Randomized Parallel List Ranking for Distributed Memory Multiprocessors

Frank Dehne$^2$ and Siang W. Song$^3$

We present a randomized parallel list ranking algorithm for distributed memory multiprocessors, using a BSP type model. We first describe a simple version which requires, with high probability, $\log(3p) + \log \ln(n) + O(\log p + \log \log n)$ communication rounds (for relations with $h = O(n/p)$) and $O(n/p)$ local computation. We then outline an improved version that requires high probability, only $r \leq (4k + 6) \log(\frac{1}{2}) + 8 = O(k \log p)$ communication rounds where $k = \min\{1 \geq 0 | \ln^{*+1}n \leq (\frac{3}{2})n^{2^{-k}}\}$. Note $k < \ln^*(n)$ is an extremely small number. For $n < 10^{10^{10^{10}}}$ and $p \geq 4$, the value of $k$ is at most 2. Hence, for a given number of processors, $p$, the number of communication rounds required is, for all practical purposes, independent of $n$. For $n \leq 1, 500,000$ and $4 \leq p \leq 2048$, the number of communication rounds in our algorithm is bounded, with high probability, by 78, but the actual number of communication rounds observed so far is 25 in the worst case. For $n \leq 10^{10^{10^{10}}}$ and $4 \leq p \leq 2048$, the number of communication rounds in our algorithm is bounded, with high probability, by 118; and we conjecture that the actual number of communication rounds required will not exceed 50. Our algorithm has a considerably smaller number of communication rounds than the list ranking algorithm used in Reid-Miller's empirical study of parallel list ranking on the Cray C-90.$^{11}$ To our knowledge, Reid-Miller's algorithm$^{11}$ was the fastest list ranking implementation so far. Therefore, we expect that our result will have considerable practical relevance.

KEY WORDS: Parallel algorithms; list ranking; coarse grained multi-computer.

---

$^1$Research partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC), Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Proc. No. 95/0767-0, 95/1367-5, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Proc. No. 52311/94-7 and PROTEM II, and the Commission of the European Community (Project ITDC-207).

$^2$School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6. E-mail: dehne@scs.carleton.ca.

$^3$Department of Computer Science - IME, University of São Paulo, 05508-900 São Paulo, SP, Brazil. E-mail: song@ime.usp.br.
1. INTRODUCTION

1.1. The Model

Speedup results for theoretical PRAM algorithms do not necessarily match the speedups observed on real machines.\textsuperscript{12, 31} Given sufficient slackness in the number of processors, Valiant's BSP approach\textsuperscript{41} simulates PRAM algorithms optimally on distributed memory parallel systems. Valiant points out, however, that one may want to design algorithms that utilize local computations and minimize global operations.\textsuperscript{4, 31} The BSP approach requires that $g$ (local computation speed/router bandwidth) is low, or fixed, even for increasing number of processors. Gerbessiotis and Valiant\textsuperscript{5} describe circumstances where PRAM simulations can not be performed efficiently, among others if the factor $g$ is high. Unfortunately, this is true for most currently available multiprocessors. The algorithm presented here considers this case for the list ranking problem. Furthermore, as pointed out in Ref. 4, the cost of a message also contains a constant overhead cost $s$. The value of $s$ can be fairly large and the total message overhead cost can have a considerable impact on the speedup observed (see e.g., Ref. 7).

We are therefore using a slightly enhanced version of the BSP model, referred to as coarse grained multicomputer model.\textsuperscript{17-9} It is comprised of a set of $p$ processors $P_1, ..., P_p$ with $O(n/p)$ local memory per processor and an arbitrary communication network (or shared memory). All algorithms consist of alternating local computation and global communication rounds. Each communication round consists of routing a single $h$-relation with $h = \hat{O}(n/p)^4$, i.e., each processor sends $\hat{O}(n/p)$ data and receives $\hat{O}(n/p)$ data. We require that all information sent from a given processor to another processor in one communication round is packed into one message. In the BSP model, a computation/communication round is equivalent to a super-step with $L = (n/p) g$ (plus the previous "packing requirement").

Finding an optimal algorithm in the coarse grained multicomputer model is equivalent to minimizing the number of communication rounds as well as the total local computation time. This considers all the discussed parameters that are affecting the final observed speedup and requires no assumption on $g$. Furthermore, it has been shown that minimizing the number of supersteps also leads to improved portability across different parallel architectures.\textsuperscript{4, 5, 11} This model has been used (explicitly or implicitly) in parallel algorithm design for various problems\textsuperscript{9, 12-15} and has shown very good practical timing results.

\textsuperscript{4} $\hat{O}(n)$ denotes $O(n)$ "with high probability." More precisely, $X = \hat{O}(f(n))$, if and only if $(\forall x > x_0 > 1) \text{ Prob} [X > x f(n)] \leq 1/n^{o(c)}$, where $c_0$ is a fixed constant and $g(c)$ is a polynomial in $c$ with $g(c) \rightarrow \infty$ for $c \rightarrow \infty$.\textsuperscript{10a}
1.2. The List Ranking Problem

Consider a linear linked list consisting of a set $S$ of $n$ nodes and, for each node $x \in S$, a pointer $(x \rightarrow \text{next}(x))$ to its successor, $\text{next}(x)$, in the list. Let $\lambda \in S$ be the last list element and $\text{next}(\lambda) = \lambda$. The list ranking problem consists of computing for each $x \in S$ the distance of $x$ to $\lambda$, referred to as $\text{dist}(x)$.

We assume that, initially, every processor stores $n/p$ nodes and, for each of these nodes, the pointer $(x \rightarrow \text{next}(x))$ to the next list element. See Fig. 1. As output we require that every processor stores for each of its $n/p$ nodes $x \in S$ the value $\text{dist}(x)$.

A trivial sequential algorithm solves the list ranking problem in optimal linear time by traversing the list. Several PRAM list ranking algorithms have been proposed.\textsuperscript{(16, 17)} Wyllie\textsuperscript{(18)} proposed a non-optimal $O(\log n)$ time algorithm with total work greater than $O(n)$. The first optimal $O(\log n)$ EREW PRAM algorithm is due to Cole and Vishkin.\textsuperscript{(19)} Another optional deterministic algorithm is given by Anderson and Miller.\textsuperscript{(20)} Parallel list ranking algorithms using randomization were proposed by Miller and Reif.\textsuperscript{(21, 22)} The algorithms use $O(n)$ processors. The optimal algorithm by Anderson and Miller\textsuperscript{(23)} improves this by using an optimal number of processors. A $O(\sqrt{n})$ time mesh algorithm is described in Ref. 24. Reid-Miller\textsuperscript{(11)} presented an empirical study for the Cray C-90 which will be discussed in the next subsection. See Section 6 for some of the many applications of list ranking.

1.3. The Results

We present a randomized parallel list ranking algorithm for the coarse grained multicomputer model discussed earlier. We first describe a simple version which requires, with high probability, $\log(3p) + \log \ln(n) = O(\log p + \log \log n)$ communication rounds. Then, we outline an improved
version which requires, with high probability, only \( r \leq (4k+6) \log(1/p) + 8 = \tilde{O}(k \log p) \) communication rounds where \( k = \min\{i \geq 0 | \ln^{i+1} n \leq (\frac{1}{4} p)^{2^{i+1}}\} \).

We observe that \( k < \ln^*(n) \) is an extremely small number. For \( n \leq 10^{10^8} \) and \( p \geq 4 \), the value of \( k \) is at most 2. That is, for a given number of processors, \( p \), the number of communication rounds required is, for all practical purposes, independent of \( n \).

For \( n \leq 10^{10^8} \) and \( 4 \leq p \leq 2048 \), the number of communication round, \( r \), is bounded, with high probability, by 118. See Table I. Note that, this is only an upper bound on the number of communication rounds. For \( 100,000 \leq n \leq 1,500,000 \) and \( 4 \leq p < 2048 \), \( 0 \) with high probability, \( r \) is

<table>
<thead>
<tr>
<th>( p )</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>( k; R )</td>
<td>( k; R )</td>
<td>( k; R )</td>
<td>( k; R )</td>
<td>( k; R )</td>
<td>( k; R )</td>
<td>( k; R )</td>
<td>( k; R )</td>
<td>( k; R )</td>
<td>( k; R )</td>
</tr>
<tr>
<td>( 10^{10} )</td>
<td>1:18</td>
<td>0:26</td>
<td>0:32</td>
<td>0:38</td>
<td>0:44</td>
<td>0:50</td>
<td>0:56</td>
<td>0:62</td>
<td>0:68</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^5} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^8} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:8</td>
</tr>
<tr>
<td>( 10^{10^9} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{10}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{11}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{12}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{13}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{14}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{15}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{16}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{17}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{18}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{19}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{20}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
<tr>
<td>( 10^{10^{22}} )</td>
<td>1:18</td>
<td>1:38</td>
<td>1:48</td>
<td>1:58</td>
<td>1:68</td>
<td>1:78</td>
<td>1:88</td>
<td>1:98</td>
<td>1:108</td>
<td>0:74</td>
</tr>
</tbody>
</table>

Table I. Values of \( k \) and \( R := (4k+6) \log(3/p) + 8 \) (Upper Bound on \( r \)) for Various Combinations of \( n \) and \( p \)
bounded by 78 in the worst case. See Table II. We simulated 100 test runs
of our algorithm for each of the $n, p$ combinations shown in Table II. The
observed numbers of communication rounds actually required were always
much lower, and never exceeded 25.

For $n \leq 10^{20}$ and $4 \leq p \leq 2048$, the number of communication rounds
in our algorithm is bounded, with high probability, by 118, and we conjec-
ture that the actual number of communication rounds required will not
exceed 50.

Our randomization technique is very different from the ones used in
Refs. 21–23. In this model, our algorithm uses considerably fewer com-
munication rounds than others. [11, 16–19, 20, 25]

The simple version of our algorithm is a generalization of the algo-
rithm used in Reid-Miller’s empirical study of parallel list ranking for the
Cray C-90 in shared memory mode. The analysis of our simple list ranking
algorithm improves the estimates on the load imbalance provided in Ref.
1. Our improved algorithm also applies to the Cray C-90. Since it requires
significantly fewer communication rounds than the algorithm used in Ref.
1, we expect that our result will considerably improve the running times
observed. To our knowledge, Reid-Miller’s algorithm was the fastest list
ranking implementation so far. Therefore, we expect that our result will
have considerable practical relevance.

As with Reid-Miller’s algorithm we will, in general, assume that
$n \gg p$ (coarse grained), because this is usually the case in practice. Note,
however, that our results hold for arbitrary ratios $n/p$.

1.4. Overview

In the remainder of this paper, we will first prove a result on random
sampling in linear linked lists. In Section 3, we outline the simple version
of our algorithm which is based on a single random sampling of list nodes.
In Section 4, we introduce an incremental method to improve the first
sample. We present a considerably improved list ranking algorithm, which
is the main result of this paper. In Section 5, we discuss the results of our
simulation of the improved list ranking algorithm. Finally, in Section 6, we
outline some applications.

2. RANDOM SAMPLING IN LINEAR LINKED LISTS

Consider a linear linked list with a set $S$ of $n$ nodes. In this section we
will show that if we select $n/p$ random elements (pivots) of $S$ then, with
high probability, these pivots will split $S$ into sublists whose maximum size
is bound by $3p \ln(n)$; see Fig. 2.
<table>
<thead>
<tr>
<th>$p$</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>100,000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>200,000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>300,000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>400,000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>500,000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>600,000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>700,000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>800,000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>900,000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

* $R := (4k + 6) \log(p) + 8$, $m^{\text{obs}}$ and $r^{\text{obs}}$ for various combinations of $n$ and $p$. (For each combination of $n$ and $p$, the $m^{\text{obs}}$ and $r^{\text{obs}}$ shown are the worst case values observed during 100 test runs.)
### Table II. Continued

<table>
<thead>
<tr>
<th>$p = \frac{n}{x}$</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>38</td>
<td>58</td>
<td>68</td>
<td>78</td>
<td>56</td>
<td>62</td>
<td>68</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>60</td>
<td>90</td>
<td>100</td>
<td>110</td>
<td>88</td>
<td>96</td>
<td>102</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>$m_{obs}^{x/2R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_{obs}^{x/4R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_{obs}^{x/8R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_{obs}^{x/16R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_{obs}^{x/32R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_{obs}^{x/64R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_{obs}^{x/128R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_{obs}^{x/256R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_{obs}^{x/512R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_{obs}^{x/1024R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_{obs}^{x/2048R}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

We recall the following Lemma from Bleloch et al.,\(^{123}\) in a slightly modified form for linked lists (rather than for arrays).

**Lemma 1.** $xk \leq n$ randomly chosen elements of $S$ (pivots) partition list $S$ into sublists $S_j$, such that the size of the largest sublist is at most $n/x$ with a probability of at least

$$1 - 2x \left(1 - \frac{1}{2x}\right)^x$$

*Fig. 2.* A linear linked list with random pivots.
Proof. (Analogous to Blelloch et al.\cite{12}) Assume that the nodes of $S$ are sorted by their rank. This sorted list can be viewed as $2x$ segments of size $n/2x$. If every segment contains at least one pivot (chosen element), then $\max_{1 \leq j \leq nk} |S_j| \leq n/x$. Consider one segment. Since the pivots are chosen randomly, the probability that a specific pivot is not in the segment is $(1 - (1/2x))$. Since $xk$ pivots are selected independently, the probability that none of the pivots are in the segment is $(1 - (1/2x))^xk$. Therefore, even assuming mutual exclusion, the probability that there exists a segment which contains no pivot is at most $2x(1 - (1/2x))^{xk}$. Hence, every segment contains at least one pivot with the probability being at least $1 - 2x(1 - (1/2x))^{xk}$. \hfill \Box

Corollary 1. $xk \leq n$ randomly chosen pivots partition list $S$ into $xk + 1$ sublists $S_i$, such that there exists a sublist $S_i$ of size larger than $c(n/x)$ with a probability of at most $(2x/c)(1 - (c/2x))^{xk} \leq (2x/c) e^{-11/2} x k$, if $c > 2$.

Lemma 2. Consider $xk < n$ randomly chosen pivots which partition $S$ into $xk + 1$ sublists $S_i$, and let $m = \max_{0 \leq i < xk} |S_i|$. If $k \geq \ln(x) + 2 \ln(n)$ then $\Pr\{m > c(n/x)\} \leq 1/n^c$, $c > 2$.

Proof. Corollary 1 implies that

$$\Pr\{m > c(n/x)\} \leq \frac{2x}{c} e^{-11/2} x k$$

We observe that, for $c > 2$,

$$\ln(x) + 2 \ln(n) \leq k \Rightarrow \frac{2}{c} \ln \left(\frac{2x}{c}\right) + 2 \ln(n) \leq k \Rightarrow \ln \left(\frac{2x}{c}\right) + c \ln(n) \leq \frac{ck}{2} \Rightarrow \frac{2x}{c} n^c \leq e^{ck/2} \Rightarrow \Pr\{m > c(n/x)\} \leq n^{-c} \hfill \Box$$

Theorem 1. $n/p$ randomly chosen pivots partition $S$ into $(n/p) + 1$ sublists $S_j$ with $m = \max_{0 \leq j \leq p} |S_j|$ such that

$$\Pr\{m \geq c3p \ln(n)\} \leq \frac{1}{n^c}, \quad c > 2$$
Proof. Let \( x = \frac{n}{3p} \ln(n) \), \( k = \ln(x) + 2 \ln(n) = 3 \ln(n) - \ln(3p \ln(n)) \).
Then \( xk = \frac{n(p)}{3 \ln(n) - \ln(3p \ln(n))} \leq \frac{n}{p} \), and Theorem 1 follows from Lemma 2.

\[ \Box \]

3. A SIMPLE ALGORITHM USING A SINGLE RANDOM SAMPLE

In this section we will present a simple list ranking algorithm which requires, with high probability, at most \( \log(3p) + \log \ln(n) = \tilde{O}(\log p + \log \log n) \) communication rounds. This algorithm is based on a single random sample of nodes. We will later improve the performance of the algorithm by improving the sample through a sequence of sampling rounds.

Consider a random set \( S' \subseteq S \) of pivots. For each \( x \in S \) let \( \text{nextPivot}(x, S') \) refer to the closest pivot following \( x \) in the list \( S \).
(We assume that the last element, \( \lambda \), of \( S \) is selected as a pivot and let \( \text{nextPivot}(\lambda, S') = \lambda \). Note that for \( x \neq \lambda \), \( \text{nextPivot}(x, S') \neq x \).) Let \( \text{distToPivot}(x, S') \) be the distance between \( x \) and \( \text{nextPivot}(x, S') \) in list \( S \).
Furthermore, let \( m(S, S') = \max_{x \in S} \text{distToPivot}(x, S') \).

The modified list ranking problem for \( S \) with respect to \( S' \) refers to the problem of determining for each \( x \in S \) its next pivot \( \text{nextPivot}(x, S') \) as well as the distance \( \text{distToPivot}(x, S') \). The input/output structure for the modified list ranking problem is the same as for the list ranking problem.

3.1. Algorithm 1

1. Select a set \( S' \subseteq S \) of \( \tilde{O}(n/p) \) random pivots as follows: Every processor \( P_i \) makes for each \( x \in S \) stored at \( P_i \) an independent biased coin flip which selects \( x \) as a pivot with probability \( 1/p \).

2. All processors solve collectively the modified list ranking problem for \( S \) with respect to \( S' \) (details will be discussed later).

3. Using an all-to-all broadcast, the values \( \text{nextPivot}(x, S') \) and \( \text{distToPivot}(x, S') \) for all pivots \( x \in S' \) are broadcast to all processors.

4. Using the data received in Step 3, each processor \( P_i \) can solve the list ranking problem for the nodes stored at \( P_i \) sequentially in time \( \tilde{O}(n/p) \).

—End of Algorithm—

For the correctness of Step 1, we recall the following in Lemma 3.
Lemma 3. [Ref. 10]. Consider a random variable $X$ with binomial distribution. Let $n$ be the number of trials, each of which is successful with probability $q$. The expectation of $X$ is $E(X) = nq$ and
\[ \Pr(X > cnq) \leq e^{-(1/2)(c - 1)^2} nq, \quad \text{for any } c > 1 \]

In order to implement Step 2, we simply simulate the standard recursive doubling technique. (For all $x$ in parallel: WHILE next($x$) $\neq$ nextPivot($x$, $S'$) DO next($x$) := next(next($x$)).) From Theorem 1 it follows that, with high probability, $m(S, S') \leq 3p \ln(n)$. Hence, Step 2 requires, with high probability, at most $\log(3p \ln(n)) = \log(3p) + \log \ln(n)$ communication rounds. Step 3 requires 1 communication round, and Step 4 is straightforward. In summary, we obtain Theorem 2.

Theorem 2. Algorithm 1 solves the list ranking problem using, with high probability, at most $1 + \log(3p) + \log \ln(n)$ communication rounds and $O(n/p)$ local computation.

We observe that, if $n/p \leq e^{-3p}$, for some $x > 1$ then,
\[ \ln(n) \leq \ln(p) + (3p)^x \]
\[ = \log \ln(n) \leq \log(\ln(p) + (3p)^x) \leq \log(2(3p)^x) \]
\[ = \log(3p) + \log \ln(n) \leq 1 + (x + 1) \log(3p) \]

This implies the following Corollary.

Corollary 2. If $n/p \leq e^{-3p}$, for some constant $x > 1$, then the number of communication rounds required by Algorithm 1 is bounded by $2 + (x + 1) \log(3p) = \Theta(\log p)$.

4. IMPROVING THE MAXIMUM SUBLIST SIZE

We will now present our algorithm which improves the maximum sublist size obtained in Algorithm 1 and solves the list ranking problem by using, with high probability, only $r < (4k + 6) \log(3/2) + 8$ communication rounds and $\tilde{O}(n/p)$ local computation where
\[ k := \min\{i \geq 0 | \ln^{(i+1)} n \leq (3/2)^{2i+1} \} \]

Note that $k < \ln^*(n)$ is an extremely small number (see Table 1). Figure 3 illustrates $\ln^{(i+1)} n$ and $(3/2)^{2i+1}$ as functions of $i$, as well as their intersection point $k$. 
The basic idea of the algorithm is that any two pivots should not be closer than $O(p)$ because this creates large "gaps" elsewhere in the list. If two pivots are closer than $O(p)$, then one of them is "useless" and should be "relocated." The nontrivial part is to perform the "relocation" without too much overhead and such that the new set of pivots has a considerably better distribution. The algorithm uses three colors to mark nodes: black (pivot), red (a node close to a pivot), and white (all other nodes).

4.1. Algorithm 2

1. Perform Step 1 of Algorithm 1. Mark all selected pivots black and all other nodes white.

2. For $i = 1, \ldots, 2$
   2a. For each black node $x$, all nodes which are to the right of $x$ (in list $S$) and have distance at most $\frac{3}{4} p$ are marked red. Note: previously black nodes (pivots) that are now marked red are no longer considered pivots.
   2b. For each black node $x$, all nodes which are to the left of $x$ (in list $S$) and have distance at most $\frac{3}{4} p$ are marked red.
   2c. Every processor $P_i$ makes for each white node $x \in S$ stored at $P_i$ and independent biased coin flip which selects $x$ as a new pivot, and marks it black, with probability $1/p$. 
(2d) Every processor $P_i$ marks white every red node $x \in S$ stored at $P_i$.

(3) Let $S' \subseteq S$ be the subset of black nodes obtained after Step 2. Continue with Steps 2–4 of Algorithm 1.

—End of Algorithm—

Observe that Steps 2a and 2b have to be performed in a left-to-right scan, respectively, as if executed sequentially. We can simulate this sequential scanning process in the parallel setting because the number of pivots is bounded by $n/p$. For Step 2a, we build linked lists of pivots by computing for each of them a pointer to the next pivot of distance at most $2p/3$, if any, and the distance. These linked lists of pivots are compressed into one processor and we run on these lists a sequential left-to-right scan to mark pivots red. We return the pivots to their original location and mark every nonpivot red for which there exists a nonred pivot that attempts to mark it red. Step 2b is performed analogously. Note that each node $x$ requires a pointer to its predecessor $\text{prev}(x)$ in the linked list. All $\text{prev}(x)$ values can be easily computed with one communication round and $O(n/p)$ local computation.

Let $r$ be the number of communication rounds required by Algorithm 2. We will now show that, with high probability,

$$r \leq (4k + 6) \log(\frac{n}{p}) + 8 = \tilde{O}(k \log p)$$

Let $n_i$ be the maximum length of a contiguous sequence of white nodes after the $i$th execution of Step 2b, and define $n_0 = n$.

Let $S_i$ be the set of black nodes after the $i$th execution of Step 2c, $1 \leq i \leq k$, and let $S_0$ be the set of black nodes after the execution of Step 1. Note that, in Step 3, $S' = S_k$. Define $m_i = m(S_i)$ for $0 \leq i \leq k$.

**Lemma 4.** With high probability, the following holds:

(a) $n_0 = n$ and $n_i \leq 3p \ln(n_{i-1})$, $1 \leq i \leq k$

(b) $m_i \leq 3p \ln(n_i)$, $0 \leq i \leq k$

**Proof.** It follows from Theorem 1 that, with high probability,

$$n_0 = n$$

$$m_0 \leq 3p \ln(n)$$

and, for a fixed $1 \leq i \leq k$

$$n_i \leq m_{i-1}$$

$$m_i \leq 3p \ln(n_i)$$
Randomized Parallel List Ranking

Since $k \leq \ln^*(n)$ and $\log^*(n) 1/n^\varepsilon \leq 1/n^{\varepsilon^{-}}$, $\varepsilon > 0$, the above bounds for $n_i$ and $m_i$ hold, with high probability, for all $1 \leq i \leq k$. □

Lemma 5. With high probability, for all $1 \leq i \leq k$,

(a) $n_i \leq 3p(2 \ln(3p) + \ln^{i+1}(n))$

(b) $m_i \leq 6p \ln(3p) + 3p \ln^{i+1}(n)$

Proof.

(a) Applying Lemma 4 we observe that

\[
\begin{align*}
 n_1 &\leq 3p \ln(n) \\
n_2 &\leq 3p \ln(3p \ln(n)) \\
 &\leq 3p(\ln(3p) + \ln(n)) \\
n_3 &\leq 3p \ln(n_2) \\
 &\leq 3p(\ln(3p) + \ln(\ln(3p) + \ln(n))) \\
 &\leq 3p(\ln(3p) + \ln(\ln(n))) \\
n_4 &\leq 3p \ln(n_3) \\
 &\leq 3p(\ln(3p) + \ln(n) + \ln \ln(3p) + \ln \ln(n)) \\
 &\vdots \\
n_i &\leq 3p(2 \ln(3p) + \ln^{i+1}(n))
\end{align*}
\]

(b) It follows from Lemma 4 that

\[
\begin{align*}
m_i &\leq 3p \ln(n_i) \leq 3p \ln(3p(2 \ln(3p) + \ln^{i+1}(n))) \\
 &\leq 3p(\ln(3p) + \ln(2) + \ln^{i+1}(3p) + \ln^{i+1}(n)) \\
 &\leq 6p \ln(3p) + 3p \ln^{i+1}(n) \quad \square
\end{align*}
\]

Theorem 3. Algorithm 2 with high probability, solves the list ranking problem with $r \leq (4k + 6) \log(\frac{1}{p}) + 8 = \tilde{O}(k \log p)$ communication rounds and $\tilde{O}(n/p)$ local computation.

Proof. The total number of communication rounds in Algorithm 2 with high probability, is bounded by
\[2k \log(\frac{3}{2}p) + \log(m_k) + 1\]
\[\leq 2k \log(\frac{3}{2}p) + \log(6p) + \log \ln(3p) + \log(3p) + \log \ln^{k+1}(n) + 1\]
\[\leq (2k + 3) \log(\frac{3}{2}p) + \log 9 + \log 4.5 + \log \ln^{k+1}(n) + 1\]
\[\leq (2k + 3) \log(\frac{3}{2}p) + \log \ln^{k+1}(n) + 8\]
\[\leq \log((\frac{3}{2}p)^{2k+3}) + \log \ln^{k+1}(n) + 8\]
\[\leq 2 \log((\frac{3}{2}p)^{2k+3}) + 8 \text{ if } (*) \ln^{k+1}(n) \leq (\frac{3}{2}p)^{2k+3}\]
\[\leq (4k + 6) \log(\frac{3}{2}p) + 8 = \tilde{O}(k \log p)\]

Condition (*) is true because we selected \(k = \min \{i \geq 0 | \ln^{i+1}(n) \leq (\frac{3}{2}p)^{2i+1}\}\). Note that, this bound is not tight. \(\square\)

5. SIMULATION RESULTS

We simulated the behavior of Algorithm 2. In particular, we simulated how this method improves the sample by reducing the maximum distance, \(m_k\), between subsequent pivots. We examined the range of \(4 \leq p \leq 2048\) and \(100,000 \leq n \leq 1,500,000\) as shown in Table II and applied Algorithm 2 for each \(n, p\) combination shown 100 times with different random samples. Table II shows the values of \(k\) and the upper bound \(R\) on the number of communication rounds required according to Theorem 3. We then measured the maximum distance observed between two subsequent pivots, \(m_k\)^s in the sample chosen at the end of the algorithm, as well as the number of communication rounds actually required, \(r\)^s. Each of the numbers shown is the worst case observed in the respective 100 test runs.

According to Theorem 3, for the range of test data used, the number of communication rounds in our algorithm should not exceed 78. This is an upper bound, though. The actual number of communication rounds observed in Table II is 25 in the worst case. The number of rounds observed is usually around 30% of the upper bound according to Theorem 3. We also observe that for a given \(p\) (i.e., in a vertical column), the values of \(m_k\)^s and \(m_k\)^s are essentially stable and show no monotone increase or decrease with increasing \(n\).

6. APPLICATIONS

The problem of list ranking is a special case of computing the suffix sums of the elements of a linked list. This algorithm obviously can be generalized to compute prefix or suffix sums for associative operators (by
Replacing the addition operation for node distances by the respective associative operator. List ranking is a very popular tool for obtaining numerous parallel tree and graph algorithms.\cite{1, 24, 25}

An important application outlined in Ref. 24 is to use list ranking for applying Euler tour techniques to tree problems. As demonstrated by Atallah and Hambrusch,\cite{24} once an efficient distributed memory parallel list ranking algorithm is available, it is easy to obtain efficient distributed memory parallel algorithms for the following problems for an undirected forest of trees: rooting every tree at a given vertex chosen as root, determining the parent of each vertex in the rooted forest, computing the preorder (or postorder) traversal of the forest, computing the level of each vertex, and computing the number of descendants of each vertex. All these problems can be easily solved with one or a small constant number of list ranking operations.

7. CONCLUSION

We presented a randomized parallel list ranking algorithm for distributed memory multiprocessors using the coarse grained multicomputer model. The algorithm requires, with high probability, \( r \leq (4k + 6) \log(\xi p) + 8 = \tilde{O}(k \log p) \) communication rounds. For all practical purposes, \( k \leq 2 \). The algorithm presented improves on the number of communication rounds required in Reid-Miller’s\cite{11} list ranking implementation for the Cray C-90 which was, to our knowledge, the fastest list ranking implementation to date. Therefore, we expect that our result will have considerable practical relevance.

REFERENCES


