The Complexity of Node Counting on Undirected Graphs

Boleslaw Szymanski
Department of Computer Science, Rensselaer Polytechnic Institute
szymansk@cs.rpi.edu

Sven Koenig
College of Computing, Georgia Institute of Technology
skoenig@cc.gatech.edu

Abstract

We analyze the complexity of Node Counting, a graph-traversal method. On many graphs arising in control problems in Artificial Intelligence, Node Counting performs as efficiently as other methods which are known to be of polynomial complexity in the number of states (e.g., Learning Real-Time A* method). We show that complexity of Node Counting on undirected graphs is $\Omega(n^{\sqrt{n}})$, which is not polynomial in the number of states. This solves an open problem from the literature.

1 Introduction

Node Counting is a simple graph-traversal method that has been used in artificial intelligence to explore unknown environments, either on its own or to accelerate reinforcement-learning methods. To the best of our knowledge, the term "Node Counting" was first used in [Thrun, 1992]. Later, it has been suggested that variants of Node Counting approximate the exploration behavior of ants, that use pheromone traces to guide their exploration [Wagner *et al.*, 1997]. Node Counting is also similar to "Avoiding the Past: A Simple but Effective Strategy for Reactive Navigation" [Balch and Arkin, 1993] with mobile robots.

Node Counting has been often compared to Learning Real-Time A* (LRTA*) [Korf, 1990] which is probably the best known real-time heuristic search method. It always moves to a successor state with a minimal approximation of the goal distance of the current state. Control methods similar to LRTA* include RTA* [Korf, 1990] and methods studied in [Russell

Initially, the u-values $u(s)$ are zero for all $s \in S$.

1. s := s_{start}.
2. If $s \in G$, then stop successfully.
3. a := one-of $\arg\min_{a \in A(s)} u(succ(s, a))$.
4. $u(s)$:= $1 + u(s)$.
5. Execute action a. This changes the current state to $succ(s, a)$.
6. s := the current state.
7. Go to 2.

Figure 1: Node Counting

and Wefald, 1991].

Experimental results indicated that Node Counting and LRTA* perform about equally well in many domains from artificial intelligence, which are typically undirected. LRTA* was know to have polynomial complexity in the number of states, thus, it was expected that Node Counting have also a small complexity on undirected graphs [Koenig and Simmons, 1996]. In this paper, we present an undirected tree on which, perhaps surprisingly, complexity of Node Counting is $\Omega(n^{\sqrt{n(1/6-\epsilon)}})$, where $0 < \epsilon < \frac{1}{6}$ is an arbitrarily small constant. Hence, its complexity is not polynomial in the number of states. We also show how the tree can be extended to show that complexity of Node Counting on undirected graphs is $\Omega(n^{\sqrt{n}})$.

We use the following notation to describe Node Counting: S denotes the finite set of states of the domain, $s_{start} \in S$ the start state, and $\emptyset \neq G \subseteq S$ the set of goal states. The number of states is $n := |S|$. $A(s) \neq \emptyset$ is the finite set of actions that can be executed in state $s \in S$. $succ(s, a)$ denotes the successor state that results from the execution of action $a \in A(s)$ in state $s \in S$. We measure the complexity of Node Counting in action executions and assume that one can reach a goal state from every state that can be reached from the start state. Domains with this property guarantee that Node Counting reaches a goal state eventually. We also use two operators with the following semantics: Given a set X, the expression "one-of X" returns an element of X according to an arbitrary rule. A subsequent invocation of "one-of X" can return the same or a different element. The expression "$\arg\min_{x \in X} f(x)$" returns the elements $x \in X$ that minimize $f(x)$, that is, the set $\{x \in X | f(x) = \min_{x' \in X} f(x')\}$.

Node Counting is shown in Figure 1. A u-value $u(s)$ corresponds to the number of times Node Counting has already been in state s. Node Counting always moves to a successor state with a minimal u-value because it wants to get to states which it has visited a smaller number of times to eventually reach a state that it has not yet visited at all, that is, a potential goal state. The program for LRTA* is nearly identical, the only difference is in line 4 which for LTRA* assigns $1 + u(succ(s, a))$ to variable $u(s)$, whereas Node Counting sets this variable to $1 + u(s)$.

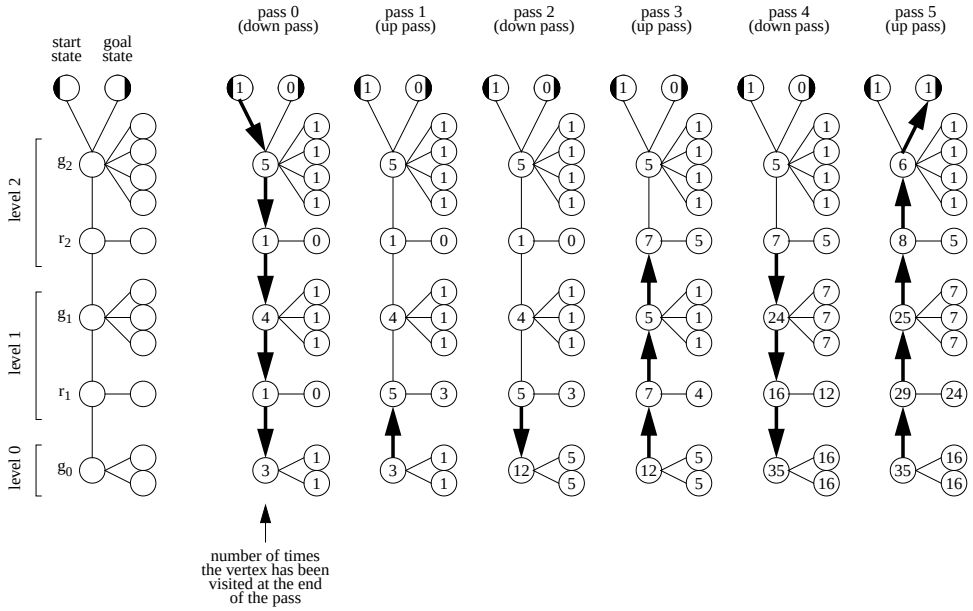

Figure 2: Node Counting has Exponential Runtime ($m = 2, n = 18$)

2 Complexity of Node Counting

In the following, we present an undirected tree that shows that the complexity of Node Counting on undirected graphs is $\Omega(n^{\sqrt{n(1/6-\epsilon)}})$, where $0 < \epsilon < \frac{1}{6}$ is an arbitrarily small constant.

Consider undirected trees of the kind shown in Figure 2. The trees have $m + 1 \geq 3$ levels. The levels consist of nodes of three different kinds: g-subroots, r-subroots, and leaves that are connected to the subroots. g-subroots and r-subroots alternate. At level $i = 0$, there is one subroot, namely a g-subroot g_0. At levels $i = 1 \ldots m$, there are two subroots, namely an r-subroot r_i and a g-subroot g_i. Subroot g_i has $m + i$ leaves connected to it, and subroot r_i has one leaf connected to it. Finally, subroot g_m is connected to two additional nodes, namely the start node and the single goal node. The trees have $n = \frac{3}{2}m^2 + \frac{9}{2}m + 3$ nodes.

Node Counting proceeds in a series of passes through the tree, each pass traversing the subroots in the opposite order than the previous pass. We call a pass that traverses the subroots in descending order a down pass, and a pass that traverses them in ascending order an up pass. We number passes from zero on upward. Thus, even passes are down passes and odd passes are up passes. A pass ends immediately before it changes directions.

The semantics of "one-of arg" operator in Line 3 in Figure 2 allows for a selection of any element of the operator's argument set. Hence, we present a selection rule for this operator that results in bad performance. The selection is possible only when the current state of Node Counting is a subroot since all leaves have only a single successor state. We make the selections as follows:

During pass zero, whenever possible, a leaf node is selected at g-subroots and a g-subroot is selected at r-subroots. The selection of the subroot is unique. In case of leaves, any eligible

leaf can be chosen. Pass zero ends in subroot g_0 after each leaf of subroot g_0 has been visited once. As a result, at the end of pass zero each subroot g_i has been visited $m+i+1$ times, and each of its leaves has been visited once. Each r-subroot has been visited once, and its leaf has not been visited at all.

During all subsequent passes, whenever possible, a subroot is selected. If two r-subroots are eligible for selection (when Node Counting is at a g-subroot) then that r-subroot is chosen which extends the current pass. If two g-subroots are eligible (when Node Counting is at an r-subroot), the g-subroot is chosen in such a way as to terminate the current pass and start a new one in the opposite direction. When only leaves are eligible for selection, one of them is chosen arbitrarily.

In the sequel, we consider a tree with arbitrary but constant number of levels $m+1$. Hence, in the three functions defined below, an argument m is omitted for the sake of brevity. Let $v_p(s)$ denote the total number of times that subroot s of the considered tree has been entered at the end of pass p. By definition, $v_p(s)$ is a nondecreasing function of p. Our selection of the successor rules ensure that all leaves of a subroot have been entered the same number of times at the end of each pass, so we will denote $w_p(s)$ the number of times each of the leaves of subroot s has been entered at the end of pass p. Finally, let $x_p(s)$ denote the total number of times subroot s has been entered from non-leaves at the end of pass p. These values relate as follows: The total number of times that a subroot was entered at the end of pass p is equal to the product of the number of its leaves and the total number of times that it was entered from each of its leaves at the end of pass p (which equals the total number of times that each of its leaves was entered at the end of pass p) plus the total number of times the subroot was entered from non-leaves at the end of pass p. For example, $v_p(g_i) = (m+i)w_p(g_i)+x_p(g_i)$.

Lemma 1 *Assume that Node Counting visits subroot s (with $s \neq g_m$) during pass p, where $0 < p < 2m$. The values $v_p(s)$ can then be calculated as follows:*

$$
\begin{aligned}
v_{2k+1}(g_0) = v_{2k}(g_0) &= m\, v_{2k}(r_1) + x_{2k}(g_0) \\
\text{and for } i > 0&\text{:} \\
v_{2k}(g_i) &= (m+i)\min(v_{2k-1}(r_i), v_{2k}(r_{i+1})) + x_{2k}(g_i) \\
v_{2k+1}(g_i) &= (m+i)\min(v_{2k}(r_{i+1}), v_{2k+1}(r_i)) + x_{2k+1}(g_i) \\
v_{2k}(r_i) &= \min(v_{2k-1}(g_{i-1}), v_{2k}(g_i)) + x_{2k}(r_i) \\
v_{2k+1}(r_i) &= \min(v_{2k}(g_i), v_{2k+1}(g_{i-1})) + x_{2k+1}(r_i)
\end{aligned}
$$

Proof: Consider first the subroot g_0 when visited by down pass $2k$. While the number of visits at any of its leaves is less than $v_{2k}(r_1)$, Node Counting will move to such a leaf. Thus, when Node Counting moves to r_1, it must be that $w_{2k+1}(g_0) = w_{2k}(g_0) = v_{2k}(r_1)$. Hence $v_{2k+1}(g_0) = v_{2k}(g_0) = m\, v_{2k}(r_1) + x_{2k}(g_0)$.

Assume now that Node Counting visits subroot g_i ($0 < i < m$) during down pass $2k \neq 0$. As long as the number of visits to any of the leaves of subroot g_i is less than $\min(v_{2k-1}(r_i), v_{2k}(r_{i+1}))$, Node Counting will move to such a leaf. Hence, when Node Counting moves to another subroot, it must hold that $w_{2k}(g_i) = \max(\min(v_{2k-1}(r_i), v_{2k}(r_{i+1})), w_{2k-1}(g_i))$. Likewise, assume that Node Counting visits subroot g_i (with $0 < i < m$) during up pass $2k+1 > 0$. Then, according to our selection of the successor rules, it holds that $w_{2k+1}(g_i) = \max(\min(v_{2k}(r_{i+1}), v_{2k+1}(r_i)), w_{2k}(g_i))$.

We now show by induction on p that, for all passes p and all subroots g_i with $1 \leq i < m$, it holds that $w_p(g_i) \leq \min(v_p(r_i), v_p(r_{i+1}))$. For pass zero, it holds that $w_0(g_i) = 1 \leq \min(1,1) = \min(v_p(r_i), v_p(r_{i+1}))$. Assume that the inequality holds for down pass $2k$. Then, if Node Counting visits subroot g_i during up pass $2k+1$, it holds that

$$\begin{aligned} w_{2k+1}(g_i) &= \max(\min(v_{2k}(r_{i+1}), v_{2k+1}(r_i)), w_{2k}(g_i)) \\ &\leq \max(\min(v_{2k+1}(r_i), v_{2k+1}(r_{i+1})), \min(v_{2k}(r_i), v_{2k}(r_{i+1}))) \\ &= \min(v_{2k+1}(r_i), v_{2k+1}(r_{i+1})), \end{aligned}$$

because $v_p(s)$ is a non-decreasing function of pass number p. If Node Counting does not visit subroot g_i during up pass $2k+1$, then it holds that $w_{2k+1}(g_i) = w_{2k}(g_i) \leq \min(v_{2k}(r_i), v_{2k}(r_{i+1})) \leq \min(v_{2k+1}(r_i), v_{2k+1}(r_{i+1}))$, again based on monotonicity of $v_p(s)$ in p. Assuming that the inequality holds for up pass $2k+1$, a similar argument shows that the inequality continues to hold for the following down pass.

Now assume again that Node Counting visits subroot g_i ($0 < i < m$) during down pass $2k \neq 0$. Then, as we just showed $w_{2k-1}(g_i) \leq \min((v_{2k-1}(r_i), v_{2k-1}(r_{i+1})) \leq \min(v_{2k-1}(r_i), v_{2k}(r_{i+1}))$ because $v_p(s)$ is nondecreasing function of p. Hence, it holds that

$$\begin{aligned} w_{2k}(g_i) &= \max(\min(v_{2k-1}(r_i), v_{2k}(r_{i+1})), w_{2k-1}(g_i)) \\ &= \min(v_{2k-1}(r_i), v_{2k}(r_{i+1})) \end{aligned}$$

and finally

$$\begin{aligned} v_{2k}(g_i) &= (m+i) w_{2k}(g_i) + x_{2k}(g_i) \\ &= (m+i) \min(v_{2k-1}(r_i), v_{2k}(r_{i+1})) + x_{2k}(g_i). \end{aligned}$$

A similar argument shows that it holds that $v_{2k+1}(g_i) = (m+i) w_{2k+1}(g_i) + x_{2k+1}(g_i) = (m+i) \min(v_{2k}(r_{i+1}), v_{2k+1}(r_i)) + x_{2k+1}(g_i)$.

The remaining two equalities defining the number of visits at r-subroots can be derived similarly. ■

We now use the lemma to prove the following theorem.

Theorem 1 *If $p = 2k$ for $0 \leq k \leq m$, then the down pass ends at subroot g_0 and it holds that*

$$v_{2k}(g_i) = \begin{cases} m(v_{2k-2}(g_i) + 2k) + k + 1 & \text{for } i = 0 < k \\ (m+i)(v_{2k-2}(g_i) + 2k - 2i + 1) + 2k - 2i + 1 & \text{for } 0 < i < k \\ m + i + 1 & \text{otherwise} \end{cases}$$

$$v_{2k}(r_i) = \begin{cases} v_{2k-1}(g_{i-1}) + 2k - 2i + 2 & \text{for } 0 < i \leq k \\ 1 & \text{otherwise} \end{cases}$$

$$x_{2k}(g_i) = \begin{cases} k + 1 & \text{for } i = 0 \\ 2k - 2i + 1 & \text{for } 0 < i < k \\ 1 & \text{otherwise} \end{cases}$$

$$x_{2k}(r_i) = \begin{cases} 2k - 2i + 2 & \text{for } 0 < i \leq k \\ 1 & \text{otherwise} \end{cases}$$

$$\begin{aligned} v_{2k}(r_i) &\geq v_{2k-1}(r_{i-1}) && \text{for } 1 < i \leq k \\ v_{2k}(g_i) &> v_{2k-1}(g_{i-1}) && \text{for } 0 < i < k \end{aligned}$$

If $p = 2k + 1$ for $0 \leq k \leq m$, then the up pass ends at subroot r_{k+1} (with the exception of up pass $2m + 1$ that ends at the goal state) and it holds that

$$
\begin{array}{lcll}
v_{2k+1}(g_i) & = & \left\{ \begin{array}{ll} v_{2k}(g_i) & \text{for } i = 0 \\ (m+i)(v_{2k-1}(g_i) + 2k - 2i) + 2k - 2i + 2 & \text{for } 0 < i < k \\ m + i + 2 & \text{for } 0 < i = k \\ m + i + 1 & \text{otherwise} \end{array} \right. \\
v_{2k+1}(r_i) & = & \left\{ \begin{array}{ll} v_{2k}(g_i) + 2k - 2i + 3 & \text{for } 0 < i \leq k \\ v_{2k+1}(g_{i-1}) + 2 & \text{for } i = k + 1 \leq m \\ 1 & \text{otherwise} \end{array} \right. \\
\\
x_{2k+1}(g_i) & = & \left\{ \begin{array}{ll} k + 1 & \text{for } i = 0 \\ 2k - 2i + 2 & \text{for } 0 < i \leq k \\ 1 & \text{otherwise} \end{array} \right. \\
x_{2k+1}(r_i) & = & \left\{ \begin{array}{ll} 2k - 2i + 3 & \text{for } 0 < i \leq k \\ 2 & \text{for } i = k + 1 \leq m \\ 1 & \text{otherwise} \end{array} \right. \\
\\
v_{2k+1}(r_i) & \geq & v_{2k}(r_{i+1}) & \text{for } 0 < i \leq k \\
v_{2k+1}(g_i) & > & v_{2k}(g_{i+1}) & \text{for } 0 \leq i < k
\end{array}
$$

Proof by induction on the number of executed actions:

Part 1: The values are correct for $p = 0$.

$$
\begin{array}{lcll}
v_0(g_i) & = & m + i + 1 & \text{for } 0 \leq i \leq m \\
v_0(r_i) & = & 1 & \text{for } 0 < i \leq m \\
\\
x_0(g_i) & = & 1 & \text{for } 0 \leq i \leq m \\
x_0(r_i) & = & 1 & \text{for } 0 < i \leq m
\end{array}
$$

At the end of the down pass, Node Counting is at subroot g_0 and is about to move to subroot r_1, starting an up pass.

For the remainder of the proof, notice that Node Counting cannot return to a subroot during a pass after it has moved from the subroot to a different subroot (such a return constitutes a change of direction of a pass, so it ends the current pass and starts a new one).

Part 2: Assume that $p = 2k + 1$ for $0 \leq k \leq m$. Up pass $2k + 1$ starts where the previous down pass ended, that is, at subroot g_0. To determine the values $v_{2k+1}(s)$ for subroot s we distinguish the following six cases:

1: $s = g_0$ for $0 \leq k \leq m$.

> Value $x_{2k+1}(g_0)$: According to the induction hypothesis, it holds that $x_{2k+1}(g_0) = x_{2k}(g_0) = k + 1$.
>
> Value $v_{2k+1}(g_0)$: According to the induction hypothesis, Node Counting starts the down pass at subroot g_0 and the next subroot that Node Counting visits is r_1. Thus, it holds that $v_{2k+1}(g_0) = v_{2k}(g_0)$.

Inequality $v_{2k+1}(g_0) > v_{2k}(g_1)$ for $k > 0$: According to the induction hypothesis, it holds for $0 \leq l \leq k$ that

$$v_{2l+1}(g_0) = v_{2l}(g_0) \quad = \quad \begin{cases} m+1 & \text{for } l = 0 \\ m(v_{2l-2}(g_0) + 2l) + l + 1 & \text{otherwise.} \end{cases}$$

From this definition, it follows that $v_{2l+1}(g_0) > mv_{2l-1}(g_0)$ and by induction on l we get $v_{2l+1}(g_0) > v_1(g_0)m^l = (m+1)m^l$. Solving the recursion yields

$$v_{2l+1}(g_0) \quad = \quad \frac{m^{l+3} + m^{l+2} + m^{l+1} - (2l+2)m^2 + (l-2)m + l + 1}{m^2 - 2m + 1}.$$

Similarly, according to the induction hypothesis, it holds for $1 \leq l \leq k$ that

$$v_{2l}(g_1) \quad = \quad \begin{cases} m+2 & \text{for } l = 1 \\ (m+1)(v_{2l-2}(g_1) + 2l - 1) + 2l - 1 & \text{otherwise.} \end{cases}$$

Solving the recursion yields

$$v_{2l}(g_1) \quad = \quad \frac{(m+1)^{l-1}(m^3 + 5m^2 + 8m + 4) - (m^2 + 4 + 2m^2 l + 4ml + 4m)}{m^2}.$$

Using the previous results, we verify for $2 \leq m < 5$ and $0 < k \leq m$ that $v_{2k+1}(g_0) > v_{2k}(g_1)$ (see Table below).

m	$v_3(g_0)$	$v_2(g_1)$	$v_4 5g_0)$	$v_4(g_1)$	$v_7(g_0)$	$v_6(g_1)$	$v_9(g_0)$	$v_8(g_1)$
2	12	4	35	24				
3	20	5	75	35	247	165		
4	30	6	139	48	584	270	2373	1392

Now assume that $m \geq 5$. Then, using the previous results, we know that $v_{2k+1}(g_0) > (m+1)m^k$ and $v_{2k}(g_1) < (m+1)^{k-1}(m^3+5m^2+8m+4)/m^2 = (m+1)^k(1+4(m+1))/m^2$ for $0 < k \leq m$. We also utilize the well known inequality

$$\left(1 + \frac{1}{m}\right)^m < e, \tag{1}$$

where e is the basis of natural algorithms (this inequality holds for all natural m, see [Finney and Thomas, 1994]). Then, it holds for $0 < k \leq m$ that

$$\begin{aligned} v_{2k}(g_1) &< (m+1)^k \frac{1 + 4(m+1)}{m^2} \\ &= m^k(1 + 1/m)^k(m+1)\left(\frac{1}{m+1} + \frac{4}{m^2}\right) \\ &\leq m^k(m+1)(1 + 1/m)^m\left(\frac{1}{m+1} + \frac{4}{m^2}\right) \\ &< m^k(m+1)e\left(\frac{1}{6} + \frac{4}{25}\right) < m^k(m+1) \\ &< v_{2k+1}(g_0). \end{aligned}$$

2: $s = r_i$ for $0 < i \leq k \leq m$.

Value $x_{2k+1}(r_i)$: According to the induction hypothesis, it holds that $x_{2k+1}(r_i) = x_{2k}(r_i) + 1 = 2k - 2i + 3$.

Value $v_{2k+1}(r_i)$: According to the induction hypothesis, it holds that $v_{2k+1}(g_{i-1}) > v_{2k}(g_i)$ for $0 < i \leq k$. Thus, the next subroot that Node Counting visits is g_i. According to the lemma and the induction hypothesis, it holds that $v_{2k+1}(r_i) = \min(v_{2k}(g_i), v_{2k+1}(g_{i-1})) + x_{2k+1}(r_i) = v_{2k}(g_i) + 2k - 2i + 3$.

Inequality $v_{2k+1}(r_i) \geq v_{2k}(r_{i+1})$: For $i = k$, according to the induction hypothesis, it holds that $v_{2k+1}(r_i) = v_{2k}(g_i) + 2k - 2i + 3 \geq 1 \geq v_{2k}(r_{i+1})$ because either no pass (for $i = k = m$) or only pass 0 ($i = k < m$) reached subroot r_{i+1}. Otherwise, if $0 < i < k$ then it holds from the induction hypothesis that $v_{2k+1}(r_i) = v_{2k}(g_i) + 2k - 2i + 3 \geq v_{2k-1}(g_i) + 2k - 2i = v_{2k}(r_{i+1})$ because $v_p(s)$ is a non-decreasing function of p.

3: $s = g_i$ for $0 < i < k$

Value $x_{2k+1}(g_i)$: According to the induction hypothesis, it holds that $x_{2k+1}(g_i) = x_{2k}(g_i) + 1 = 2k - 2i + 3$.

Value $v_{2k+1}(g_i)$: The induction hypothesis implies that $v_{2k}(r_{i+1}) \leq v_{2k+1}(r_i)$ so the next visited subroot is r_{i+1} and the lemma implies that $v_{2k+1}(g_i) = (m+i)v_{2k}(r_{i+1}) + x_{2k+1}(g_i) = (m+i)(v_{2k-1}(g_{i-1}) + 2k - 2i) + 2k - 2i + 2$.

Inequality $v_{2k+1}(g_i) > v_{2k}(g_{i+1})$: We start by noticing that $v_{2k+1}(g_i) = v_{2k}(g_i) + 1$ for $0 < i \leq k$. This is true for $i = k$ directly from equations defining the initial number of visits and the first changed values of $v_{2k+1}(g_k)$. For $i < k$ we have by induction from i to $i - 1$:

$$\begin{aligned} v_{2k+1}(g_i) &= (m+i)(v_{2k-1}(g_i) + 2(k-i)) + 2(k-i) + 2 \\ &= (m+i)(v_{2(k-1)}(g_i) + 2(k-i) + 1) + 2(k-i) + 2 \\ &= v_{2k}(g_i) + 1. \end{aligned}$$

As a result, the postulated inequality holds if, and only if

$$v_{2k}(g_i) \geq v_{2k}(g_{i+1}). \tag{2}$$

According to the induction hypothesis, it holds for $0 \leq l \leq k$ that

$$v_{2l}(g_i) = \begin{cases} m+i+1 & \text{for } l = i \\ (m+i)(v_{2l-2}(g_i) + 2l - 2i + 1) + 2l - 2i + 1 & \text{otherwise.} \end{cases}$$

From this definition, it follows that $v_{2l}(g_i) > (m+i)v_{2l-2}(g_i)$ and by induction on l we get $v_{2l}(g_i) > v_0(g_i)(m+i)^{l-1} = (m+i+1)(m+i)^{l-i}$. Solving the recursion yields

$$\begin{aligned} v_{2k}(g_i) &= (m+i+1) \\ &\quad \left((m+i)^{k-i} + \left((m+i)^{k-i} - 1\right) \frac{3(m+i)-1}{(m+i-1)^2} - (k-i)\frac{2}{m+i-1} \right) \\ &< (m+i+1)(m+i)^{k-i} \left(1 + \frac{3}{m+i-1} + \frac{2}{(m+i-1)^2} \right). \end{aligned}$$

In the proof of this inequality we will use again inequality (1), so for $m \geq 5,\ i > 0$ we obtain

$$\begin{aligned} v_{2k}(g_{i+1}) &< (m+i+2)(m+i+1)^{k-i-1}\left(1+\frac{3}{m+i}+\frac{2}{(m+i)^2}\right) \\ &\leq (m+i+1)^{k-i}\left(1+\frac{1}{m+i+2}\right)\left(1+\frac{1}{2}+\frac{1}{18}\right) \\ &< (m+i)^{k-i}e\frac{14}{9} \leq (m+i+1)(m+i)^{k-1}\frac{2e}{9} \\ &< (m+i+1)(m+i)^{k-i} < v_{2k}(g_i). \end{aligned}$$

proving the postulated inequality for $m \geq 5$. In the table below, we verify that $v_{2k}(g_i) \geq v_{2k}(g_{i+1}$ for $2 \leq m < 5$ using the solution of the recursion.

m	k	$v_{2k}(g_1)$	$v_{2k}(g_2)$	$v_{2k}(g_3)$	$v_{2k}(g_4)$
2	2	42	5		
3	2	35	6		
3	3	165	48	7	
4	2	48	7		
4	3	270	63	8	
4	4	1392	413	80	9

4: $s = g_k$ for $0 < k \leq m$

Value $x_{2k+1}(g_k)$: According to the induction hypothesis, it holds that $x_{2k+1}(g_k) = x_{2k}(g_k) + 1 = 2$.

Value $v_{2k+1}(g_k)$: Consider two cases. For $0 < k < m$, according to the induction hypothesis, it holds that $v_{2k+1}(r_k) \geq v_{2k}(r_{k+1})$ for $0 < k < m$. Thus, the next subroot that Node Counting visits is r_{k+1}. According to the lemma and the induction hypothesis, it holds that $v_{2k+1}(g_k) = (m+k)\min(v_{2k}(r_{k+1}), v_{2k+1}(r_k)) + x_{2k+1}(g_k) = m+k+2$. The complementary case is for $k = m$, when, according the induction hypothesis, it also holds that $v_{2k+1}(r_k) = v_{2m+1}(r_m) = v_{2m}(g_m) + 3 \geq 1$. According to the selection of the successor rule, Node Counting moves to the goal state and terminates since the value of the start state is still one and the value of the goal state is still zero. Thus, the next subroot that Node Counting visits is r_{k+1}. According to the lemma and the induction hypothesis, it holds that $v_{2k+1}(g_k) = v_{2m+1}(g_m) = 2m+2$.

5: $s = r_{k+1}$ for $0 \leq k < m$

Value $x_{2k+1}(r_{k+1})$: According to the induction hypothesis, it holds that $x_{2k+1}(r_{k+1}) = x_{2k}(r_{k+1}) + 1 = 2$.

Value $v_{2k+1}(r_{k+1})$: It follows from the induction hypothesis, that $v_{2k}(g_{k+1}) = m+k+2 \geq v_{2k+1}(g_k)$ because $v_{2k+1}(g_k) = m+k+2$ for $0 < k < m$ and $v_{2k+1}(g_0) = m+1$ for $k = 0$. According to the selection of the successor rules, this ends the up pass and starts a down pass. Thus, the next subroot that Node Counting visits is g_k. According to the lemma and the induction hypothesis, it holds that $v_{2k+1}(r_{k+1}) = \min(v_{2k}(g_{k+1}), v_{2k+1}(g_k)) + x_{2k+1}(r_{k+1}) = v_{2k+1}(g_k) + 2$.

6: $s = r_i$ for $1 \leq k+1 < i \leq m$ or $s = g_i$ for $0 \leq k < i \leq m$.

Values: Since up pass $2k+1$ starts at subroot r_0 and ends at subroot r_{k+1} (with the exception of up pass $2m+1$ that ends at the goal state), Node Counting does not visit the subroots r_i for $i > k+1$ nor the subroots g_i for $i > k$ during up pass $2k+1$. Thus, according to the induction hypothesis, it holds that $x_{2k+1}(r_i) = x_{2k}(r_i) = 1$ and $v_{2k+1}(r_i) = v_{2k}(r_i) = 1$ for $i > k+1$, and $x_{2k+1}(g_i) = x_{2k}(g_i) = 1$ and $v_{2k+1}(g_i) = v_{2k}(g_i) = m+i+1$ for $i > k$.

Part 3: Assume that $p = 2k$ for $0 < k \leq m$. Down pass $2k$ starts where the previous up pass ended, that is, at subroot r_k. To determine the values $v_{2k}(s)$ for subroot s, we distinguish five cases:

1: $s = r_k$ for $0 < k \leq m$.

Value $x_{2k}(r_k)$: According to the induction hypothesis, all previous passes, except pass 0, ended before reaching r_k, so $x_{2k}(r_k) = x_{2k-1}(r_k) = 2$.

Value $v_{2k}(r_k)$: According to the induction hypothesis, Node Counting starts the down pass at subroot r_k and the next subroot that Node Counting visits is g_{k-1}. Thus, it holds that $v_{2k}(r_k) = v_{2k-1}(r_k)$.

Inequality $v_{2k}(r_k) \geq v_{2k-1}(r_{k-1}$ for $k \geq i > 1$: As show above, $v_{2k}(r_k) = v_{2k-1}(r_k)$ and, according to the induction hypothesis, $v_{2k-1}(r_k) = v_{2k-1}(g_{k-1}) + 2 = m+k+3 = v_{2k-2}(g_{k-1}) + 3 = v_{2k-1}(r_{k-1})$.

2: $s = g_i$ for $0 < i < k \leq m$.

Value $x_{2k}(g_i)$: According to the induction hypothesis, it holds that $x_{2k}(g_i) = x_{2k-1}(g_i) + 1 = 2k - 2i + 1$.

Value $v_{2k}(g_i)$: According to the induction hypothesis, it holds that $v_{2k}(r_{i+1}) \geq v_{2k-1}(r_i)$. Thus, according to the selection of the successor rule, Node Counting continues with the down pass and the next subroot that it visits is r_i. Thus, according to the lemma and the induction hypothesis, it holds that $v_{2k}(g_i) = (m+i)\min(v_{2k-1}(r_i), v_{2k}(r_{i+1})) + x_{2k}(g_i) = (m+i)v_{2k-1}(r_i) + 2k - 2i + 1 = (m+i)(v_{2k-2}(g_i) + 2k - 2i + 1) + 2k - 2i + 1$.

Inequality $v_{2k}(g_i) > v_{2k-1}(g_{i-1})$: As shown above $v_{2k}(g_i) = (m+i)(v_{2k-2}(g_i) + 2k - 2i + 1) + 2k - 2i + 1$. According to the induction hypothesis $1 < v_{2k-2}(g_i) + 2k - 2i + 1 = v_{2k-1}(r_i)$, so $v_{2k}(g_i) = (m+i-1)v_{2k-1}(r_i) + 2k - 2i + 2 - (1 - v_{2k-1}(r_i)) > (m+i-1)v_{2k-1}(r_i) + 2k - 2i + 2$. On the other hand, from lemma we have $v_{2k-1}(g_{i-1}) = (m+i-1)v_{2k-2}(r_i) + x_{2k-2}(g_{i-1})$ either directly (for $i = 0$) or on the basis of inequality $v_{2k-2}(r_i) \leq v_{2k-3}(r_{i-1})$ satisfied according to the induction hypothesis for $k > i > 1$. However, $v_{2k-2}(r_i) \leq v_{2k-1}(r_i)$ based on monotonicity of $v_p(s)$ in p and $x_{2k-1}(g_{i-1} \leq 2k - 2i + 2$ from the induction hypothesis, so finally $v_{2k-1}(g_{i-1}) \leq (m+i-1)v_{2k-1}(r_i) + 2k - 2i + 2 < v_k(g_i)$.

3: $s = r_i$ for $0 < i < k \leq m$.

Value $x_{2k}(r_i)$: According to the induction hypothesis, it holds that $x_{2k}(r_i) = x_{2k-1}(r_i) + 1 = 2k - 2i + 2$.

Value $v_{2k}(r_i)$: According to the induction hypothesis, it holds that $v_{2k}(g_i) > v_{2k-1}(g_{i-1})$. Thus, the next subroot that Node Counting visits is g_{i-1}. According to the lemma and the induction hypothesis, it holds that $v_{2k}(r_i) = \min(v_{2k-1}(g_{i-1}), v_{2k}(r_i)) + x_{2k}(r_i) = v_{2k-1}(g_{i-1}) + 2k - 2i + 2$.

Inequality $v_{2k}(r_i) \geq v_{2k-1}(r_{i-1})$ for $1 < i \leq k$: According to the induction hypothesis, Node Counting visits subroot g_{i-1} during up pass $2k-1$. Thus, it holds that $v_{2k-1}(g_{i-1}) > v_{2k-2}(g_{i-1})$ and, according to the induction hypothesis, $v_{2k}(r_i) = v_{2k-1}(g_{i-1}) + 2k - 2i + 2 \geq v_{2k-2}(g_{i-1}) + 2k - 2i + 3 = v_{2k-1}(r_{i-1})$.

4: $s = g_0$ for $0 < k \leq m$.

Value $x_{2k}(g_0)$: According to the induction hypothesis, it holds that $x_{2k}(g_0) = x_{2k-1}(g_0) + 1 = k + 1$.

Value $v_{2k}(g_0)$: The next subroot that Node Counting visits is r_1, which ends the down pass and starts an up pass. For $k > 1$, according to the induction hypothesis $v_{2k}(r_1) = v_{2k-1}(g_0) + 2k = v_{2k-2}(g_0) + 2k$. For $k = 1$, $v_2(r_1) = v_1(r_1) = v_1(g_0) + 2 = v_0(g_0) + 2$. Thus, according to the lemma and the induction hypothesis, it holds that $v_{2k}(g_0) = m\, v_{2k}(r_1) + x_{2k}(g_0) = m\,(v_{2k-2}(g_0) + 2k) + k + 1$.

5: $s = r_i$ for $0 < k < i \leq m$ or $s = g_i$ for $0 < k \leq i \leq m$.

Values: Since down pass $2k$ starts at subroot r_k and ends at subroot g_0, Node Counting does not visit the subroots r_i for $i > k$ nor the subroots g_i for $i \geq k$ during down pass $2k$. Thus, according to the induction hypothesis, it holds that $x_{2k}(r_i) = x_{2k-1}(r_i) = 1$ and $v_{2k}(r_i) = v_{2k-1}(r_i) = 1$ for $i > k$, and $x_{2k}(g_i) = x_{2k-1}(g_i) = 1$ and $v_{2k}(g_i) = v_{2k-1}(g_i) = m + i + 1$ for $i \geq k$.

This completes the proof. ■

Thus, Node Counting reaches the goal node during up pass $2m + 1$. Setting $l = m$ in Equation (1) results in

$$v_{2m}(g_0) = \frac{m^{m+3} + m^{m+2} + m^{m+1} - 2m^3 - m^2 - m + 1}{m^2 - 2m + 1} > m^m.$$

For example, $v_{4,2}(g_0) = 35$ as shown in Figure 2.

Recall that $n = \frac{3}{2}m^2 + \frac{9}{2}m + 3$. Consider an arbitrary constant $0 < \epsilon < 1/6$. Assume that $m > 1/\epsilon - 4$. First, this implies that $1/m < \epsilon/(1 - 4\epsilon)$. Second, it implies that $m > 2$ and thus $n = \frac{3}{2}m^2 + \frac{9}{2}m + 3 < \frac{3}{2}\left(1 + \frac{4}{m}\right)m^2$. Put together, it follows that $n < \frac{3}{2}\left(1 + \frac{4\epsilon}{1-4\epsilon}\right)m^2 = \frac{3}{2}\frac{m^2}{1-4\epsilon}$ and thus $m > \sqrt{\frac{2}{3}n(1-4\epsilon)}$.

In the following, we utilize an inequality $(an)^k > n^{(1-\epsilon)k}$ valid for $n > (1/a)^{1/\epsilon}$ and $a > 0$. Hence, it holds for $n > m > \max\left(\frac{1}{\epsilon} - 4, \left(\frac{3}{2-8\epsilon}\right)^{1/\epsilon}\right)$ that

$$
\begin{aligned}
m^m &> \sqrt{2n(1-4\epsilon)/3}^{\sqrt{2n(1-4\epsilon)/3}} \\
&= ((2-8\epsilon)n/3)^{1/2\sqrt{2n(1-4\epsilon)/3}} \\
&> n^{(1-\epsilon)\sqrt{n(1-4\epsilon)/6}} = n^{\sqrt{(1-\epsilon)^2 n(1-4\epsilon)/6}} \\
&> n^{\sqrt{(1-2\epsilon)n(1-4\epsilon)/6}} = n^{\sqrt{(1/6-\epsilon+4\epsilon^2/3)n}} \\
&> n^{\sqrt{(1/6-\epsilon)n}}
\end{aligned}
$$

Hence, $v_{2m}(g_0) > m^m = \Omega\left(n^{\sqrt{(1/6-\epsilon)n}}\right)$, where $0 < \epsilon < 1/6$ is an arbitrarily small constant. Thus, the complexity of Node Counting on undirected graphs is $\Omega\left(n^{\sqrt{(1/6-\epsilon)n}}\right)$.

3 Conclusion

We conclude that the performance of Node Counting can be exponential in the number of states even if the domains are undirected trees. Determining a tight bound on the performance of Node Counting in undirected domains is still an open problem.

The construction of the tree can be generalized. Let the tree have $m+1$ levels with the bottom subtree having $t(m)$ leaves. To ensure that passes up change direction in subsequent levels and there are $2m+1$ passes altogether, the number of leaves of g-subroots must increase by one on each level. For inequality (2) to hold, we must also have $t(m)$ in $O\left(m^a\right)$, where $a > 0$. Hence, such a tree will produce $O\left(t(m)^m\right)$ visits to the bottom g-subroot and will have $n = O\left(mt(m) + m^2\right)$ nodes. A quick analysis shows that selecting $a = 1$ ensures the fastest asymptotic growth of the number of visits. Hence, $t(m)$ is a linear function of m, i.e. $t(m) = cm + d$ for some constants c and d. We can assume $d = 0$ because this coefficient does not impact asymptotic complexity of Node Counting. As a result, the tree has $n = (c+1/2)m^2 + o(m^2)$ nodes and therefore $m > \sqrt{\frac{2n}{1+2c}}$. For any given small constant ϵ we can select $c = \epsilon/2$ and for sufficiently large m (and corresponding to it n), the created tree will force Node Counting to make $\Omega\left(n^{\sqrt{n(1/2-\epsilon)}}\right)$ steps before reaching a goal.

Another generalization is to increase the number of leaves in the r-subtrees. Without changing the behavior of the algorithm we can have up to $m^{1-\beta},\ \beta > 0$ leaves for each r-subroot. The number of nodes in the tree will remain in $(1/2+c)m^2 + o(m^2)$ but each pass will now result in multiplication of the visits to leaves by $m^{1-\beta}$ before the result is added to the number of visits to the corresponding r-subroots. As a result, the complexity of Node Counting on such a tree is in $\Omega\left(n^{\sqrt{(2-\epsilon)n}}\right)$ and also in $\Omega\left(n^{\sqrt{n}}\right)$.

References

(Balch and Arkin, 1993) Balch, T. and Arkin, R. 1993. Avoiding the past: A simple, but effective strategy for reactive navigation. In *International Conference on Robotics and Automation*. 678–685.

(Bonet *et al.*, 1997) Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A robust and fast action selection mechanism. In *Proceedings of the National Conference on Artificial Intelligence*.

(Koenig and Simmons, 1996) Koenig, S. and Simmons, R.G. 1996. Easy and hard testbeds for real-time search algorithms. In *Proceedings of the National Conference on Artificial Intelligence*. 279–285.

(Finney and Thomas, 1994) Finney, R. and Thomas, G. 1994. *Calculus*. Addison Wesley, New York, NY.

(Korf, 1990) Korf, R. 1990. Real-time heuristic search. *Artificial Intelligence* 42(2-3):189–211.

(Pirzadeh and Snyder, 1990) Pirzadeh, A. and Snyder, W. 1990. A unified solution to coverage and search in explored and unexplored terrains using indirect control. In *Proceedings of the International Conference on Robotics and Automation*. 2113–2119.

(Russell and Wefald, 1991) Russell, S. and Wefald, E. 1991. *Do the Right Thing – Studies in Limited Rationality*. MIT Press.

(Smirnov and Veloso, 1997) Smirnov, Y. and Veloso, M. 1997. Gensat: A navigational approach. In *Proceedings of the Portuguese Conference on Artificial Intelligence*.

(Thrun, 1992) Thrun, S. 1992. The role of exploration in learning control with neural networks. In White, D. and Sofge, D., editors 1992, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold. 527–559.

(Wagner *et al.*, 1997) Wagner, I.; Lindenbaum, M.; and Bruckstein, A. 1997. On-line graph searching by a smell-oriented vertex process. In *Proceedings of the AAAI Workshop on On-Line Search*.