# Probabilistic Solitude Detection I: Ring Size Known Approximately**

Karl Abrahamson*
Andrew Adler†
Lisa Higham*
David Kirkpatrick*

Technical Report 87-8
March 1987

*Department of Computer Science
†Department of Mathematics

** A preliminary version of this paper appeared in the Proceedings of the Fifth ACM Symposium on Principles of Distributed Computing. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Killam Foundation.

**Abstract**

Matching upper and lower bounds for the bit complexity of a problem on asynchronous unidirectional rings are established, assuming that algorithms must reach a correct conclusion with probability $1 - \epsilon$, for some $\epsilon > 0$. Processors can have identities, but the identities are not necessarily distinct. The problem is that of a distinguished processor determining whether it is the only distinguished processor. The complexity depends on the processors' knowledge of the size $n$ of the ring. When no upper bound is known, only nondistributive termination is possible, and $\Theta(n \log(\frac{1}{\epsilon}))$ bits are necessary and sufficient. When only an upper bound $N$ is known, distributive termination is possible, but the complexity of achieving distributive termination is $\Theta(n\sqrt{\log(\frac{N}{n})} + n\log(\frac{1}{\epsilon}))$. When processors know that $(\frac{1}{2}+\rho)N \leq n \leq N$ for $\rho > 0$, then the bound drops to $\Theta(n \log\log(\frac{1}{\epsilon}) + n\log(\frac{1}{\rho}))$, for both distributive and nondistributive termination, for sufficiently large $N$.

# 1 Introduction

An asynchronous unidirectional ring of processors is one of the simplest of network topologies. Nonetheless, rings exhibit features which can be expected to show up in many topologies. Consequently, rings serve as a suitable test-bed for studies in distributed computing.

Numerous studies in distributed computing have made it clear that, in general, the complexity, or indeed the solvability, of a distributed problem depends on features of the processors and on the nature of the desired solution. Features which are relevant to an asynchronous ring are:

**Knowledge.** What does each processor know about the size of the ring? Do processors have identities? To what extent can an algorithm exploit identities?

**Type of algorithm.** Is the desired algorithm deterministic, randomized (always correct) or probabilistic (correct with probability $1 - \epsilon$)?

**Type of termination.** Must the algorithm terminate distributively, or is nondistributive termination acceptable? An algorithm terminates distributively if each processor, upon reaching a conclusion, will not subsequently revoke its conclusion on receipt of further messages. (Nondistributively terminating algorithms are rarely admitted in the literature. Reasons for considering them here will become apparent.)

This paper is one of a series of three papers (see [1,2]) addressing the question of how the above features affect the bit complexity, that is, the total number of bits transmitted, of a fundamental problem, *solitude detection*, on

asynchronous unidirectional rings. Taken together, the three papers give a fairly complete answer. The bit complexity of solitude detection exhibits a surprisingly rich dependence on features of the ring and the nature of algorithms, but not so rich that analysis is intractable.

The solitude detection problem is as follows. A nonempty set of processors, called contenders, are distinguished. Each processor must determine whether or not there is exactly one contender.

In this paper, we are primarily concerned with versions of solitude detection which cannot be solved with certainty, but which can be solved probabilistically, with arbitrarily small positive probability of error.

One of a few recognized fundamental problems on rings is that of electing a leader. It was pointed out in [5], and later expanded upon in [1], that leader election is composed of two more fundamental problems: attrition and solitude verification. The attrition problem is that of reducing a set of contenders to just one contender. Solitude verification is a weak form of solitude detection; a solitary contender must verify that it is the only contender. When there are two or more contenders, solitude verification requires only that no contender concludes that it is the only contender.

Attrition, solitude verification and solitude detection all reduce to leader election in $O(n)$ bits, where $n$ is the size of the ring. Thus our lower bounds for solitude verification imply corresponding lower bounds for leader election.

Deterministic [4,8,7] and randomized [1,5] solutions to leader election and solitude detection in unidirectional rings have been considered elsewhere. Pachl [6] shows that probabilistic algorithms are not significantly better than deterministic algorithms for the closely related problem of maximum finding when processors have distinct identities.

However, for solitude verification, probabilistic algorithms can be more powerful than deterministic algorithms. We establish complementary upper and lower bounds on the bit complexity of probabilistic solitude detection in two situations where processors do not necessarily have distinct identities, and an error-free solution is impossible. The two situations are (1) each processor knows an upper bound on the size of the ring, and termination is distributive, and (2) each processor knows nothing about the size of the ring, and termination is nondistributive.

In addition, we consider one situation where solution with certainty is possible, but admitting error can decrease the complexity significantly. That is the situation in which each processor knows a quantity $N$ such that the ring size $n$ lies in the range $(1/2 + \rho)N \leq n \leq N$, where $\rho > 0$.

Section 2 summarizes the results of this paper. Results of the two com-

panion papers are described in Section 7. Section 3 contains descriptions and correctness proofs of some solitude detection algorithms, assuming various properties of the ring. The corresponding lower bounds are proved in Section 5, using a model described in Section 4. An additional section (section 6) compares two types of error tolerance in order to enhance the generality of the lower bounds.

## 2 Overview of Results

A solitude detection algorithm runs on a ring of processors of two kinds: contenders and non-contenders. There is guaranteed to be at least one contender. The problem is to determine whether or not there is exactly one contender. It is presumed without loss of generality that only contenders may send a message without first receiving one.

There are some important differences between the algorithms with which we establish upper bounds and the model of computation to which our lower bounds apply.

Our algorithms assume that processors of each kind (contenders and non-contenders) are indistinguishable from one another. Each processor of a given type runs the same probabilistic process. On the other hand, our lower bounds apply when processors can have arbitrary identities, or equivalently, when each processor might run a different probabilistic process. Algorithms may rely on identities for achieving efficiency. The crucial requirement is that identities are not guaranteed to be distinct. Specifically, algorithms must not rely on either the distinctness or the distribution of identities for their correctness. This generality in the lower bounds is a result of a nondeterministic attribute of the model for which the lower bounds hold. A more extensive discussion of the implications of the nondeterministic aspects of the model appears in the conclusions of a related paper, [2].

Our algorithms make errors on one side only. If there is only one contender, then with probability one, our algorithms cause that contender to reach the conclusion "alone". If there are two or more contenders, then with probability at least $1 - \epsilon$ every processor reaches the conclusion "not alone". Our lower bounds permit two-sided error of at most $\epsilon$ on each side. Although the error tolerance is the same for both the "alone" and "not alone" cases, even this is not an essential restriction. The extension to tolerance of $\delta$ error when there is one contender, and tolerance of $\epsilon$ error when there are more than one contender, is presented in section 6.

Our algorithms all solve solitude detection. In fact, when there are two or more contenders, the algorithms send only $O(n)$ expected bits. But our

lower bounds apply to a weaker problem which we call weak solitude verification. Specifically, the lower bounds apply to the expected bit complexity of an algorithm applied to any configuration where *there is one contender*, regardless of the complexity when there are two or more contenders, or of the complexity of other configurations with one contender. An algorithm can loop forever or deadlock for some rings of processors, and our lower bounds still apply. Thus the lower bounds preclude the existence of an algorithm which is very efficient on some ring with one contender in which processors are labelled in a particular way, while still tolerating error of at most $\epsilon$ on all rings.

(Examination of our algorithms gives some indication of why the lower bounds can be so strong. Essentially, a solitude detection algorithm runs until it has run "long enough" without detecting non-solitude. Moreover, non-solitude is usually detected with few bits, when there are two or more contenders. So it is pointless for an algorithm to fail to detect non-solitude, and it gives nothing away to permit such behaviour.)

All complexity bounds stated below apply when there is one contender.

Without distinct identities, no algorithm can distinguish with certainty between a ring of size $n$ containing one contender and a ring of size $2n$ containing 2 contenders directly opposite each other. Consequently, no algorithm can solve solitude verification with certainty, even if all processors know that $N \leq n \leq 2N$. Furthermore, for any possible communication history, $h$, of one processor on a ring $R$ of size $n$ with one contender, it is possible, by splicing together enough copies of $R$, to form a ring $R'$ with several contenders in which the probability that more than one processor has a history of which $h$ is an initial segment is arbitrarily close to one. Therefore, without any knowledge about ring size, it is impossible for a processor to halt and declare its solitude with any degree of certainty. This implies that when processors have no knowledge of ring size, there can be no distributively terminating algorithm $\alpha$ for solitude detection even if $\alpha$ is allowed to give the wrong answer with arbitrarily high preassigned probability less than one.

Solitude detection can be solved probabilistically with nondistributive termination even if nothing is known about the ring size. A simple nondistributively terminating probabilistic algorithm for solitude detection (cf. Theorem 3.1) achieves probability of correctness at least $1 - \epsilon$ by exchanging a total of $O(n \log(\frac{1}{\epsilon}))$ bits. A matching lower bound (Theorem 5.2) shows that $\Omega(n \log(\frac{1}{\epsilon}))$ bits are required to solve solitude detection with probability $1 - \epsilon$ of correctness, with nondistributive termination.

If processors know that ring size $n$ is at most $N$, there is a distribu-

tively terminating probabilistic solitude detection algorithm with expected bit complexity $O(n\sqrt{\log(\frac{N}{n})} + n\log(\frac{1}{\epsilon}))$ (Theorem 3.4). That algorithm uses as a subroutine another algorithm that solves solitude detection with probability at least $1-\epsilon$ of correctness using $O(n\sqrt{\log C} + n\log(\frac{1}{\epsilon}))$ bits, where $C$ is an upper bound on the number of contenders (Theorem 3.2).

There is a matching lower bound (Theorems 5.1 and 5.2) of $\Omega(n\sqrt{\log(\frac{N}{n})} + n\log(\frac{1}{\epsilon}))$ when an upper bound $N$ on ring size is known and $n \leq N/2$. The lower bound applies to expected case.

The proof of the latter lower bound depends upon a doubling argument and therefore does not hold when $n > N/2$. In fact, there is a distributively terminating solitude detection algorithm which is correct, with probability at least $1-\epsilon$, for all rings of size $n \leq N$, and which sends only $O(n\log\log(\frac{1}{\epsilon}) + n\log(\frac{1}{\rho}))$ expected bits when $n \geq (\frac{1}{2}+\rho)N$, for any fixed $\rho > 0$ (Theorem 3.7). For sufficiently large $N$, a matching lower bound of $\Omega(n\log\log(\frac{1}{\epsilon}) + n\log(\frac{1}{\rho}))$ is proved (Theorems 5.4 and 5.5), for algorithms which know that $(\frac{1}{2}+\rho)N \leq n \leq N$, for fixed $0 < \rho < \frac{1}{2}$. The restriction to sufficiently large $N$ is necessary, since, when $n$ is known to lie between $N/2+1$ and $N$, it is possible to solve solitude detection with zero error probability in $O(n\log n)$ bits [1], and for small $N$ it is possible that $n\log n < n\log\log(\frac{1}{\epsilon}) + n\log(\frac{1}{\rho})$.

To summarize qualitatively, our results show that, although there can be no distributively terminating solution to solitude detection without some knowledge about the ring, even a large overestimate of ring size permits a distributively terminating solution. In addition, the complexity of the solution is very insensitive to the degree of overestimation.

## 3 Solitude Detection Algorithms

In the proofs of the following four theorems, algorithms are described which solve four different versions of probabilistic solitude detection. In all cases, the size of the ring, denoted by $n$, is not known exactly by the individual processors. Each algorithm uses $O(n)$ expected bits when there are two or more contenders.

Messages travel counter-clockwise around the ring. Say that a processor receives messages from its left, and sends them to the right.

The simplest case is presented in Theorem 3.1 which assumes only a unidirectional ring of identical processors with no additional knowledge about the ring. Specifically, processors cannot assume any upper bound on the size of the ring. As pointed out in section 2, only nondistributive termination is possible in this case. In Theorem 3.2, processors are given an

upper bound on the number of contenders. Because of this upper bound, distributive termination can be achieved. For Theorem 3.4, processors are given an upper bound, $N$, on the size of the ring. Theorem 3.7 applies when processors know that $(\frac{1}{2}+\rho)N \leq n \leq N$ for some $\rho > 0$.

**Theorem 3.1:** There is a nondistributively terminating probabilistic solitude detection algorithm that communicates $O(n \log(\frac{1}{\epsilon}))$ bits in the worst case, ($O(n)$ bits in the expected case when there are two or more contenders) verifies solitude with certainty, and erroneously asserts solitude with probability at most $\epsilon$.

**Proof:** In the basic algorithm, each contender flips an unbiased coin $t = \lceil \log(\frac{1}{\epsilon}) \rceil$ times, and sends the sequence of $t$ coin tosses to its neighbour. Non-contenders forward the packages of coin tosses to the next contender. A contender which receives a sequence different from the one it sent concludes that it is not alone, and sends an alarm. A contender which receives a sequence of $t$ flips identical to what it sent concludes that it is alone, but is willing to change its mind on receipt of an alarm. Alarms are forwarded until received by a processor which has already sent an alarm.

Clearly, if there is only one contender then after exactly $nt$ bits have been transmitted the algorithm terminates correctly, though termination cannot be detected by the processors. Suppose there are two or more contenders. If any contender discovers that it is not alone, then it will send an alarm and eventually all contenders will be alerted. The algorithm reaches the wrong conclusion only if all contenders send and receive the same sequence of coin tosses. Thus $\Pr(\text{error}) \leq 2^{-t} \leq \epsilon$.

This algorithm always sends $nt$ bits. A simple modification reduces the expected cost to $O(n)$ bits when there are two or more contenders. Rather than sending the coin tosses as a package, alternately send and receive them, one at a time. As soon as an inconsistency is detected, or an alarm arrives, send an alarm and abort the algorithm. Then each processor sends $O(1)$ expected bits when there are two or more contenders. ■

Now suppose processors are given an upper bound $C$ on the number of contenders $c$.

**Theorem 3.2:** There is a distributively terminating probabilistic solitude detection algorithm that communicates $O(n\sqrt{\log C} + n \log(\frac{1}{\epsilon}))$ bits in the worst case, ($O(n)$ bits in the expected case when there are two or more contenders) verifies solitude with certainty, and erroneously asserts solitude with probability at most $\epsilon$.

**Proof:** As in the proof of Theorem 3.1, contenders exchange coin tosses, alternately sending and receiving one bit at a time. Any contender which receives a message different from that most recently sent sends an alarm in place of its next coin toss. This continues until a maximum of $t = \left\lceil \sqrt{2 \log C} + \log(\frac{1}{\epsilon}) \right\rceil$ coin tosses have been sent by each contender. Contenders which received a sequence of $t$ flips that matched those transmitted conclude that they are alone. In contrast to the previous algorithm, the conclusion is irrevocable here.

As before, if there is only one contender then after exactly $nt$ messages have been transmitted the algorithm terminates correctly. Suppose that the number of contenders $i$ is at least two. Then, as before, the expected bit complexity is $O(n)$. We must show that the probability of error is at most $\epsilon$.

A contender is said to be *fooled* for $k$ flips if its first $k$ outputs match its first $k$ inputs. Let $Q_0, Q_1, \ldots, Q_{c-1}$ denote the contenders in sequence around the ring. The algorithm terminates incorrectly if any one of the contenders is fooled for $t$ flips.

If any one processor, say $Q_j$, is fooled for $t$ flips then it must be the case that processor $Q_{j-k}$ is fooled for $t-k$ flips, for $1 \leq k \leq t-1$. (Subscripts are implicitly reduced modulo $c$). If $c \geq t$ then all of these flips are independent and hence

$$\Pr(Q_j \text{ is fooled for } t \text{ flips}) < 2^{-t^2/2}.$$

On the other hand, if $c < t$ then at least $ct/2$ of these flips are independent and hence

$$\Pr(Q_j \text{ is fooled for } t \text{ flips}) < 2^{-ct/2}.$$

In either case, it follows that if $t \geq \sqrt{2 \log C} + \log(\frac{1}{\epsilon})$ then

$$\Pr(Q_j \text{ is fooled for } t \text{ flips}) < \epsilon/c$$

and hence,

$$\Pr(\text{some processor is fooled for } t \text{ flips}) < \epsilon.$$

■

Theorem 3.2 at once gives an $O(n\sqrt{\log N} + n \log(\frac{1}{\epsilon}))$ upper bound for solitude detection when an upper bound $N$ on the ring size is known by all processors. This upper bound can be sharpened to $O(n\sqrt{\log(\frac{N}{n})} + n \log(\frac{1}{\epsilon}))$ by using the following idea. Each contender $Q_j$ forms an estimate $\hat{g}_j$ of the size of the gap separating $Q_j$ from the nearest contender to the left. Then $N/\hat{g}_j$ can be used as an estimated upper bound on the number of contenders. (Here, each contender makes the bold assumption that all gaps are equal. It is remarkable that the algorithm performs well when the gaps are not at all

equal.) This upper bound is then used in an algorithm essentially the same as the one described in the proof of Theorem 3.2.

The basic gap estimation algorithm proceeds as follows. Let the contenders be $Q_0, \ldots, Q_{c-1}$, counter-clockwise around the ring. Let $g_j$ be the number of processors in the interval $[Q_{j-1}, Q_j)$. Gap estimation proceeds in rounds, starting with round 0. In round $t$, each passive processor tosses a coin with success probability $\min(2^{t^2}/N, 1)$. Each contender $Q_{j-1}$ sends a message across the gap to its right, informing $Q_j$ whether there were any successes in the gap. If there were no successes, then $Q_j$ proceeds to the next round. If there was a success in the gap to its left in round $t$, then $Q_j$ sets its estimate $\hat{g}_j = N\,2^{-(t+1)^2}$, sends a message indicating the end of the gap estimation phase, and waits to receive such a message. Such messages end gap estimation for any processor receiving them, and are forwarded until they reach a processor which started one. Not all contenders $Q_j$ will produce an estimate $\hat{g}_j$, but at least one must do so. Any contender which does not produce an estimate can be sure that it is not alone, so we can ignore such processors.

**Lemma 3.3:**

(a) $E(\hat{g}_j) \leq g_j$.

(b) If $c = 1$ then $E(\sqrt{\log(N/\hat{g}_0)}) = O(\sqrt{\log(\frac{N}{n})})$.

(c) If $c = 1$, the expected number of bits communicated during the estimation of $\hat{g}_0$ is $O(n\sqrt{\log(\frac{N}{n})})$.

**Proof:** (a) $\Pr(\hat{g}_j = N\,2^{-(t+1)^2})$ is easy to bound crudely. If $g_j$ coins are tossed with probability of head $2^{t^2}/N$, then the probability of at least one head is at most $g_j\,2^{t^2}/N$, so

$$\Pr(\hat{g}_j = N\,2^{-(t+1)^2}) \leq g_j\,2^{t^2}/N.$$

Therefore

$$E(\hat{g}_j) \leq \sum_{t=0}^{\infty} g_j\,2^{-(2t+1)} < g_j.$$

(b) When there is a single contender, $T = \sqrt{\log(N/\hat{g}_0)}$ rounds are used for gap estimation. We bound $E(T)$, the expected stopping time. Suppose that $N\,2^{-k^2} < n \leq N\,2^{-(k-1)^2}$. The stopping times less than $k+3$ give contribution less than $k+2$ to the expectation. It remains to show that the later stopping times make only a small contribution. But $\Pr(T = k+2+j) \leq \Pr(\text{no successes at stage } k+1+j)$. This is zero if $2^{(k+j)^2} \geq N$, and

otherwise it is $(1 - 2^{(k+j)^2}/N)^n$. But $n > N2^{-k^2}$, so $\Pr(T = k + 2 + j) < \exp(-2^{2kj+j^2})$. So from $k+3$ to the end, the contribution to $E(T)$ is at most $\sum_{j=1}^{\infty}(k + 2 + j)\exp(-2^{2kj+j^2})$, which is $O(1)$.

(c) This follows easily from the fact that $E(T) = O(\sqrt{\log(\frac{N}{n})})$. ∎

**Theorem 3.4:** If an upper bound $N$ on ring size is known by all processors, there is a probabilistic solitude detection algorithm that communicates a mean of $O(n\sqrt{\log(\frac{N}{n})} + n\log(\frac{1}{\epsilon}))$ bits when there is a single contender ($O(n)$ bits for two or more contenders), terminates distributively, verifies solitude with certainty, and erroneously asserts solitude with probability at most $\epsilon$.

**Proof:** Contenders execute the gap estimation algorithm, with a small modification. Alternating with gap estimation rounds are coin tossing rounds. In each coin tossing round, each contender sends and receives a coin toss. If the coin toss received is different from that sent, then the contender aborts the algorithm, sends an alarm as before, and concludes that it is not alone. The purpose of the modification is to reduce the complexity when there are two or more contenders.

If contender $Q_j$ obtains gap estimate $\hat{g}_j$, it executes the algorithm of Theorem 3.2, with the number of coin flips $t = \log(2/\epsilon) + K_j$, where $K_j = \sqrt{2\log(N/\hat{g}_j)}$.

When there is a single contender, on the average $O(n\sqrt{\log(\frac{N}{n})} + n\log(\frac{1}{\epsilon}))$ bits are communicated, since by lemma 3.3(c), $O(n\sqrt{\log(\frac{N}{n})})$ bits on the average are communicated in the gap estimation phase, and by lemma 3.3(b), $E(K_0) = O(\sqrt{\log(\frac{N}{n})})$. When there are two or more contenders, each processor sends $O(1)$ bits in the gap estimation phase and $O(1)$ bits in later phases. It remains to show that the algorithm erroneously asserts solitude with probability at most $\epsilon$. This part of the analysis is a more elaborate version of the analysis in Theorem 3.2.

Let $c$ be the actual number of contenders. For any particular run of the gap estimation algorithm, let $S_1 = \{j : K_j \geq c\}$ and $S_2 = \{j : K_j < c\}$. Let $F_j$ be the probability that the contender $Q_j$ is fooled, given that the gap estimation algorithm has produced particular estimates.

**Claim 3.5:** If $j \in S_1$, then $F_j \leq (\epsilon/4)^{c-1}$.

**Proof:** Since $K_j \geq c$, in order for $Q_j$ to be fooled $K_j + \log(2/\epsilon)$ times, every contender must be fooled at least $1 + \log(2/\epsilon)$ times. Hence $F_j \leq 2^{-(c-1)\log(4/\epsilon)} = (\epsilon/4)^{c-1}$. ∎

**Claim 3.6:** If $j \in S_2$, then $F_j \leq \epsilon \hat{g}_j / 2N$, where $\hat{g}_j$ is the gap estimate produced in that run.

**Proof:** In order for $Q_j$ to be fooled, the $K_j$ distinct processors $Q_j$, $Q_{j-1}, \ldots, Q_{j-K_j+1}$ must be fooled a total of

$$\sum_{k=1}^{K_j} (\log(2/\epsilon) + k) \geq \log(2/\epsilon) + K_j^2/2$$

times. Hence

$$F_j \leq 2^{-(\log(2/\epsilon)+K_j^2/2)} = \epsilon \hat{g}_j / 2N.$$

■

If $Q_j$ does not produce a gap estimate, then $Q_j$ is not fooled. Suppose $Q_j$ produces an estimate. The probability that $Q_j$ is fooled is at most $(\epsilon/4)^{c-1} + (\epsilon/2N) E(\hat{g}_j \mid \hat{g}_j < c)$. But the conditional expectation of $\hat{g}_j$ is clearly at most $E(\hat{g}_j)$, which by Lemma 3.3(a) is at most $g_j$. So the probability that $Q_j$ is fooled is at most $(\epsilon/4)^{c-1} + \epsilon g_j / 2N$.

Hence the probability that some contender is fooled is at most $c(\epsilon/4)^{c-1} + (\epsilon/2N) \sum_j g_j$. Since $c(\epsilon/4)^{c-1} \leq \epsilon/2$ and $\sum_j g_j = n \leq N$, the probability that some contender is fooled is at most $\epsilon$. ■

When $n$ is known to lie between $(\frac{1}{2} + \rho)N$ and $N$, where $\rho > 0$, it is possible to improve further on the bit complexity of solitude detection; there is a solitude detection algorithm of expected bit complexity $O(n \log\log(\frac{1}{\epsilon}) + n \log(\frac{1}{\rho}))$ expected bits when there is one contender.

**Theorem 3.7:** Suppose each processor knows that the ring size $n$ satisfies $(\frac{1}{2} + \rho)N \leq n \leq N$ where $\rho > 0$. Then there is a solitude detection algorithm which terminates distributively, has one-sided error probability at most $\epsilon$ and sends $O(n \log\log(\frac{1}{\epsilon}) + n \log(\frac{1}{\rho}))$ expected bits when there is a single contender ($O(n)$ bits when there are two or more contenders).

**Proof:** We begin by producing an algorithm with two-sided error, and then show how to eliminate the error on one side.

Let $g_j$ be the distance from the $(j-1)^{st}$ to the $j^{th}$ contender counterclockwise around the ring. An obvious algorithm has the $j^{th}$ contender obtain counts $c_1 = g_j$ and $c_2 = g_{j-1}$, concluding that it is alone iff $c_1 = c_2 > N/2$. Implemented deterministically, that algorithm uses $O(n \log n)$ bits. The complexity can be reduced by obtaining estimates of the counts. Each contender starts a counter, which propagates to the next contender. Each non-contender increments the counter with probability $\lambda/N$, where $\lambda$ is a

chosen constant. Let $c_1$ be the count obtained. Each contender sends $c_1$ to the next contender, and receives $c_2$. It concludes that it is alone iff $c_1 = c_2 \geq s\lambda$, where $s = \frac{1+\rho}{2}$.

We can presume that $\rho$ is small. Then an appropriate choice of $\lambda$ is $\lambda = \frac{1}{s}\left\lceil \frac{5}{\rho^2} \ln \frac{2}{\epsilon\rho} \right\rceil$. Since each counter has an expected value less than $\lambda$, the expected bit complexity is $O(n \log \lambda) = O(n \log\log(\frac{1}{\epsilon}) + n \log(\frac{1}{\rho}))$. The complexity can be reduced to $O(n)$ when there are two or more contenders by starting the algorithm with a sequence of $\log\log(\frac{1}{\rho}\log(\frac{1}{\epsilon}))$ coin tosses, with the usual provisions for alarms. It remains only to bound the error probabilities.

**Claim 3.8:** Let $p_1$ be the probability that a sole contender erroneously concludes that it is not alone. Then $p_1 \leq \epsilon$.

**Proof:** Let $r = \frac{1}{2} + \rho$. The quantity $s\lambda$ is an integer, and we can presume without loss of generality that $rN$ is also an integer.

A sole contender is fooled if it gets an estimate less than $s\lambda$. The probability of that occurring is maximum for $n = rN$, so

$$p_1 \leq \sum_{k=0}^{s\lambda} \binom{rN}{k} \left(\frac{\lambda}{N}\right)^k \left(1 - \frac{\lambda}{N}\right)^{rN-k}.$$

The ratio between consecutive terms in the sum is at least $r/s = \frac{1+2\rho}{1+\rho}$. So the entire sum is at most $\frac{1+2\rho}{\rho} < \frac{2}{\rho}$ times its largest term. Using the approximations $\binom{n}{k} \leq n^k e^k k^{-k}$ and $\left(1 - \frac{\lambda}{N}\right)^{rN-k} \leq e^{-r\lambda}\left(\frac{N}{N-\lambda}\right)^k$ gives

$$p_1 \leq \left(\frac{2}{\rho}\right)\left(\frac{r}{s}\right)^{s\lambda}\left(\frac{N}{N-\lambda}\right)^{s\lambda} e^{(s-r)\lambda}.$$

We can presume that $N > \frac{16\lambda}{\rho^2}$, since otherwise an $O(n \log n)$ bit algorithm has the desired complexity, with zero error probability. So $\left(\frac{N}{N-\lambda}\right)^s < \frac{N}{N-\lambda} < 1 + \frac{\rho^2}{8}$. But $e^{s-r} = 1 - \frac{\rho}{2} + \frac{\rho^2}{8} - \cdots$ and $\left(\frac{r}{s}\right)^s = 1 + \frac{\rho}{2} - \frac{\rho^2}{8} + \cdots$. So

$$\begin{aligned} p_1 &\leq \left(\frac{2}{\rho}\right)\left(1 - \frac{\rho^2}{8} + \cdots\right)^\lambda \\ &\leq \epsilon \end{aligned}$$

for sufficiently small $\rho$. ∎

**Claim 3.9:** Let $p_2$ be the probability that some contender erroneously concludes that it is alone. Then $p_2 \leq \epsilon$.

**Proof:** Say that a contender is almost fooled if $c_1 + c_2 \geq 2s\lambda$. A contender erroneously concludes that it is alone only if it is almost fooled, so it suffices to bound the probability that some contender is almost fooled. That probability is maximized when there are just two contenders, since changing any one of $k \geq 3$ contenders to a non-contender can only increase the probability that some contender is almost fooled. In the case of two contenders, $c_1+c_2$ is just the total number of non-contenders which increment a counter. It suffices to consider the case $n = N$. Then

$$\begin{aligned} p_2 &\leq \sum_{k=2s\lambda}^{N} \binom{N}{k} \left(\frac{\lambda}{N}\right)^k \left(1 - \frac{\lambda}{N}\right)^{N-k} \\ &\leq e^{2s\lambda-\lambda}(2s\lambda)^{-2s\lambda}\left(\tfrac{N\lambda}{N-\lambda}\right)^{2s\lambda} \sum_{k\geq 2s\lambda} \left(\tfrac{N}{2s(N-\lambda)}\right)^{k-2s\lambda}. \end{aligned}$$

Now the approximations $\frac{N}{N-\lambda} \leq 1 + \frac{\rho^2}{8}$, $e^{2s-1} = 1 + \rho + \frac{\rho^2}{2} + \cdots$ and $\left(\frac{1+\rho^2/8}{1+\rho}\right)^{1+\rho} = 1 - \rho + \frac{\rho^2}{8} + \cdots$ gives

$$\begin{aligned} p_2 &\leq \left(1 - \frac{3\rho}{8} + \cdots\right)^{\lambda} \left(\frac{1+\rho}{\rho - \rho^2/8}\right) \\ &< \epsilon \end{aligned}$$

for sufficiently small $\rho$. ∎

The algorithm given has two-sided error. Errors when there actually is a single contender can be eliminated as follows. Run the above algorithm. If the outcome is "alone", then conclude alone. Otherwise, run the algorithm of Theorem 3.4, and adopt its conclusion. The modified algorithm does not err when there is one contender, and has error probability at most $2\epsilon$ when there are two or more contenders. Moreover, the modified algorithm is correct for all $n \leq N$. When $n \geq (\frac{1}{2}+\rho)N$, and there is in fact one contender, it runs in $O(n \log\log(\frac{1}{\epsilon}) + \epsilon n \log(\frac{1}{\epsilon}))$ expected bits. But $\epsilon \log(\frac{1}{\epsilon}) \leq \frac{1}{2}$ for $\epsilon \leq 1/4$. The results of section 5 imply that the modified algorithm is optimal, to within a constant factor, for $n \leq N/2$ and $n \geq (\frac{1}{2} + \rho)N$, for sufficiently large $N$. ∎

## 4 A Model for Solitude Detection Algorithms

Our objective is to study the inherent bit complexity of distributed probabilistic algorithms that verify solitude with a high probability of correctness on unidirectional rings. A distributed algorithm can be viewed as an assignment of processes to processors. So it suffices to model computations as a

ring of processes. In order to describe such a computation we first state some relevant attributes of a process, and deduce useful properties of sequences of processes. The relationship between algorithms and sequences of processes is then made precise in order to highlight the generality of the lower bounds which follow.

### 4.1 Processes

The following description of a process incorporates two non-restrictive assumptions, namely, that messages are self-delimiting, and that communication is message driven, with only one message sent in response to receipt of a message. What follows is a collection of definitions concerning processes and sequences of processes, then a statement of the relationship between a line of processes and the same sequence considered as a ring, and finally a number of tools that allow us to manipulate sequences of processes.

A *message* is an element of $M = \{0,1\}^* \cdot \square$. The symbol $\square$ is called the end of message marker. If $m$ is a message we denote by $\|m\|$ the length of the binary encoding of $m$, including the end marker, using an encoding scheme in which each symbol $\{0, 1, \square\}$ is encoded with two bits. A *communication event* is an element of $M \cup \{\triangle\}$. The null event $\triangle$ denotes the absence of an input or an output message and should be distinguished from the empty message '$\square$'.

Any even length sequence of communication events, $C = (e_1, e_2, \ldots, e_{2t})$ describes a (possible) *computation*. The subsequence $(e_1, e_3, \ldots e_{2t-1})$ is called the *input history* of $C$ and subsequence $(e_2, e_4, \ldots, e_{2t})$ is called the *output history*. Computation $C$ is said to be *reduced* if $C$ does not begin or end with a pair of null events. The null events are used to ensure that every input event has an associated output event and vice versa. Since we have restricted our attention to message driven processes, we can assume that input histories are elements of $\triangle^* M^*$.

If $h \in (M \cup \{\triangle\})^*$ is any history we denote by $|h|$ the length of $h$ and $\|h\|$ the *cost* of $h$, that is, the sum of the lengths of all encoded messages in $h$. Note that the last two bits of the encoded form of $h$ must encode a $\square$. Hence, the last two bits are redundant, and there are at most $2^{i-1}$ histories of cost at most $i$.

A (*probabilistic*) *process*, $\pi$, is modelled by an assignment of probabilities to reduced computations. Specifically, for each element $x \in M^*$, the set of all reduced computations with input history $\triangle^t x$, for $t \geq 0$, form a probability space. We denote by $h\,\{\pi\}\,h'$ the event that process $\pi$ produces a computation with output history $h'$, given that it has input history $h$.

On occasion we need to deal with unreduced computations. The notation is extended, by assigning identical probabilities to each of the events $h\{\pi\}h'$, $\triangle h\{\pi\}\triangle h'$ and $h\triangle\{\pi\}h'\triangle$, since those events are indistinguishable.

If $h$ is a history let $h_{(i)}$ denote the length $i$ prefix of $h$. By the sequential nature of communication, we have,

**Property 4.1:** For all histories $h$ and $h'$ and all $i \geq 1$, $\Pr(h\{\pi\}h') \leq \Pr(h_{(i)}\{\pi\}h'_{(i)})$.

We say that process $\pi$ is a *t-initiator* for $t \geq 1$ if $\sum_{h\in M^t} \Pr(\triangle^t\{\pi\}h) > 0$. All other processes are *non-initiators*. An initiator (i.e. a $t$-initiator for some $t \geq 1$) is just a process which has nonzero probability of sending at least one message before it receives one. If $\pi$ is a 1-initiator and $\pi$ is not a $t$-initiator for any $t \geq 2$ then $\pi$ is a *single initiator*. We assume that the contenders in any ring are just single initiators and all other processes are non-initiators.

If $\pi_1$ and $\pi_2$ are processes then their composition, denoted $\pi_1\pi_2$, is the process $\pi$ satisfying

$$\Pr(h\{\pi\}h') = \sum_{h''} \Pr(h\{\pi_1\}h'') \cdot \Pr(h''\{\pi_2\}h').$$

Thus, $\pi_1\pi_2$ is obtained by identifying the output of $\pi_1$ with the input of $\pi_2$. If exactly one of $\pi_1$ and $\pi_2$ are single initiators then so is $\pi$. The statement $\pi_1\pi_2$ *contains exactly one initiator* is used informally to mean that $\pi_1\pi_2$ is a single initiator.

If $\pi_1, \ldots, \pi_t$ is a sequence of processes, let $\pi_{i,j}$ denote the composition $\pi_i \cdots \pi_j$, for $1 \leq i \leq j \leq t$. It is convenient to view a sequences of processes $\pi_1, \ldots, \pi_t$ as a single process, namely the composition $\pi_{1,t}$. By abuse of notation, a sequence is frequently identified with its composition. The real difference between the two is that, although $\pi_{1,t}$ is a single process, with communication cost assigned only to its input and its output, the sequence $\pi_1, \ldots, \pi_t$ has communication cost assigned to each link from $\pi_i$ to $\pi_{i+1}$, for $1 \leq i < t$.

The notion of a computation can be extended to sequences of processes. A sequence $h_0, \ldots, h_t$ of histories describes a computation of the process line $\pi_{1,t}$ which is equivalent to the conjunction of the independent events $h_i\{\pi_{i+1}\}h_{i+1}$, for $0 \leq i < t$, in the appropriate product space. The *cost* of such a computation is given by $\sum_{i=1}^{t} \|h_i\|$. Note that $h_0$ does not contribute to the cost. If $\pi_1, \ldots, \pi_t$ are processes (or sequences of processes), let $h_0\{\pi_1\}h_1 \cdots \{\pi_t\}h_t$ denote the event described by sequence $h_0, \ldots, h_t$.

We distinguish a subset $M_a \subseteq M$ called *accepting messages*, and a subset $M_r \subseteq M$ called *rejecting* messages. A history is an *accepting history* (respectively, *rejecting history*) if and only if its *last* message is an accepting message (respectively, rejecting message). A computation $h_0, \ldots, h_t$ of $\pi_1, \ldots, \pi_t$ *asserts solitude* if *any* history $h_i$ where $\pi_i$ is an initiator, is an accepting history, and *asserts non-solitude* if *each* of the $h_i$ is a rejecting history. A process $\pi$ is said to *terminate distributively* if $\pi$ never outputs another message after having output an accepting message. The obvious bias in these definitions reflects the fact that we are addressing the complexity of an algorithm when there is one initiator. The definitions given here lead to slightly stronger results than the obvious unbiased ones would.

The preceding definitions allow us to study the behaviour of a sequence of processes *on a line* as a function of the behaviours of the individual processes. Our objective, however, is to study the behaviour of processes on a ring. Informally, process $\pi$ is on a ring if its output is fed back into its input. Fortunately, the essential properties of process rings are reflected by properties of associated process lines. The mapping from lines back to rings is characterized by two properties - one for distributively terminating and one for nondistributively terminating sequences. Let $h\,[\pi]\,*$ denote the event that given input $h$ the computation of $\pi$ asserts solitude. The event $h\,[\pi]\,h'$ denotes the conjunction of the events $h\,\{\pi\}\,h'$ and $h\,[\pi]\,*$.

**Property 4.2:** Let $\pi$ be any process. If $\Pr(\triangle^t h\,[\pi]\,h\triangle^t) = p$ for some $t \geq 1$, then with probability at least $p$, computations of $\pi$ on a ring assert solitude.

**Property 4.3:** Let $\pi$ and $\pi'$ be any distributively terminating processes. If $\Pr(\triangle^t\,[\pi]\,*) = p$ then, with probability at least $p$ computations of $\pi'\,\pi$ on a ring assert solitude.

The placement of nulls in Property 4.2 is important. For example, it is quite possible that, when given input history $h$, process $\pi$ produces output history $h$ with positive probability, that is, $\Pr(h\,\{\pi\}\,h) > 0$. But when $\pi$'s output is fed back into its input, $\pi$ produces no messages; $\pi$ is deadlocked, waiting for itself.

Property 4.2 is used to draw conclusions about nondistributively terminating processes on a ring. History $h$ appears in both the input and the output of $\pi$ but shifted by $t$ messages. This allows the individual messages of $h$ to be fed back into $\pi$ as input as soon as they are produced as output. Therefore none of the processes in $\pi$ can distinguish between computations on a line with input $\triangle^t h$ which produce output $h\triangle^t$ and computations on

a ring in which each process makes the corresponding probabilistic choices. Property 4.3 is used for conclusions that require the assumption of distributive termination. Suppose sequence $\pi$ asserts solitude with probability $p$ without any input. Then, when $\pi$ is a segment of a ring, with probability at least $p$ there is some initiator in $\pi$ with an accepting message in its output history. The distributive termination assumption then ensures that the entire ring asserts solitude.

The next property serves a complementary role to the previous two properties. It produces computations on lines from computations on rings.

**Property 4.4:** Let $\pi = \pi_1, \ldots, \pi_t$ be a single initiator process sequence. Suppose that with probability $p$ computations of $\pi$ on a ring assert solitude and some fixed process $\pi_i$ has a fixed output history $h$. Then $\Pr(\triangle h\,[\pi_{i+1,t}, \pi_{1,i}]\,h\triangle) = p$.

A sequence $\pi_{1,t}$ of processes is said to *assert solitude* (respectively, *non-solitude*) *on a ring* with probability $p$ if computations of $\pi_{1,t}$ on a ring assert solitude (respectively, non-solitude) with probability $p$.

Let $\chi$ be a number (which will be given a specific value whenever necessary) called the *cheapness threshold*. A computation of an arbitrary sequence of processes is said to be *cheap* if it has total cost at most $\chi$. Let $h\,\langle\pi_{1,t}\rangle\,*$ denote the event $h\,[\pi_{1,t}]\,* \,\&\, A$, where $A$ is the event that the computation of processor sequence $\pi_{1,t}$ with input history $h$ is cheap. Let $h\,\langle\pi_{1,t}\rangle\,h'$ denote the conjunction of the events $h\,\langle\pi_{1,t}\rangle\,*$ and $h\,\{\pi\}\,h'$. Similarly, $h_0\,\langle\pi_1\rangle\,h_1 \cdots \langle\pi_t\rangle\,h_t$ denotes the event $h_0\,\langle\pi_{1,t}\rangle\,* \,\&\, h_0\,\{\pi_1\}\,h_1 \cdots \{\pi_t\}\,h_t$.

The following lemma shows that if the expected cost of computations of a single initiator sequence $\pi_{1,t}$ on a ring is bounded, then inexpensive computations of the form $\triangle h\,\{\rho_{1,t}\}\,h\triangle$ occur with reasonably high probability, where $\rho_{1,t}$ is some cyclic permutation of $\pi_{1,t}$ and $h$ is some *fixed* element of $M^*$.

**Lemma 4.5:** Let $\pi_1, \ldots, \pi_t$ be any single initiator process sequence. Suppose that $\pi_{1,t}$ asserts solitude on a ring with probability at least $1 - \epsilon$. Suppose also that the expected cost of computations of $\pi_{1,t}$ that assert solitude is at most $\mu t$ bits. Then there exists an integer $i$, where $1 \leq i \leq t$, and a history $h$ with $\|h\| \leq 4\mu$ such that

$$\Pr(\triangle h\,\langle\pi_{i+1,t}\,\pi_{1,i}\rangle\,h\triangle) \geq (1-\epsilon)2^{-(4\mu+1)}$$

where the cheapness threshold $\chi$ has the value $2\mu t$.

**Proof:** Since the expected cost of accepting computations is at most $\mu t$, the probability that an arbitrary computation of $\pi_{1,t}$ asserts solitude and communicates fewer than $2\mu t$ bits is at least $(1-\epsilon)/2$.

Let $e_i$ denote the expected number of bits in the output history of process $\pi_i$, over all accepting computations of $\pi_{1,t}$ with costs at most $2\mu t$ bits. For some $i$, $e_i \leq 2\mu$, and hence with probability at least $(1-\epsilon)/4$, $\pi_i$ has an output history with no more than $4\mu$ bits and the entire computation has cost at most $2\mu t$ and the computation asserts solitude. But there are fewer than $2^{4\mu-1}$ distinct histories with at most $4\mu$ bits and hence, with probability at least $(1-\epsilon)2^{-(4\mu+1)}$, $\pi_i$ outputs some fixed history $h$, where $\|h\| \leq 4\mu$ and the entire accepting computation has cost at most $2\mu t$. Thus, using Property 4.4 and conditioning over cheap computations, $\Pr(\triangle h \langle \pi_{i+1,t}\, \pi_{1,i} \rangle\, h\triangle) \geq (1-\epsilon)2^{-(4\mu+1)}$ where $\chi = 2\mu t$. ■

A process sequence $\pi$ can be replicated to form the new sequence $\pi^k$ consisting of the concatenation of $k$ copies of $\pi$. The following two lemmas express the probability that $\pi^k$ asserts solitude as a function of the probability that $\pi$ does. The proofs of both lemmas follow from applications of elementary probability theory, and are therefore omitted.

**Lemma 4.6:** If $\Pr(\triangle h\, [\pi]\, h\triangle) = p$ then $\Pr(\triangle^k h\, [\pi^k]\, h\triangle^k) \geq p^k$.

**Lemma 4.7:** If $\Pr(\triangle^t\, [\pi]\, *) = p$ then $\Pr(\triangle^{tk}\, [\pi^k]\, *) \geq 1-(1-p)^k$.

At the heart of our lower bound proofs is the observation that a sequence of histories of sufficiently small total cost must contain the same history twice. The following lemma refines that observation to a probabilistic setting, and provides information about the separation between the repetitions.

**Lemma 4.8:** Let $\pi_{1,t}$ be any single initiator sequence of processes. Let $\sigma$ and $\tau$ be positive integers satisfying

(a) $\tau \geq 36^2$,

(b) $24\chi < t \log \tau$, and

(c) $t > \tau\sigma$.

Let $h$ be any element of $M^*$. Then there exist integers $i$ and $j$ where $1 < i < j \leq t$ and a history $h^*$ such that

i) $\sigma \leq j - i < \tau\sigma$ and

ii) $\Pr(\triangle h \langle \pi_{1,i-1} \rangle h^* \langle \pi_{i,j-1} \rangle h^* \langle \pi_{j,t} \rangle h\triangle) \geq \tau^{-1} \Pr(\triangle h \langle \pi_{1,t} \rangle h\triangle)$.

**Proof:** Suppose without loss of generality that $\xi = \Pr(\triangle h \langle \pi_{1,t} \rangle h\triangle) > 0$. For $1 \leq i < t$, let $e_i$ be the expected cost of the output history of $\pi_i$, conditional on $\triangle h \langle \pi_{1,t} \rangle h\triangle$. That is, $e_i = (1/\xi) \sum_{h_i} \|h_i\| \cdot \Pr(\triangle h \langle \pi_{1,i} \rangle h_i \langle \pi_{i+1,t} \rangle h\triangle)$. Let $\delta = \log \tau$.

If $e_i < \delta/8$, say that link $i$ is *cheap*. If $\|h\| < \delta/4$, say that history $h$ is *short*. Suppose that link $i$ is cheap and let $h_i^*$ be the short history which maximizes $\Pr(\triangle h \langle \pi_{1,i} \rangle h_i^* \langle \pi_{i+1,t} \rangle h\triangle)$. Each history has either one $\triangle$ at its start or one $\triangle$ at its end, which is not included in its encoding. Therefore there are fewer than $2^{\delta/4} = \tau^{1/4}$ short histories. It follows that $\Pr(\triangle h \langle \pi_{1,i} \rangle h_i^* \langle \pi_{i+1,t} \rangle h\triangle) \geq \frac{\xi}{2\tau^{1/4}}$, since otherwise $e_i > (\delta/4)(1 - \frac{\tau^{1/4}}{2\tau^{1/4}}) = \delta/8$, contradicting the cheapness of link $i$.

For $1 \leq j < t - (\tau - 1)\sigma$, let $B_j = \{j + k\sigma \ : \ 0 \leq k < \tau\}$. Choose a $j$ such that at least $1/3$ of the $\tau$ members of $B_j$ are cheap links. Such a $j$ must exist, since otherwise at least $2/3$ of at least $\tau\sigma\lfloor (t-1)/\tau\sigma \rfloor \geq t/2$ links are not cheap, contradicting the assumption that $\sum_{i=1}^{t} e_i \leq \chi < t\delta/24$.

Again, because there are at most $\tau^{1/4}$ short histories, at least $w = \lceil \tau^{3/4}/3 \rceil$ of the cheap members $k$ of $B_j$ have identical $h_k^*$. Let $i_1, \ldots, i_w$ be $w$ such members, and let $h^*$ denote the common history. Let $D_s$ denote the event $\triangle h \langle \pi_{1,i_s-1} \rangle h^* \langle \pi_{i_s,t} \rangle h\triangle$, for $1 \leq s \leq w$. By the inclusion-exclusion principle,

$$\sum_{r<s} \Pr(D_r \,\&\, D_s) \geq \left( \sum_s \Pr(D_s) \right) - \xi.$$

Since $Pr(D_s) \geq \frac{\xi}{2\tau^{1/4}}$, there must exist $r$ and $s$ such that

$$\begin{aligned} \Pr(D_r \,\&\, D_s) &\geq \frac{\left(\frac{w}{2\tau^{1/4}} - 1\right)\xi}{\binom{w}{2}} \\ &\geq \frac{\xi}{\tau}, \qquad \text{for } \tau \geq 36^2. \end{aligned}$$

Thus $\Pr(D_r \,\&\, D_s) \geq \tau^{-1} \Pr(\triangle h \langle \pi_{1,t} \rangle h\triangle)$. So it suffices to choose $i = i_r$ and $j = i_s$. ∎

Lemma 4.8 only locates repeated histories. The following property uses repeated histories to relate lines of different sizes.

**Property 4.9:** Let $\pi_1, \ldots, \pi_t$ be a single initiator sequence of processes and let $1 < i < j \leq t$. Let $h$ and $h^*$ be histories. Let $p = \Pr(\triangle h \langle \pi_{1,i-1} \rangle h^* \langle \pi_{i,j-1} \rangle h^* \langle \pi_{j,t} \rangle h\triangle)$. Then $\Pr(\triangle h \langle \pi_{1,i-1} \rangle h^* \langle \pi_{j,t} \rangle h\triangle) \geq p$ and $\Pr(h^* \{\pi_{i,j-1}\} h^*) \geq p$.

**Proof:** Any computation satisfying $\triangle h \{\pi_{1,i-1}\} h^* \{\pi_{i,j-1}\} h^* \{\pi_{j,t}\} h\triangle$ includes disjoint subcomputations satisfying $\triangle h \{\pi_{1,i-1}\} h^*$, $h^* \{\pi_{i,j-1}\} h^*$ and $h^* \{\pi_{j,t}\} h\triangle$. Clearly if the total computation is cheap then each piece is. Furthermore, histories preceding an initiator must differ from those following an initiator because their first events differ. Therefore the initiator cannot be in the segment $\pi_i \cdots \pi_{j-1}$ and the initiator's history remains the same accepting history. ∎

### 4.2 Solitude Detection Algorithms

Let $\mathcal{A}$ denote the set of all probabilistic processes. A distributed probabilistic algorithm is normally specified by assigning a fixed initiating process from $\mathcal{A}$ to all contenders and a fixed non-initiating process to all non-contenders. (Certainly all of the algorithms of Section 3 satisfy this property). It is convenient to generalize this notion of a distributed algorithm to permit assignments from an arbitrary set of processes. In fact, we define an *algorithm* to be just the set $\alpha \subseteq \mathcal{A}$ available for assignment.

This generalization gives algorithms both probabilistic and nondeterministic attributes. Like conventional probabilistic algorithms, an algorithm is said to solve a problem with probability $p$ if for *all* possible process assignments, the resulting computation reaches the desired conclusion with probability at least $p$. Like conventional nondeterministic algorithms, it is said to solve a problem efficiently if for *some* choice of process assignments the resulting computation has low expected cost.

More formally, let $[a, b]$ denote an interval of positive integers and let $\mathcal{R}_{[a,b]}$ denote the class of all rings of size $n$ where $n \in [a, b]$. If $\alpha \subseteq \mathcal{A}$ is an algorithm, we denote by $\alpha^n$ the set of sequences $\pi_1, \ldots, \pi_n$ where $\pi_i \in \alpha$ for $1 \leq i \leq n$. $\alpha^{[a,b]}$ denotes $\bigcup_{n\in[a,b]} \alpha^n$. $\alpha^{[a,b]}$ corresponds to the set of all assignments of processes in $\alpha$ to processors on rings in the set $\mathcal{R}_{[a,b]}$.

This paper is concerned with three closely related problems; *solitude detection, solitude verification* and *weak solitude verification* defined as follows.

**Solitude Detection.** $\alpha$ solves *solitude detection* with confidence $1 - \epsilon$ on rings in $\mathcal{R}_{[a,b]}$ if:

i) For any element of $\alpha^{[a,b]}$ containing exactly one initiator, solitude is asserted with probability at least $1 - \epsilon$.

ii) For any element of $\alpha^{[a,b]}$ containing more than one initiator, nonsolitude is asserted with probability at least $1 - \epsilon$.

**Solitude Verification.** $\alpha$ solves *solitude verification* with confidence $1-\epsilon$ on rings in $\mathcal{R}_{[a,b]}$ if:

i) For any element of $\alpha^{[a,b]}$ containing exactly one initiator, solitude is asserted with probability at least $1-\epsilon$.

ii) For any element of $\alpha^{[a,b]}$ containing more than one initiator, solitude is not asserted, with probability at least $1-\epsilon$.

**Weak Solitude Verification.** $\alpha$ solves *weak solitude verification* with confidence $1-\epsilon$ on rings in $\mathcal{R}_{[a,b]}$ if:

> For any element of $\alpha^{[a,b]}$ containing more than one initiator, solitude is not asserted, with probability at least $1-\epsilon$.

These definitions make it clear that weak solitude verification is a subproblem of solitude detection. Lower bounds for weak solitude verification imply lower bounds for solitude detection. We recognize that nonsolitude can be ascertained with a low expected cost. But the problem we focus on is the cost of verifying that with high probability there is only one initiator. Therefore the complexity of weak solitude verification is defined to be the expected complexity when solitude is correctly asserted. (In the case of algorithms which never correctly assert solitude, the complexity of weak solitude verification is undefined.)

Let $\alpha$ be an algorithm that solves weak solitude verification with confidence $1-\epsilon$. $\alpha$ has *complexity $f(n)$ on rings of size $n$* if: for every $\pi_{1,n} \in \alpha^n$ with exactly one initiator, if solitude is asserted with probability at least $1-\epsilon$, then the expected number of bits communicated by $\pi_{1,n}$ on a ring when solitude is asserted is at least $f(n)$.

The next section provides lower bounds for some versions of weak solitude verification obtained by varying the size of the interval $[a,b]$ which describes the class of rings for which an algorithm is required to work.

## 5 Lower Bounds

The following theorems, together with the upper bounds of section 3 completely characterize (to within a constant factor) the bit complexity of solitude detection with confidence $1-\epsilon$ for various classes of rings. The lower bounds all proceed similarly. We assume that there is some element of $\alpha^{[a,b]}$ for which the complexity of weak solitude verification is smaller than the desired threshold. Thus there is some sequence $\pi_1, \ldots, \pi_n$ with a single initiator and with $n \in [a,b]$ which asserts solitude with high probability and

with low expected communication complexity. Lemma 4.5 is used to create a line of processes from the ring of processes. If necessary lemma 4.8 and property 4.9 are used to collapse this sequence to a shorter one. Lemma 4.6 or lemma 4.7 is used to replicate the shorter sequence. The result is a new sequence with more than one initiator and size still in $[a, b]$. But the lemmas imply that this sequence erroneously asserts solitude with too high a probability. Finally property 4.2 or 4.3 is used to conclude that this probability of error carries over to computations of the sequence on a ring. Since the new sequence is also in $\alpha^{[a,b]}$, a contradiction is presented to the requirement of weak solitude verification on rings with more than one initiator.

In the interest of ease of presentation, little effort is made to establish strong constants.

**Theorem 5.1:** Let $\alpha$ be a distributively terminating algorithm which solves weak solitude verification with confidence $1-\epsilon$ on rings in $\mathcal{R}_{[1,N]}$. Then the complexity of $\alpha$ on rings of size $n \in [1, N]$ is $\Omega\left(n\sqrt{\log(N/n)}\right)$ bits for any $\epsilon < \frac{1}{2}$.

**Proof:** Let $\pi_1, \ldots, \pi_n$ be an element of $\alpha^{[1,N]}$ with exactly one initiator. Suppose $\pi_{1,n}$ asserts solitude with probability at least $1 - \epsilon$ and that the expected cost of computations of $\pi_{1,n}$ that assert solitude is at most $\mu n$.

First suppose that $\mu < \frac{1}{2}$. Then, with probability greater than $\frac{1}{2}$, some processor communicates nothing. But, by the message driven nature of computations, it follows that with probability greater than $\frac{1}{2}$ the initiator concludes that it is alone without receiving any communication. Thus, for computations of $\alpha$ on rings with two or more initiators, some initiator erroneously concludes that it is alone with probability greater than $\frac{1}{2}$.

Hence we have a linear lower bound, and it suffices to assume that $\lfloor N/n \rfloor \geq 16$. Suppose $\mu < \left(\sqrt{\log(N/n)}\right)/5$. By lemma 4.5 there is a cyclic permutation $\rho_{1,n} = \pi_{i+1,n}\,\pi_{1,i}$ of $\pi_{1,n}$ and a history $h$ with $\|h\| \leq 4\mu$ such that $\Pr(\triangle h\,[\rho_{1,n}]\,h\triangle) \geq (1-\epsilon)2^{-4\mu-1}$. Now splice together $t = \max(1, |h|) \leq 2\mu$ copies of $\rho_{1,n}$. Using property 4.1,

$$\begin{aligned}
\Pr(\triangle^t\,[\rho_{1,n}^t]\,*) &\geq \Pr(\triangle^t\,[\rho_{1,n}^t]\,h) \\
&\geq \prod_{j=0}^{t-1} \Pr(\triangle^{t-j} h_{(j)}\,[\rho_{1,n}]\,\triangle^{t-j-1} h_{(j+1)}) \\
&\geq \left((1-\epsilon)2^{-4\mu-1}\right)^t \\
&> 2^{-4t(\mu+1)}
\end{aligned}$$

Now splice together $k = \lfloor N/(nt) \rfloor$ blocks of $\rho_{1,n}^t$. By lemma 4.7

$$\Pr(\triangle^{tk}[\rho_{1,n}^{tk}]*) \geq 1 - \left(1 - 2^{-4t(\mu+1)}\right)^k$$

But $\mu < \left(\sqrt{\log(N/n)}\right)/5$ and $t \leq 2\mu$. So $k = \lfloor N/(nt) \rfloor > 2^{4t(\mu+1)}$ if $\mu > \frac{1}{2}$. Therefore $\Pr(\triangle^{tk}[\rho_{1,n}^{tk}]*) > 1 - \frac{1}{e} > \frac{1}{2}$. By property 4.3, $\rho_{1,n}^{tk}$ errs with probability at least $\frac{1}{2}$. ■

**Theorem 5.2:** Let $\alpha$ be any (even nondistributively terminating) algorithm which solves weak solitude verification with confidence $1 - \epsilon$ on rings in $\mathcal{R}_{[1,N]}$. Then the complexity of $\alpha$ on rings of size $n \in [1, N/2]$ is $\Omega\left(n \log(\frac{1}{\epsilon})\right)$ bits.

**Proof:** Let $\pi_1, \ldots, \pi_n$ be an element of $\alpha^{[1,N/2]}$ with exactly one initiator. Suppose $\pi_{1,n}$ asserts solitude with probability at least $1 - \epsilon$ and that the expected cost of computations of $\pi_{1,n}$ that assert solitude is at most $\mu n$ with $\mu < (\log(\frac{1}{\epsilon}))/16$. Since a linear lower bound follows easily, (as in theorem 5.1), it suffices to assume $\epsilon < 1/20$. By lemma 4.5 there is a cyclic permutation $\rho_{1,n} = \pi_{i+1,n}\pi_{1,i}$ of $\pi_{1,n}$ and a history $h$ with $\|h\| \leq 4\mu$ such that $\Pr(\triangle h[\rho_{1,n}]h\triangle) \geq (1-\epsilon)2^{-4\mu-1}$. Consider the sequence $\rho_{1,n}\rho_{1,n}$ formed by splicing together two copies of $\rho_{1,n}$. By lemma 4.6

$$\begin{aligned}
\Pr(\triangle\triangle h[\rho_{1,n}\rho_{1,n}]h\triangle\triangle) &\geq \left((1-\epsilon)2^{-4\mu-1}\right)^2 \\
&\geq \frac{(1-\epsilon)^2}{4}\sqrt{\epsilon} \\
&> \epsilon
\end{aligned}$$

if $\epsilon < 1/20$.

By property 4.2, $\rho_{1,n}\rho_{1,n}$ errs on a ring with probability more than $\epsilon$ even under nondistributive termination. ■

**Corollary 5.3:** When it is known that $n \leq N$, and in fact $n \leq N/2$, the expected bit complexity of solitude detection is $\Theta(n\sqrt{\log(\frac{N}{n})} + n\log(\frac{1}{\epsilon}))$, when there is in fact a single contender.

The preceding lower bound holds only when the actual ring size $n$ is at most half of the known upper bound $N$. The next result concerns the case when $n > N/2$. Although theorem 5.4 holds when it is known that $cN \leq n \leq N$ for any $c < 1$, it is stated and proved for the case when $3N/4 \leq n \leq N$.

**Theorem 5.4:** Let $\alpha$ be any (even nondistributively terminating) algorithm which solves weak solitude verification with confidence $1-\epsilon$ on rings in $\mathcal{R}_{[3N/4,N]}$. Then the complexity of $\alpha$ on rings of size $n \in [3N/4, N]$ is $\Omega\left(n \min(\log\log(\frac{1}{\epsilon}), \log N)\right)$ bits.

**Proof:** Since the theorem is trivially true for moderately sized $N$ and $1/\epsilon$, assume that $N$ is very large and $\epsilon$ is very small. Let $\pi_1, \ldots, \pi_n$ be an element of $\alpha^{[3N/4,N]}$ with exactly one initiator. Suppose $\pi_{1,n}$ asserts solitude with probability at least $1-\epsilon$ and that the expected cost of computations of $\pi_{1,n}$ that assert solitude is at most $\mu n$ where $\mu < \frac{1}{120}\min\left(\log\log(\frac{1}{\epsilon}), \log N\right)$.

By lemma 4.5 there is a cyclic permutation $\rho_{1,n} = \pi_{i+1,n}\pi_{1,i}$ of $\pi_{1,n}$ and a history $h$ with $\|h\| \leq 4\mu$ such that $\Pr(\triangle h \langle \rho_{1,n} \rangle h \triangle) \geq (1-\epsilon)2^{-4\mu-1}$ where the cheapness threshold is $\chi = 2\mu n$. The sequence $\rho_{1,n}$ is collapsed by repeated application of lemma 4.8 and property 4.9. Let $\tau = \lfloor 2^{97\mu} \rfloor$ and $\sigma = \left\lfloor \frac{N}{8\tau} \right\rfloor$. Since $\mu < \frac{1}{120}\log N$ it is easily seen that $\sigma > 1$ for large $N$ and that the conditions of lemma 4.8 are satisfied for any $t > N/2$. During each collapsing step, a segment of length less than $\tau\sigma \leq N/8$ is removed, so it is assured that a new sequence $z$ of length $m \in [3N/8, N/2]$ can be achieved. Since at least $\sigma$ is removed at each step, in the worst case no more than $N/(2\sigma)$ applications are required. Therefore

$$\Pr(\triangle h \langle z \rangle h \triangle) \geq \tau^{-N/(2\sigma)}(1-\epsilon)2^{-4\mu-1}$$

A single replication results in the sequence $zz$ of length $2m \in [3N/4, N]$. By lemma 4.6,

$$\begin{aligned}
\Pr(\triangle\triangle h [zz] h \triangle\triangle) &\geq \left(\tau^{-N/(2\sigma)}(1-\epsilon)2^{-4\mu-1}\right)^2 \\
&> \left(\tau^{-8\tau}2^{-4(u+1)}\right)^2 \\
&\geq \epsilon
\end{aligned}$$

because $\mu < \frac{1}{120}\log\log(\frac{1}{\epsilon})$.

By property 4.2, $zz$ errs on a ring with probability more than $\epsilon$ even under nondistributive termination. ∎

Lemma 4.8 also provides the tool to show that the $\log(\frac{1}{\rho})$ term in the complexity of the algorithm for all $n$ between $(\frac{1}{2}+\rho)N$ and $N$ where $\rho > 0$ is really necessary.

**Theorem 5.5:** Let $\alpha$ be any (even nondistributively terminating) algorithm which solves weak solitude verification with confidence $1-\epsilon$ on rings in $\mathcal{R}_{[(\frac{1}{2}+\rho)N,N]}$ where $\rho > 0$ and $(\frac{1}{2}+\rho)N$ is an integer. Then the complexity of $\alpha$ on a ring of size $n = (\frac{1}{2}+\rho)N$ is $\Omega\left(n \min(\log(\frac{1}{\rho}), \log(\frac{1}{\epsilon}), \log N)\right)$ bits.

**Proof:** The theorem is true for $N$, $\rho$, and $\epsilon$ of moderate size so assume that $N$ is very large and that both $\rho$ and $\epsilon$ are very small. In particular, assume $M = \left\lfloor \min(\log(\frac{1}{\rho}), \log(\frac{1}{\epsilon}), \log N) \right\rfloor \geq 32$. Let $\pi_1, \ldots, \pi_n$ be an element of $\alpha^{(\frac{1}{2}+\rho)N}$ with exactly one initiator. Suppose $\pi_{1,n}$ asserts solitude with probability at least $1-\epsilon$ and that the expected cost of computations of $\pi_{1,n}$ that assert solitude is at most $\mu n$ where $\mu < M/145$.

By lemma 4.5 there is a cyclic permutation $\rho_{1,n} = \pi_{i+1,n}\pi_{1,i}$ of $\pi_{1,n}$ and a history $h$ with $\|h\| \leq 4\mu$ such that $\Pr(\triangle h \langle \rho_{1,n} \rangle h\triangle) \geq (1-\epsilon)2^{-4\mu-1}$ where the cheapness threshold is $2\mu n$. The sequence $\rho_{1,n}$ is first collapsed by using just one application of lemma 4.8 and property 4.9. Let $\sigma = \left\lfloor \frac{N}{2^{M-1}} \right\rfloor$ and $\tau = 2^{M/3}$. Then $\sigma \geq \rho N$ and $\tau\sigma < N/4$ so with one collapsing operation the resulting sequence $z$ will have length $m$ well within $[\left(\frac{1}{4}+\frac{\rho}{2}\right)N, N/2]$. The preconditions for lemma 4.8 are easily checked. Therefore

$$\begin{aligned} \Pr(\triangle h \langle z \rangle h\triangle) &\geq \tau^{-1}(1-\epsilon)2^{-4\mu-1} \\ &> \tau^{-1}2^{-4(\mu+1)} \end{aligned}$$

A single replication results in the sequence $zz$ of length $2m \in [\left(\frac{1}{2}+\rho\right)N, N]$. By lemma 4.6, $\Pr(\triangle\triangle h\,[zz]\,h\triangle\triangle) > \left(\tau^{-1}2^{-4(\mu+1)}\right)^2$. But

$$\begin{aligned} \log\left(\tau^{-1}2^{-4(\mu+1)}\right)^{-2} &= 2\log\tau + 8\mu + 8 \\ &< \frac{2}{3}M + \frac{8}{145}M + 8 \\ &< \log(\frac{1}{\epsilon}) \end{aligned}$$

since $M \leq \log(\frac{1}{\epsilon})$. Therefore $\Pr(\triangle\triangle h\,[zz]\,h\triangle\triangle) > \epsilon$.

By property 4.2, $zz$ errs on a ring with probability more than $\epsilon$ even under nondistributive termination. ∎

**Corollary 5.6:** When it is known that $(\frac{1}{2}+\rho)N \leq n \leq N$, the expected bit complexity of solitude detection is $\Theta(n \min(\log\log(\frac{1}{\epsilon}), \log N) + n \min(\log(\frac{1}{\rho}), \log(\frac{1}{\epsilon}), \log N))$, when there is in fact a single contender.

## 6 One-sided Versus Two-sided Error

The preceding lower bounds hold for solitude verification algorithms that allow the probability of error to be at most $\epsilon$ either when there is one or more than one contender. A natural generalization might permit probability of error at most $\delta$ when there is one contender and probability of error at most

$\epsilon$ when there is more than one contender. The upper bounds, on the other hand, have only one-sided error, since, when there is only one contender, they assert solitude with probability one. It turns out that the different versions of error tolerance are closely related. We demonstrate the relationship for nondistributively terminating algorithms. A similar, and simpler, approach works for distributive termination.

Let $\alpha$ be a nondistributively terminating (nondeterministic) probabilistic algorithm for solitude detection, with two-sided error $(\epsilon, \delta)$. Let $\beta$ be a similar algorithm with one-sided error $\epsilon$. Consider the following algorithm $\gamma$:

1. All contenders flip a coin at random and send the result to the next contender.

2. Contenders which receive a different bit from that sent send an alarm. (Receipt of an alarm forces the decision "not alone".)

3. Contenders which receive the same bit as was sent initiate algorithm $\alpha$. (Note: we only need to worry about the case when all contenders initiate $\alpha$.)

4. If a contender enters, even tentatively, the state "not alone" then it sends a *sweepup* message after possibly sending its tentatively last message.

5. Sweepup messages get forwarded by non-contenders.

6. If a contender receives a sweepup message without having sent one, it sends an alarm.

7. If a contender receives a sweepup message without having received anything else since it sent one, it initiates algorithm $\beta$. (Again we need only worry about the case where all processors initiate algorithm $\beta$.)

8. Assuming no alarms, the result of algorithm $\beta$ is the result of the entire algorithm.

**Complexity.** First, consider the case of a single contender. Suppose that $\alpha$ costs $f$ expected bits and $\beta$ costs $g$ expected bits on a particular ring with one contender. Steps 1, 3 and (4,5) cost a total of $f + O(n)$ bits, since each processor sends at most one sweepup message. The lone contender will start algorithm $\beta$ only if the sweepup message travels all the way around the ring, without any more messages arriving at the contender. So $\beta$ is started only if $\alpha$ terminates with the contender erroneously concluding that it is not

alone, which happens with probability at most $\delta$. The expected cost of step 7 is thus $\delta g$.

Algorithm $\gamma$ has lower expected complexity than algorithm $\beta$ unless $f + \delta g + O(n) \geq g$. That is, $f \geq (1-\delta)g - O(n)$.

When there are two or more contenders, the expected complexity can be made $O(n)$ by interleaving coin tosses with the regular messages, which only doubles the cost when there is a single contender.

**Error analysis.** It is clear that a lone contender cannot err, since algorithm $\gamma$ only concludes "not alone" when algorithm $\beta$ does so. So consider the case of two or more contenders. With probability $\geq \frac{1}{2}$, some contender sends an alarm at step 2. So the probability that $\gamma$ proceeds to step 4 without any alarms is $\leq \frac{1}{2}$. With probability $\leq \epsilon$ algorithm $\alpha$ answers "alone", and algorithm $\gamma$ halts in error. With probability $\leq \epsilon$, algorithm $\alpha$ concludes "not alone", and $\beta$ answers "alone". Thus $\Pr(\text{error}) \leq \frac{1}{2}(\epsilon + \epsilon)$.

If follows from the complexity analysis that if we have a lower bound of $\Omega(f(n,\epsilon))$ on nondistributively terminating algorithms for solitude detection (or verification) with one-sided error $\epsilon$, then we have an $\Omega((1-\delta)(f(n,\epsilon))$ lower bound for nondistributively terminating algorithms for solitude detection (or verification) with two-sided error $(\epsilon,\delta)$. As remarked earlier, the same holds if nondistributive termination is replaced by distributive termination.

Note that the converse also holds. If we have an $O(f(n,\epsilon))$ bit algorithm $\alpha$ for solitude detection with one-sided error $\epsilon$ and with either type of termination, then we can construct an $O(n + (1-\delta)(f(n,\epsilon))$ bit algorithm for solitude detection with two-sided error $(\epsilon,\delta)$ and the corresponding type of termination. Each contender simply, with probability $\delta$, sends an alarm forcing the conclusion "not alone". Otherwise, it runs algorithm $\alpha$.

## 7 Conclusions

We have presented upper and lower bounds that match to within a constant factor for the bit complexity of solitude detection on various classes of rings. The type and complexity of the solution were found to depend not only upon the amount of error that could be tolerated, but also upon the amount of knowledge of the ring size which the algorithm could assume. Without any knowledge of ring size, only nondistributive termination is possible for solitude detection. When size is bounded, distributive termination is possible. And when ring size is known to within a constant factor, there is no additional cost for insisting on distributive termination over nondistributive termination.

This contrasts with the case when an algorithm need only work for one fixed ring size. This situation is explored in a companion paper [2] in which the following is proved. Let $\nu(n)$ be the smallest nondivisor or $n$. The inherent complexity of distributively terminating solutions for solitude detection with confidence $1-\epsilon$ is $\Theta(n \min(\log \nu(n) + \sqrt{\log\log(\frac{1}{\epsilon})}, \sqrt{\log n}, \log\log(\frac{1}{\epsilon})))$ expected bits when $n$ is known exactly. This complexity reduces to $\Theta(n \min(\log \nu(n) + \log\log\log(\frac{1}{\epsilon}), \log\log n, \log\log(\frac{1}{\epsilon})))$ expected bits for nondistributively terminating solitude detection with confidence $1-\epsilon$.

When no error can be tolerated, these results simplify to $\Theta(n\sqrt{\log n})$ bits for distributive termination and $\Theta(n \log\log n)$ bits for nondistributive termination with exact knowledge of ring size.

Solitude detection is related to some other well studied problems. As pointed out in the introduction, solitude detection reduces to leader election in $O(n)$ bits. For distributively terminating algorithms, the reductions are natural ones. Reductions for nondistributively terminating algorithms are more subtle and can be found in [2]. Algorithms for leader election can be constructed from the solitude detection algorithms described here together with an attrition algorithm. A discussion of the various resulting leader election algorithms appears in [3]. The same paper comments on the relation between solitude detection and the "$n$-finding" problem – the problem of determining the size of the ring.

## References


[1] K. Abrahamson, A. Adler, R. Gelbart, L. Higham, and D. Kirkpatrick. *The Bit Complexity of Randomized Leader Election on a Ring.* Technical Report 86-3, University of British Columbia, Vancouver B.C., 1986. submitted for publication.

[2] K. Abrahamson, A. Adler, L. Higham, and D. Kirkpatrick. *Probabilistic Solitude detection II: Rings Size Known Exactly.* Technical Report 86-26, University of British Columbia, 1986. submitted for publication.

[3] K. Abrahamson, A. Adler, L. Higham, and D. Kirkpatrick. Probabilistic solitude verification on a ring. In *Proc. 5th Annual ACM Symp. on Principles of Distributed Computing*, pages 161–173, 1986.

[4] D. Dolev, M. Klawe, and M. Rodeh. An $O(n \log n)$ unidirectional distributed algorithm for extrema finding on a circle. *J. Algorithms*, 3(3):245–260, 1982.

[5] A. Itai and M. Rodeh. Symmetry breaking in distributed networks. In *Proc. 22nd Annual Symp. on Foundations of Comput. Sci.*, pages 150–158, 1981.

[6] J. Pachl. *A Lower Bound for Probabilistic Distributed Algorithms.* Technical Report CS-85-25, University of Waterloo, Waterloo, Ontario, 1985.

[7] J. Pachl, E. Korach, and D. Rotem. Lower bounds for distributed maximum finding. *J. Assoc. Comput. Mach.*, 31(4):905–918, 1984.

[8] G. Peterson. An $O(n \log n)$ algorithm for the circular extrema problem. *ACM Trans. on Prog. Lang. and Systems*, 4(4):758–752, 1982.