Bounds on the Cleaning Times of Robot Vacuums

Zhentao Li* and Adrian Vetta†

Abstract. We show a robot vacuum using a protocol that next cleans the "dirtiest" incident edge may take exponential time to clean a network. This disproves a conjecture of Messinger and Nowakowski [2]. We also present two simple variants of this protocol that are polynomial time.

1 Introduction

Messinger and Nowakowski [2] examine the behaviour of the following deterministic walk W in an undirected graph $G = (V, E)$: When the walk reaches vertex v, the next edge traversed is the edge incident to v that was last traversed the furthest back in time. The motivation behind their work is a cleaning process where a robot has to clean the edges and/or vertices of a network. For example, the robot may clean a set of rooms, a set of algae infested pipelines, etc. A simple protocol for the robot would be to clean the most contaminated (that is, dirtiest) incident edge. Thus, we may view each edge as having a *weight* or *timestamp* w_e recording the time it was last cleaned (in either direction). The robot begins at a starting vertex s at time $T_0 = 1 + \max_{e \in E} w_e$. Then, at time T, the robot traverses the incident edge of minimum weight and then sets the weight of that edge to T.

The *cover time*, $c(G)$, of a connected graph G is the maximum number of steps, over all initial edge weightings w and all possible starting vertices s, until each edge has been visited. We define, C_m to be the worst-case cover time over all graphs containing exactly m edges.

In [2], it is shown that the *cover time* C_m is at most exponential in m, and the authors conjecture that it is, in fact, polynomial in m. Our main result, given in Section 2.1, is an example network whose cover time is exponential. Specifically, we present a graph on which the walk mimics a ternary counter. The counter is slightly strange and is not the standard ternary counter, but it still has the property that it takes exponential time before the walk activates the nth bit - this will give our result.

Finally, in Section 3, we present two simple modifications to this walk protocol that both lead to polynomial time cover times. The first is to make the protocol asymmetric: the walk choses the incident edge that was last traversed in the *same* direction the furthest back in time. The second modification is to incorporate a simple counter: the walk takes the incident edge that has been cleaned the fewest number of times.

*School of Computer Science, McGill University. Email: zhentao.li@mail.mcgill.ca

†Department of Mathematics and Statistics, and School of Computer Science, McGill University. Email: vetta@math.mcgill.ca

2 Exponential Bounds on the Cover Time

Here we present exponential bounds on the cover-time.

2.1 An Exponential Lower Bound

Theorem 1. *There exists a constant $\alpha > 0$ such that, for all m,*

$$C_m \geq \lceil \alpha(3/2)^{m/5} - 1/2 \rceil$$

Proof. Consider the graph G shown in Figure 1. We begin the walk at vertex s.

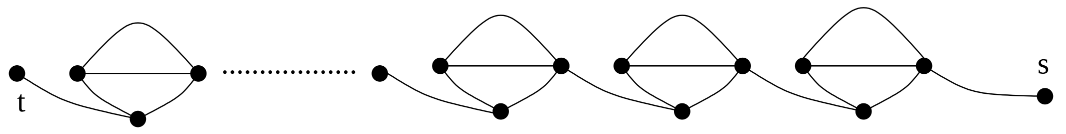

Figure 1: Graph G used to prove the exponential lower bound

There is a initial set of weights w that forces the deterministic walk to take an exponential amount of time to reach vertex t. Thus G consists of a sequence of copies of the subgraph H, shown in Figure 2. Clearly, each gadget shares one edge with the neighbouring gadget to its left and one edge with the neighbouring gadget to its right. These shared edges are dubbed *(left/right) exit edges* as when we traverse them we will exit the gadget and enter another gadget. Observe that the gadgets are ordered from right to left - each gadget will correspond to a bit in a counter, so as we move leftwards the gadgets correspond to more and more significant bits.

For each gadget, the weights on edges in the gadget are ordered relatively as in Figure 2. More specifically, the edge labelled 0 in the ith gadget to the right will have initial weight $5(i-1)+0$, the edge labelled 1 will have initial weight $5(i-1)+1$, etc.

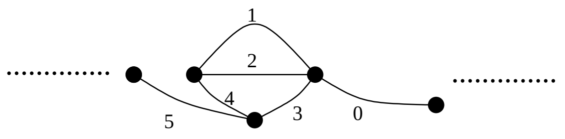

Figure 2: A single gadget H in the graph G

Observe then that the walk W on entering a gadget from right will traverse the edges of that gadget in the following cyclic order (where the label of an edge corresponds to the initial relative ordering of their weights):

$$0, 1, 2, 3, 4, 1, 0, 0, 2, 4, 5, 5, 3, 1, 2, 0, 0, 3, 4, 1, 2, 4, 5, 5, 3, 0$$

Note that for each subsequence $\{5,5\}$ the walk actually leaves the gadget on its left exit edge 5, then traverses through some gadgets on its left, before re-entering the gadget via the left exit edge 5. A similar observation applies to the subsequences $\{0,0\}$. Moreover, notice that in this cyclic order we exit the gadget three times via the right exit edge 0 but only twice via the left exit edge 5. In particular, the walk W starting at s exits each gadget in the cyclic order $RLRLR\cdots$, where R and L refer to right and left, respectively. Thus, if the robot leaves H from the left, then the next time it returns to H it exits the gadget from the right.

We will bound the cleaning time of the robot by number of times the robot visits the starting vertex s before the leftmost vertex is visited. We call a subwalk of the robot beginning at s and returning to s a *run* of the robot. Thus, any run has the property that the robot traverse leftwards through a number of copies of H and then immediately traverses rightwards and returns to the starting vertex.

Therefore, refering to the sequence $RLRLR\cdots$, whenever the robot is at the starting vertex, all copies of H are in one of three states. Specifically, we say that a gadget is in state $i \bmod 3$ if we have exited it from the right exactly i times. Consequently, we can then describe all the states in the graph by a number in base 3.

The total effect of a run on all the states can be thought of as incrementing the lowest digit of this number. However, instead of following the ordinary arithmetics of ternary numbers, we also increase the next digit (i.e., carry) when the current digit is increased from 1 to 2.

More formally, if x is a string of digits and S is the successor function then $S(x0) = x1$, $S(x1) = S(x)2$ and $S(x2) = S(x)0$. We write S^i to mean the function S composed i times. For example, the first few numbers are: 0, 1, 12, 120, 121, 1202, 1210, 1211, 12022, ...

Let B_k be the subsequence of this sequence consisting only of numbers with k digits. From the above example, we see that $|B_0| = 1, |B_1| = 1, |B_2| = 1, |B_3| = 2, |B_4| = 3$ (here 0 is viewed as having 0 digits). We will show that $|B_i| \geq \frac{3}{2}|B_{i-1}| - \frac{1}{2}$. To do so, we need the following lemma.

Lemma 2. *For all $i > 0$, $S^i(0) = S^j(0)c$ for some $j < i$ and $c \in \{0,1,2\}$.*

Proof. We prove this by induction. Clearly this is true for $i = 1$ as $S(0) = 1 = S^0(0)1$. Now suppose the lemma is true for $i-1$. Then $S^{i-1}(0) = S^j(0)c$ for some $c \in \{0,1,2\}$ and some $j < i-1$. We have three possibilities:
(i) If $c = 0$ then $S^i(0) = S^j(0)1$.
(ii) If $c = 1$ then $S^i(0) = S(S^j(0))2 = S^{j+1}(0)2$.
(iii) If $c = 2$ then $S^i(0) = S(S^j(0))0 = S^{j+1}(0)0$.
Since $j < i-1$, we have that $j+1 < i$. □

Observe that in the previous lemma, $S^j(0)$ is $S^i(0)$ with the last digit truncated. From the proof, we see that when we increase i by 1, j either increases by 0 or 1. If $S^i(0)$ is the first element of the sequence with d digits, then $S^i(0) = S^j(0)c$ where

$c \in \{0, 2\}$ and $S^j(0)$ is the first element of the sequence to have $d-1$ digits (otherwise $S^{i-1}(0)$ has d digits).

Therefore, we can conclude that if $b_1, \ldots, b_k$ are the elements of B_i then the elements of B_{i+1} are either $b_1 0, b_1 1, b_2 2, b_3 0, b_3 1, \ldots$ or $b_1 2, b_2 0, b_2 1, b_3 2, \ldots$

Therefore, $|B_{i+1}| \geq \lfloor \frac{3}{2}|B_i| \rfloor \geq \frac{3}{2}|B_i| - \frac{1}{2}$ as required. The theorem follows immediately, by repeatedly applying this inequality. □

2.2 An Exponential Upper Bound

For completeness, we present a simple proof of an exponential upper bound. We require this variant of Theorem 1 from [2].

Lemma 3. *In any connected graph G with at least i edges, any C_i consecutive steps of the robot visit at least i edges.*

Proof. Suppose not. Let H be the graph consisting of the edges visited during the the first $C_i - 1$ of these steps *plus* the two edges obtained by subdividing the edge visited in step C_i. Then $c(H) > C_i$ and $|E(H)| \leq i$, a contradiction. □

Theorem 4. $C_m \leq 1 + \max_{j \leq m} j(C_{m-j} + 1)$

Proof. Let G, w and s be chosen to attain the maximum cover time on connected graphs with m edges. Then, let F_{m-1} be the set of visited edges of G, starting at s, after C_{m-1} steps. By Lemma 3, $|F_{m-1}| \geq m-1$.

If $E(G) = F_{m-1}$ then we visited every edge of G in $C_{m-1} \leq 1 + \max_{j \leq m} j(C_{m-j} + 1)$ steps. So $E(G) - F_{m-1}$ must consist of a single edge $e = (u, v)$. First, assume that $s \neq u$ and $s \neq v$. Let E_e consist of all edges (including e itself) that share at least one endpoint with e. Let $i = |E_e|$. Note that whenever the robot exits $\{u, v\}$, it does so using a previously unvisited edge of E_e. We think of traversing an edge parallel to e as exiting $\{u, v\}$ and re-entering $\{u, v\}$ in the same step.

Observe, by Lemma 3, that every C_{m-i+1} steps, the robot is guaranteed to visit an edge of E_e. In particular, since $s \in V - \{u, v\}$, the first time the robot enters $\{u, v\}$, it does so via an unvisited edge of E_e, as all edges of E_e are unvisited at that point. This step is immediately followed by traversing another unvisited edge in E_e. Thus, in the first $C_{m-i+1} + 1$ steps, two edges of E_e are visited. Subsequently, all successive next visits to $\{u, v\}$ may occur via a visited edge, but they are certainly followed by traversing an unvisited edge. Hence edge e is traversed after at most $(i-1)(C_{m-i+1} + 1)$ steps. Letting $j = i - 1$, we bound C_m by the worst case $C_m \leq \max_{j \leq m} j(C_{m-j} + 1)$.

Finally, if $s = u$ or $s = v$ then we simply take one step and apply the previous argument. This gives the additional constant 1 in the bound. □

We now use the above theorem to show

Theorem 5. $C_m \leq 3^{m/3+1} - 3$.

Proof. We prove this by induction. The theorem is true for $m < 3$ since $C_0 = 0, C_1 = 1$ and $C_2 = 3$. So, consider $m \geq 3$. By Theorem 4, either $C_m \leq 1 + C_{m-1} + 1$ or $C_m \leq 1 + j(C_{m-j} + 1)$ for $j \geq 2$. In the first case,

$$\begin{aligned} C_m &\leq 1 + C_{m-1} + 1 \\ &\leq 3^{(m-1)/3+1} - 3 + 2 \\ &\leq 3^{m/3+1} - 3 \end{aligned}$$

Here, the last inequality follows from the fact that $(1-3^{-1/3})3^{m/3+1} \geq 9(1-3^{-1/3}) > 2$, for $m \geq 3$. In the second case, as $j \geq 2$, we have

$$\begin{aligned} C_m &\leq 1 + j(C_{m-j} + 1) \\ &= jC_{m-j} + 1 + j \\ &\leq j(3^{(m-j)/3+1} - 3) + 1 + j \\ &= j3^{(m-j)/3+1} + 1 - 2j \\ &\leq 3^{j/3}3^{(m-j)/3+1} + 1 - 2j \\ &\leq 3^{m/3} - 3 \end{aligned}$$

□

2.3 What if we buy Two Robots, or Three...?

Interactions between multiple cleaning robots cannot be exploited to overcome the inherent exponentiality in our bad example. To construct bad examples with multiple robots, we simply adapt the example of Section 2.1. (Here, we make the assumption that two robots cannot clean an edge simultaneously; this can be achieved, for example, by perturbing each robots clocks very slightly so that moves are not simultaneous.) With two robots, we can just take two copies of each edge in the example with appropriate weightings; more generally, replace each edge with as many copies as there are robots and give copies of the same edge consecutive weightings. Specifically, if there are k robots, we replace each edge e with weight w_e by k parallel edges $e_1, \ldots, e_k$ where e_i has weight $w_e + \varepsilon i$ where ε is chosen to be smaller than the smallest difference between two edge weights in the original graph divided by k. Thus, we obtain

Corollary 6. *With k robots, the worst-case cover time is at least $\alpha \left(\frac{3}{2}\right)^{\frac{m}{5k}} - \frac{1}{2}$ for some constant α.* □

3 Polynomial Time Protocols

In this section we present two simple deterministic walks that do lead to polynomial cover times.

3.1 An Asymmetric Walk

The asymmetic variant of the above walk is the following: at any vertex the walk choses the incident edge that was last traversed in the *outgoing* direction the furthest back in time. We will see that this walk always does produce a polynomial cover time. First, however, we remark that this protocol would correspond to using only one chip in the rotor router model of Propp [1]. To see this, observe that the walk can be viewed as taking the edge that has so far been cleaned the fewest number of times in the *outgoing* direction. Consequently, it can be implemented using a pointer (at each vertex) that rotates in a circular order through the incident edges at the vertex; at each step, the walk follows the edge indicated by the pointer and then the pointer shifts one place.

Let $\hat{C}(G)$ be the worst case cover time on a graph using this walk. Then $\hat{C}(G)$ has a polynomial upper bound.

Theorem 7. $\hat{C}(G) \leq 2mn$

Proof. Take the last edge $e = (u, v)$ to be traversed, say, in the direction from u to v. We build sets S_i, for $1 \leq i \leq n$ where each edge leaving S_i is traversed at most i times from S_i to $V - S_i$. This is true for $S_1 = \{u\}$. Now take S_i. We leave on any arc out of S_i at most i times. Then, as $|\delta^-(S_i)| = |\delta^+(S_i)|$ and the number of exits from S_i is at least the number of entries into S_i, there is some arc f in $\delta^-(S_i)$ that is traversed at most i times. Let $f = (x, y)$. Then no arc out of x is used more that $i + 1$ times. Hence we may set $S_{i+1} = S_i \cup \{x\}$. Consequently, no arc is used more than n times. The theorem follows. □

3.2 Counting Robots

For our second protocol modification, instead of traversing the least recently cleaned edge incident to the current vertex, the robot will keep count of the number of times an edge is cleaned and, at each step, it will traverse the adjacent edge which has been cleaned the least number of times. Let $C'(G)$ be the worst case cover time on a graph using this walk. Then $C'(G)$ also has a polynomial upper bound.

Theorem 8. $C'(G) \leq 2mn$

Lemma 9. *Let W be the walk performed by the robot before the last edge is visited. Then at that point, there is a spanning tree T of G with edges $e_0, \ldots, e_{n-1}$ such that e_i has weight at most $2i$.*

Proof. We recursively build this spanning tree. Let e_0 be the only unvisited edge. Given $e_0, \ldots, e_k$, $k < n - 1$, we find e_{k+1} as follows.

Let $F_k = \bigcup_{i=0}^{k} e_k$ and V_k be the union of the endpoints of edges in F_k. If there is no edge in W which is traversed from V_k to $V - V_k$ then we can let e_{k+1} be the only edge in W traversed from $V - V_k$ to V_k since e_{k+1} is traversed exactly once.

Otherwise, let $e = (u, v)$ be the last edge in W traversed from $u \in V_k$ to $v \in V - V_k$. At that point, e has weight at most $2k + 1$ (since it the robot did not traverse the edge in F_k which has an endpoint u). After that point, by definition of e, the robot only returns to V_k at most once and thus traverses e at most once more. Therefore e has weight at most $2k + 2$ and we can let $e_{k+1} = e$.

It is clear that in each case, $V_{k+1} - V_k$ contains a vertex. □

Lemma 10. *Let W be the walk performed by the robot before the last edge is visited. Every edge $e = (u, v)$ is traversed at most $2n$ times in W.*

Proof. Suppose, without loss of generality, that the last time e is traversed in W, it is traversed from u to v. Let T be the spanning tree in the previous lemma. Then u is incident to an edge of weight at most $2n - 2$. Thus, just before we traverse e for the last time it has weight at most $2n - 2$, so it is used at most $2n - 1$ times in total. □

Now Theorem 8 follows as a corollary by simply taking the sum over all edges. □

Acknowledgments. We thank the referee for many helpful suggestions and improvements.

References

[1] J. Cooper and J. Spencer, "Simulating a Random Walk with Constant Error", *Combinatorics, Probability and Computing*, **15(6)**, pp815-822, 2006.

[2] M. Messinger and R. Nowakowski, "The Robot Cleans Up", *Proceedings of the 2nd Annual International Conference on Combinatorial Optimization and Applications (COCOA)*, **LNCS 5161**, pp309-318, 2008.