TRAVERSALS OF INFINITE GRAPHS WITH RANDOM LOCAL ORIENTATIONS

DAVID WHITE

Thank Danny, then committee, also Michel Dekking.

1. Background on random walks

Let G be a finite connected graph, but you can think of G as $\mathbb{Z}^2$ for what follows. A **random walk** on G is a sequence of vertices $X_0, X_1, X_2, \ldots$ where each X_{n+1} is chosen uniformly at random from the neighbors of X_n (i.e. each option with probability $1/d(X_n)$ or $1/4$ in $\mathbb{Z}^2$). It is **recurrent** if it always returns to the starting vertex. This forces it to returns infinitely many times: the walk after the n-th visit is again a random walk, so an $(n+1)$-st visit is guaranteed. Equivalently: for each vertex v, $Pr(X_n = v \text{ for infinitely many } n) = 1$, since there is a constant probability of walking from the origin to any vertex and given infinitely many tries this will occur with probability 1. The walk is **transient** if it is not recurrent. Equivalently: with probability 1 every vertex is visited only finitely often. If it doesn't return then we say the walk has escaped to infinity. All walks on a connected graph are either recurrent or transient.

Polya studied this question for graphs of the form $\mathbb{Z}^d$. In his formulation, $\mathbb{Z}^2$ was a city and the edges were city blocks. The particle undergoing the random walk was a drunkard, and the walk was called the "drunkard's walk." Polya proved the following amazing theorem:

Theorem 1. *A simple random walk on $\mathbb{Z}^d$ is recurrent if $d = 1$ or $d = 2$ and is transient for all $d > 2$.*

Shizuo Kakutani described this result as follows: "A drunk man will always find his way home, but a drunk bird may not."

2. Basic Walk

A problem computer scientists are interested in is **graph exploration by a mobile entity using only constant memory**. Useful if it's software moving on a network, searching for pages on the internet, or for a robot exploring an unfamiliar terrain. Simplest case: the graph is $[n] \times [n]$, or $\mathbb{Z}^2$ if you want to approximate letting the agent move continuously (i.e. any direction).

One option is to **give the agent no memory**. At a given vertex it picks a direction uniformly at random, i.e. with probability $1/4$ on the internal nodes, $1/3$ on the sides, and $1/2$ in the corners. The path of the robot is then a simple random walk. The problem is that we have no control over where it goes, how long it takes to cover all the ground, or how many times it re-vacuums the same spot. It turns out that this method will explore all n^2 vertices, but will take $O(|V| \log^2 |V|)$ time on average.

Leszek Gasieniec from Liverpool (Go-She-Nietz) considered another way to explore, where we give the agent 2 bits of memory and **help it out by labeling the edges** in the graph. From each

Date: May 8, 2012.

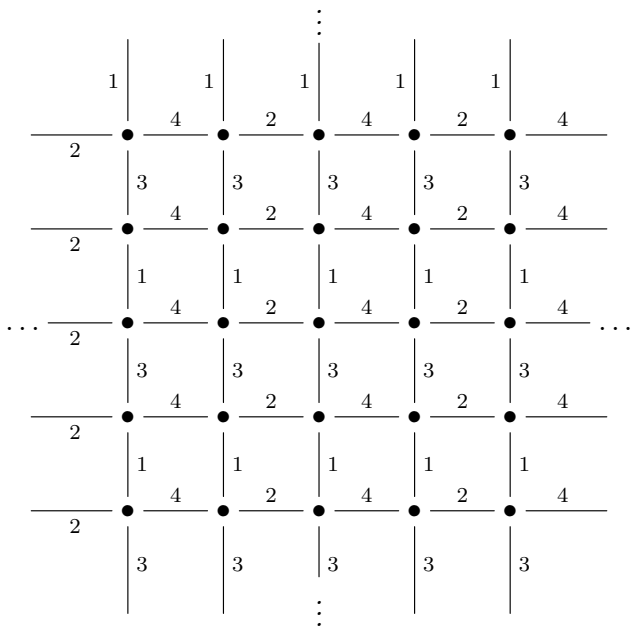
vertex, label the outgoing edges by $0, 1, 2, 3$ in such a way that each label is used once. Note that this means an edge can have two different labels; one from each direction. The agent only has to remember the label it just walked on and it can decide it's next label. When the agent enters a vertex by label $i \in \{0, 1, 2, 3\}$, it should exit following label $i \mod 4 + 1$. Practically, this means the agent can distinguish between the edges at any given vertex, i.e. **has a local orientation**. It is possible to assign a labeling s.t. the agent **explores the whole graph in $O(|V|)$ time**, but it requires the nodes to have a large memory and do a lot of work updating their labelings, so it might be infeasible from the point of view of applications.

Let's try to reduce the amount of memory we're requiring and make the labeling scheme easier. We'll just assign the labels randomly, all at once, and see what happens. This process is called the **Random Basic Walk**. Note that **all the randomness is in the initial labeling**. Once that's fixed and the starting vertex is chosen it's deterministic. Examples:

BOX GRAPH, STAR GRAPH

Note that with an arbitrary labeling a **agent can become trapped**. DRAW THE SQUIGGLY LINE AND CYCLE. Must replace "recurrence" with "gets trapped with probability 1" or "cycles." Then transient means it doesn't cycle. Do $\mathbb{Z}$ case, i.e. any random basic walk cycles. To avoid the trap on the line, the labels must alternate $1, 2, 1, 2, 1, 2, \ldots$. This occurs with probability $0 = \lim_{n\to\infty}(1/2)^n$.

Example 2. *A labeling in $\mathbb{Z}^2$ where any basic walk escapes to infinity:*



The reader can easily verify that every starting vertex and port number leads to an infinite staircase which moves in the directions specified by the first two steps and which gets further away from the starting location with every step. It is clear how this generalizes to $\mathbb{Z}^d$ with a staircase consisting of a sequence of moves, one in each of the d directions. This example generalizes to create an infinite family of examples where the basic walk escapes to infinity from any starting vertex and any initial label. We simply add in blocks of 4 columns which act like plateaus for the agent to move east or west for $4n$ steps between a given north-south step on the staircase.

3. Quasirandom and Self-Avoiding

Obviously, a big property of the Basic Walk is that it's **all defined locally**. There is another model you may have heard of which is similarly defined locally. It's the **Rotor Router model of Jim Propp**. It's **deterministic** (mention **random rotor router**) and it doesn't matter where you enter v from–you will leave by the direction the rotor points. You **can't trap** in any way. Also, there is a theorem which says it behaves like the simple random walk enough that it'll have the same recurrence/transience behavior.

Another big property of the Basic Walk is its ability to get trapped. The random basic walk acts like SRW at vertices which have never been visited before, but appears to act like a **self-avoiding random walk** at vertices which have been visited before. This is because if the walk previously left a vertex by label 2 and if it comes in by a label other than 1, then it must avoid the edge previously traveled. **As soon as the agent uses the same edge in the same direction it's in a cycle**. Self avoiding random walks can also get trapped. What's known is that for $d \leq 2$ the walk is expected to get trapped and for $d \geq 5$ there's enough space and it's expected to escape.

DRAW TABLE WITH RECURRENCE/TRANSIENCE, state conjecture

4. New Results for exploration with memory

Polya didn't have the option for his walk to get trapped in a small space. This option exists for the agent and it's exactly what we want to study. On the grid a trap configuration is where center vertex is entered from below by label i and the path goes left on $i+1$, back to center on $i+2$, right by $i+3$, and back to center by i, so the trap bounces the agent back and forth forever among these three vertices. Because these traps are small and local (they don't depend on how far you've come on the random walk), they occur with constant, nonzero probability. We'll use these traps to prove the following theorem:

Theorem 3. *In $\mathbb{Z}^2$, the random basic walk cycles with probability* 1.

Shells Method. The trap with 3 vertices occurs with constant probability $c > 0$. Draw concentric squares S_n (shells) of side length $2n$ centered at the origin. Let E_n be the event that the walk reaches S_n and the first time it does so is not a trapping configuration. Note that this "**first vertex hit**" cannot be a corner because corners are not adjacent to interior vertices. Because the trap exists entirely on the shell, it is independent of the walk up to that point.

Let E be the event that the walk escapes to infinity. It's not hard to see $E = \bigcap E_n$ because to escape to infinity you must never trap and you must pass each S_n. These E_n are not independent, but because $E_1 \subset E_2 \subset E_3 \ldots$, we can still write $P(E) = \prod P(E_n|E_{n-1})$. The probability of a path from S_{n-1} to S_n existing is ≤ 1. The probability that the first vertex on S_n hit is not a trap is $1-c$. Thus, $P(E_n|E_{n-1}) \leq 1 * (1-c)$ and so $P(E) \leq \prod 1 - c = 0$ because $c > 0$ implies $1 - c < 1$. □

This is not so surprising, since it's the same for Polya and for self-avoiding random walks. Much more surprising is that the method of proof generalizes to show:

Theorem 4. *For any d, the random basic walk on $\mathbb{Z}^d$ cycles with probability* 1.

Proof. For $d = 3$ each vertex is degree 6 and a trap can be found using only 3 neighbors (i.e. a trap can be found on a cubical shell around the origin). The trap occurs with constant probability $c' > 0$ and so the probability of escape is $\leq (1-c') * (1-c') * \cdots = 0$. The same idea works for

$d > 3$ as you can always find a trap on the surface of the **hypercube** and this means independence of the previous steps is no problem. This is because a vertex on S_n will have $2d-2$ neighbors on the shell and only needs d to make a trap.

Also worth noting: this proves that the probability of escape from the origin is zero. By symmetric, the probability of escape from any fixed vertex is zero. $Pr(\exists$ vertex which the walk can escape to infinity from$) = 0$ because it's a countable union of events, each of probability zero. □

5. Other locally finite graphs

To generalize this proof to other graphs you'd need them to have shells, and this comes down to expander properties. A graph which fails to have shells is the hexagonal lattice (think chicken-wire or honeycomb). We found a proof there for trapping and it generalizes to show that any regular graph has trapping basic walk. The trick is to use spires of length d where d is the degree. What about graphs which are not regular? Well, a third proof works in that case:

Theorem 5. *On any locally finite graph G with all vertex degrees bounded by a constant D, the basic walk cycles with probability 1.*

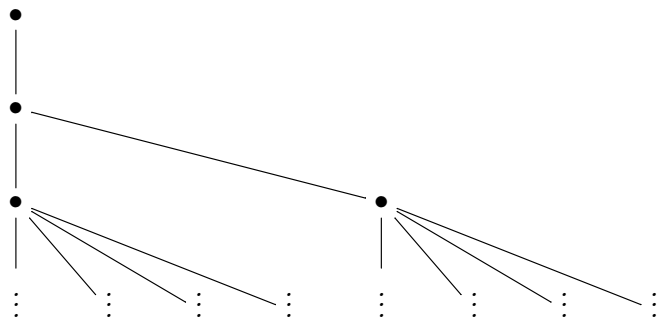
Star Method. Idea: Trap at v is a star, with $i-1$ pointing back at v_{-1} and for all neighbors w, if the arc from v to w is labeled by j, then the arc from w to v must be labeled by $j' = (j \bmod d(w)) + 1$. Need $d(v_{-1}) \leq d(v)$, which occurs infinitely often if the agent is to escape, using the hypothesis of bounded degree. Whenever the basic walk takes such a step, we consider the event E that the configuration C_v is achieved. Let N denote the set of neighbors of v which are not v_{-1}. Because $d(v) < D$, $|N| < D$. Furthermore, $d(w) < D$ for all $w \in N$, so

$$P(E) = \frac{1}{d(v)} * \prod_{w \in N(v)} \frac{1}{d(w)} > c = \frac{1}{D} * \left(\frac{1}{D}\right)^D > 0$$

□

The final question I will consider is whether or not this hypothesis of bounded degree is necessary. It's clear that finite degree is necessary to even define the process.

Example 6. *Let T be the tree where every vertex in level n has 2^n children. As usual, the root is in level 0.*



Then the random basic walk on T is transient, i.e. has nonzero probability of escaping to ∞.

Proof. Sketch: $E \supset P$ so $\Pr(E) \geq \Pr(P) = \prod \Pr(P_n \mid P_{n-1})$. Note that $\Pr(P_m \mid P_{m-1}) \geq 1 - \frac{1}{2^{m-1}}$. Thus:

$$\Pr(P) \geq \prod_{m=2}^{\infty} 1 - \frac{1}{2^{m-1}}$$

$$= \prod_{n=1}^{\infty} 1 - \frac{1}{2^n} \text{ and taking logs yields } \ln(\Pr(P)) \geq \sum_{n=1}^{\infty} \ln\left(1 - \frac{1}{2^n}\right)$$

Recall the Taylor Series expansion of $\ln(1-x)$ around 0:

$$\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \dots \text{ for } -1 \leq x < 1$$

$$\ln(\Pr(P)) \geq \sum_{k=1}^{\infty} \sum_{n=1}^{\infty} \frac{-1}{2^{kn}} \geq \sum_{k=1}^{\infty} \frac{-1}{2^{k-1}} = -2$$

To show $2^{-k}/(1-2^{-k}) \leq 2^{-k+1}$ as needed in the last inequality, note that $1 - 2^{-k} \geq 1/2$ so that $2^{-k}/(1-2^{-k}) \leq 2^{-k}/(2^{-1}) = 2^{-k+1}$.

Undoing the log shows that $\Pr(\text{escape}) \geq \Pr(P) \geq e^{\ln(\Pr(P))} \geq e^{-2} > 0$. This proves there is a positive chance that the agent escapes, i.e. the random basic walk is transient. □

6. Finite Graphs

The **original problem was seeking a supersize tour** on $G_{k,n}$, i.e. proving the **expected maximum number** of vertices visited is $c * |V|$. From the theorems we can get **upper bounds on expected average number** of vertices visited as $k, n \to \infty$. Also, we cannot get bounds on the expected max number because:

Proposition 1. *Let G be a locally finite, d-regular graph with port orientations selected uniformly at random. Then the following statement holds with probability 1: for all $n \in \mathbb{N}$ there are infinitely many pairs (v, ℓ) such that a random basic walk starting at v with initial port ℓ will visit n vertices.*

Proposition 2. *Let G be a locally finite, d-regular graph with $d > 2$ and with port orientations selected uniformly at random. Then the following statement holds with probability 1: for all $n \in \mathbb{N}$ there is some vertex v which is contained in a cycle of length greater than n. Indeed, there are infinitely many such v.*

Proof idea: pick infinitely many spaced out vertices. Each has a probability of a labeled path going out of length n, which could also be a spire. With infinitely many tries you'll achieve this event. Indeed, you'll get it infinitely many times. Note, it's unlikely the agent will hit one of these, so the question on $G_{k,n}$ is still open.

Theorem 7. *As $n \to \infty$, a basic walk on K_n is expected to visit at least $(1 - 1/e) * n$ nodes*

Conjecture 8. *The expected number of arcs traversed by a basic walk on K_n is $1.8 * n$ as $n \to \infty$.*

7. Future Directions

- In $\mathbb{Z}^2$, what is the expected length of time (i.e. the expected number of steps taken) before the random basic walk hits a cycle? What is the expected size of a cycle? What about these three questions on $G_{k,n}$?
- Study the constrained random basic walk where the constraint means there are no vertices which have two different incoming arcs with the same label.
- Create and study an analogue of the hitting time, mixing time, and load balancing for the random basic walk.
- Study random rotor routers and determine if they bear any resemblance to the random basic walk.

8. Proofs

Obviously, a big property of the Basic Walk is that it's all defined locally. There is another model you may have heard of which is similarly defined locally. It's the **Rotor Router model of Jim Propp**. It's **deterministic** and designed to give the **same limiting behavior but with faster convergence**. Place a rotor at each v, like what balls come out of at the batting cages. Next, fix a rotor pattern $e_1, e_2, \ldots, e_{d(v)}$ running through all edges out of v and point the rotor via e_1. When v is first visited, the particle exits by e_1 and then the rotor rotates to e_2. When v is next visited, the particle exits by e_2 and we repeat this process. These rotor routers have been studied quite a bit in the past 10 years, and have found numerous applications, e.g. load balancing. It's supposed to be **"better than random"** because the central limit theorem behavior is achieved immediately, e.g. on $[n]$ it will have exactly half the particles (every other particle) leave by the left as by the right, which the CLT predicts but only with an error depending on the number of trials.

Differences from Basic Walk: It doesn't matter where you enter v from, you will leave by the direction the rotor points. The scheduling policy is purely local and deterministic–there is no randomness at all. You can't trap in any way. Also, there is a theorem which says it behaves like the simple random walk enough that it'll have the same recurrence/transience behavior (in terms of number of times the origin is visited if you start a large number of particles at once) so it's recurrent in $\mathbb{Z}^2$ and transient in $\mathbb{Z}^d$ for $d > 2$.

Another big property of the Basic Walk is its ability to get trapped. The random basic walk acts like SRW at vertices which have never been visited before, but appears to act like a **self-avoiding random walk** at vertices which have been visited before. This is because if the walk previously left a vertex by label 2 and if it comes in by a label other than 1, then it must avoid the edge previously traveled. As soon as the agent uses the same edge in the same direction it's in a cycle. Self avoiding random walks can also get trapped. What's known is that for $d \leq 2$ the walk is expected to get trapped and for $d \geq 5$ there's enough space and it's expected to escape. Based on this and the example above, the problem group conjectured that the agent would get trapped (this is the analog of recurrence) in $\mathbb{Z}^2$ but would escape in sufficiently high dimension.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Before progressing, we wish to note that for any graph G, the way in which labels are placed on arcs may be changed. Rather than labeling every arc simultaneously at the start, we may define just one new label at each step. Suppose the robot enters vertex v by port i (for the starting vertex an initial port is provided, which we will assume is 1 in our examples). If v has an outgoing arc labeled by $(i \mod d(v)) + 1$ already then the robot will follow this arc and no labeling will be done.

If v does not have such an arc, then there must be a non-empty set of arcs leaving v with no label. Choose one of these arcs uniformly at random and assign it the label $(i \mod d(v)) + 1$.

We sketch an argument that the formulations above are equivalent. If all port numbers at v are assigned simultaneously, then each outgoing arc has probability $1/d(v)$ of receiving a fixed port number i. We claim that if the port numbers are assigned one at a time, only as needed by the robot, then the same probabilities are achieved. For the first outgoing arc, the probability of receiving any given port number i_1 is obviously $1/d(v)$. For the second arc to be labeled, the probability is $\frac{d(v)-1}{d(v)} * \frac{1}{d(v)-1} = \frac{1}{d(v)}$ because we must first know that the port number chosen is not i_1, and then we have $d(v) - 1$ choices for which i_2 will be chosen of the $d(v) - 1$ possibilities remaining. For the k-th arc to be labeled, we must first know that $i_1, i_2, \ldots, i_{k-1}$ are not chosen and then there are $d(v) - k$ possibilities remaining. So the probability is $\frac{d(v)-k}{d(v)} * \frac{1}{d(v)-k} = \frac{1}{d(v)}$. This proves the two random processes are the same, so we are free to think of the assignment of port numbers as occurring one assignment per step of the robot.

Star Method. The pigeonhole principle guarantees us that there are infinitely many vertices v with a neighbor w of degree $d(w) \geq d(v)$. This is because every time a vertex v only has neighbors of smaller degree, all those neighbors have a neighbor (v) with larger degree. If the basic walk is to have any chance of escaping to infinity, then it must be the case that infinitely often the agent moves from a vertex v to a vertex w such that $d(v) \geq d(w)$. Here we are using the hypothesis of bounded degree, which means the agent cannot move from a vertex of degree 1 to one of degree 2, then one of degree 3, etc. Such a chain would eventually hit a vertex of degree D and then need to move to one of degree $\leq D$. We label these steps of the agent (from larger degree to smaller degree) by $w_1 \to v_1, w_2 \to v_2, \ldots$, where we do not assume $w_i \neq w_j$ for $i \neq j$ but we do assume $v_i \neq v_j$ by simply removing the pairs (w_i, v_i) where v_i has appeared in the list before. Clearly this will not change the fact that there are infinitely many such pairs.

Whenever the basic walk takes a step $w_i \to v_i$ from the list above, we consider the event E_i that the configuration C_{v_i} is achieved. Let N denote the set of neighbors of v_i which are not w_i. Because $d(v_i) < D$, $|N| < D$. Furthermore, $d(x) < D$ for all $x \in N$, so

$$P(E_i) = \frac{1}{d(v)} * \prod_{x \in N(v)} \frac{1}{d(x)} > c = \frac{1}{D} * \left(\frac{1}{D}\right)^D > 0$$

In order for the basic walk to escape to infinity, this event must be avoided for infinitely many v_i. So the probability p that the basic walk escapes to infinity satisfies $p \leq (1 - c) * (1 - c) * \cdots = 0$, proving that the basic walk cycles with probability 1. □

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Example. A random basic walk on T will always have a higher probability of going downwards than of going upwards. It is clear that the random basic walk starting at the root will be more likely to cycle than a random basic walk starting at a vertex lower down on T, since the probability of returning will always be higher near the root. Thus, we will focus on the case where the initial vertex is the root. From the root, the initial step is determined because the root has degree 1.

Define an event P to be "for each i, the agent is distance i away from the root at step i" i.e. "all steps are away from the root." Note that $\Pr(\text{escape}) \geq \Pr(P)$ since P is a way for the agent to escape. Define events P_i to be "at step i the agent is distance i from the root." Clearly, $P_0 \supset P_1 \supset P_2 \supset \cdots \supset P = \bigcap P_i$, so $\Pr(P) = \prod \Pr(P_n \mid P_{n-1})$. Because each vertex has only one arc pointing

back at the root, $\Pr(P_0) = Pr(P_1) = 1, \Pr(P_2 \mid P_1) = 1 - \frac{1}{3} \geq 1 - \frac{1}{2}, \Pr(P_3 \mid P_2) = 1 - \frac{1}{5} \geq 1 - \frac{1}{4}$, and in general

$$\Pr(P_m \mid P_{m-1}) = 1 - \frac{1}{2^{m-1}+1} \geq 1 - \frac{1}{2^{m-1}}. \text{ Thus: } \Pr(P) \geq \prod_{m=2}^{\infty} 1 - \frac{1}{2^{m-1}}$$

$$= \prod_{n=1}^{\infty} 1 - \frac{1}{2^n} \text{ and taking logs yields } \ln(\Pr(P)) \geq \sum_{n=1}^{\infty} \ln\left(1 - \frac{1}{2^n}\right)$$

Note that the inequality is preserved after applying ln because ln is an increasing function. Proving $\Pr(P) > 0$ is equivalent to proving this sum is greater than $-\infty$. Recall the Taylor Series expansion of $\ln(1-x)$ around 0:

$$\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \dots \text{ for } -1 \leq x < 1$$

Because $-1 \leq 1/2^n \leq 1$ for $n \geq 1$, this equality holds. Thus:

$$\ln(\Pr(P)) \geq -\sum_{n=1}^{\infty} \frac{1}{2^n} - \frac{1}{2}\sum_{n=1}^{\infty} \frac{1}{2^{2n}} - \frac{1}{3}\sum_{n=1}^{\infty} \frac{1}{2^{3n}} - \dots - \frac{1}{k}\sum_{n=1}^{\infty} \frac{1}{2^{kn}} - \dots \geq$$

$$\sum_{n=1}^{\infty} \frac{-1}{2^n} + \sum_{n=1}^{\infty} \frac{-1}{2^{2n}} + \sum_{n=1}^{\infty} \frac{-1}{2^{3n}} \dots = \frac{-1/2}{1-1/2} + \frac{-1/2^2}{1-1/2^2} + \frac{-1/2^3}{1-1/2^3} \dots \geq \sum_{n=0}^{\infty} \frac{-1}{2^n} = -2$$

To show $2^{-k}/(1-2^{-k}) \leq 2^{-k+1}$ as needed in the last inequality, note that $1 - 2^{-k} \geq 1/2$ so that $2^{-k}/(1-2^{-k}) \leq 2^{-k}/(2^{-1}) = 2^{-k+1}$.

Undoing the log shows that $\Pr(\text{escape}) \geq \Pr(P) \geq e^{\ln(\Pr(P))} \geq e^{-2} > 0$. This proves there is a positive chance that the agent escapes, i.e. the random basic walk is transient. □

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Prop1. Fix n and let v be an arbitrary vertex. Because G is infinite, there must be some path in G of length n which always moves away from v in the graph metric. Let $P_{n,v}$ be the event that one such path has the appropriate labels so that a random basic walk starting at v with initial port ℓ will move directly away from v along the path for n steps. The probability of $P_{n,v}$ is a constant, non-zero number $c_n = (1/d)^n$ which does not depend on v. Select a sequence of vertices $(v_1, v_2, \dots)$ which are all distance at least $2n$ away from each other, so the events P_{n,v_i} are independent from each other. Then the probability that none of the events P_{n,v_i} occur is $(1-c_n)*(1-c_n)*(1-c_n)*\dots = 0$. This proves there is some v_j which has a path of length n going out.

Let $w_1 = v_j$. Removing v_j from the list $(v_1, v_2, \dots)$, one can repeat the same argument and conclude that some other v_k must have a path of length n going out, since the probability of not having such a v_k is $(1-c_n)*(1-c_n)*\dots = 0$. Set $w_2 = v_k$. Repeating this ad infinitum proves that there is an infinite sequence $(w_1, w_2, \dots)$ each of which has a path of length n going out. □

Prop2. Fix $n \in \mathbb{N}$ and let M denote the first multiple of d which is greater than n. By Proposition above, there is an infinite sequence of vertices $(w_1, w_2, \dots)$ each having a path of length M going out. For each w_i there is a constant, nonzero probability k_n that this path is actually a spire, i.e. that the random basic walk will move out to the end of this path, then turn around and return to w_i. Because d divides the length of the path, this spire will be a cycle using M vertices, i.e. the random basic walk will traverse it back and forth forever. Note that $d > 2$ is needed in order for the random basic walk to return along the spire rather than getting trapped in a cycle on the final two vertices of the spire.

The probability that none of the paths from Proposition above are spires is $(1-k_n)*(1-k_n)*\cdots = 0$. Thus, there must be some w_i which is the first vertex in a cycle of length greater than n. Set $z_1 = w_i$ and remove w_i from the sequence $(w_1, w_2, \dots)$. Repeating the argument above proves that there is some $z_2 = w_j \neq w_i$ which is the first vertex in a cycle of length greater than n. Continuing forever gives an infinite sequence $(z_1, z_2, \dots)$ where each z_i is the first vertex in a cycle of length greater than n. □

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Complete. We prove a stronger statement, namely that $(1-1/e)*n$ of the nodes are expected to be visited within just the first $n-1$ steps. Restricting to the first $n-1$ steps guarantees that cycles are impossible, since any cycle in a d-regular graph a cycle requires at least d arcs. Label the vertices $0, 1, 2, \dots, n-1$ and assume that the starting vertex v_0 is labeled by 0. Consider the sequence of vertex numbers $(v_0, v_1, v_2, \dots)$ visited by the agent. This sequence cannot have adjacent numbers equal, since K_n has no loops. Furthermore, this sequence cannot have a pair of adjacent numbers occur twice, since acyclicity implies no arc is traversed twice.

Let $v \neq v_0$ be a vertex chosen at random. We will prove that the probability that v is visited is $\geq 1 - 1/e$. The probability that v is visited on step 1 of the agent is $\Pr(v = v_1) = 1/(n-1)$, since there are $n-1$ possible steps the agent could take after v_0. If v is not visited in step 1, then the probability that v is visited on step 2 of the agent is $\Pr(v = v_2 \mid v \neq v_1) = 1/(n-1)$. Step 3 is more complicated, because it is possible that $v_2 = v_0$, in which case the arc $v_2 \to v_1$ already has a label. So there are two ways to get $v = v_3$ given that v has not been visited previously:

$$\begin{aligned}\Pr(v = v_3) &= \Pr(v = v_3 \mid v_2 = v_0) * \Pr(v_2 = v_0) + \Pr(v = v_3 \mid v_2 \neq v_0) * \Pr(v_2 \neq v_0) \\ &= \frac{1}{n-1} * \frac{1}{n-2} + \frac{1}{n-1} * \frac{n-2}{n-1} > \frac{1}{n-1} * \frac{1}{n-1} + \frac{n-2}{n-1} * \frac{1}{n-1} = \frac{1}{n-1}\end{aligned}$$

The cases for $v_4, v_5, \dots$ get even more complicated, but the point remains that $1/(n-1)$ is always a lower bound on the probability that v is first visited in step $i+1$. This is because if v_i has been visited k times before, then the existence of these previous visits rules out more of the possible arcs the agent could follow. This can only increase the probability that the next vertex to be visited is one the agent has not visited before (e.g. v), and this makes the messy computations summations which factor in the entire path of the agent unnecessary. Formally, the probability that v is first visited on the $i+1$-st step given that v_i has been visited k times previously will be $1/(n-k) \geq 1/(n-1)$. This proves that $\Pr(v$ not visited on i-th step$) \leq 1 - 1/(n-1)$ for all i, which implies

$$\Pr(v \text{ not visited in first } n-1 \text{ steps}) = \prod_{i=1}^{n-1} \Pr(v_i \neq v) \leq \left(1 - \frac{1}{n-1}\right)^{n-1}$$

As $n \to \infty$, this bound tends to $1/e$, so the probability that v is visited in the first $n-1$ steps is at least $1 - 1/e$. Thus, the expected number of vertices missed is $\leq n/e$ and the expected number of vertices visited is $\geq (1 - 1/e) * n$. □

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Applications: Network routing, rumor routing, searching, query processing, load balancing on distributed networks, self-stabilization of such networks as a way to counteract transmission failure, energy savings on large networks, image processing, exploration of unknown terrain, and clustering. There is an increasing number of monitoring applications which make use of a large network of small, smart sensors. Many of these applications discussed above rely on simple random walks because of their simplicity of implementation, savings on time and memory, and local nature. The random basic walk shares many of these features.

Rumor routing is a compromise between flooding queries and flooding event notifications on a network. Rumor routing works by creating paths which lead to each event, so queries move on the network via a simple random walk to find the event path to the correct event. Peer-to-peer networks are more general than sensor networks or the internet network. Another type of network is an ad-hoc network, which relies on wireless links between entities rather than infrastructure such as telephone lines. Due to partial transmission failure, failure of communication links, and noisy transmission, this is a field where algorithms which can stabilize themselves after a failure are highly valued.

As many of these applications make use of facts about cover time and load balancing from the theory of simple random walks, the analogous notions for the random basic walk would need to be developed. One application of random walks to sensor networks is [6], which focuses on robust query processing. This paper suggests an application of the random basic walk because only a constant fraction of the sensor network needs to be visited. Another application is to allow nodes to switch from active to inactive and vice versa at random times (e.g. to save energy). You can study routing in this context via constrained random walks on dynamic graphs. You'd need to develop constrained random basic walks.

One famous algorithm for searching the internet is PageRank. Another is topic-sensitive PageRank, which computes the stationary probability distribution coming from a simple random walk on websites.