37 – 110, 1991

[HC 88] Hickey, T. and Cohen, J. *Automating Program Analysis* Journal of the ACM (35)1, pp. 185 – 220, 1988

[Ka 91] Karp, R. M., *Probabilistic Recurrence Relations*, Proc. 23^{rd} ACM STOC (1991), pp. 191-197.

[Ma 91] Martinez, C. *Average Case Analysis of Equality of Binary Trees Under the BST Probability Model* in: Proceedings of the 8th International Conference on Fundamentals of Computation Theory, LNCS 529, pp. 350 – 359, Springer 1991

[Mé 75] LeMétayer D. *ACE: An Automatic Complexity Evaluator* ACM Transactions on Programming Languages and Systems (10)2, pp. 248–266, 1988

[We 75] Wegbreit, B., *Mechanical Program Analysis*, Communications of the ACM (18)9, pp. 528-539, 1975.

[ZZ 91] Zimmermann, P. and Zimmermann W. *The Automatic Complexity Analysis of Divide-and-Conquer Algorithms* in: The Proceedings of the 6th International Symposium on Computing and Information Sciences, 1991

[Zi 90] Zimmermann, W. *Automatische Komplexitätsanalyse funktionaler Programme*, Informatik Fachberichte 261, Springer 1990