Hunting a Random Walker on a Low Dimensional Torus

Yariv Yaari

Abstract

We will define a problem regarding locating a simple random walk on a graph, and give bounds for the optimal solution on toruses of dimension 2,3.

1 Hunting a Random Walker

There are several natural variants of the hunting problem, here is one. A bird performs a random walk on a graph. At each step a hunter picks a vertex and either catches the bird or gets to see if the bird visited there before (and possibly when) or not. Will a clever hunter almost surely catch the bird? In particular when the bird performs a fat tail symmetric random walk on $\mathbb{Z}$, how big should the tail be to guarantee escaping with positive probability? Can it escape once transient?

One can study versions of the problem for infinite countable Markov chains (or other random processes), in which the question is: will the hunter catch the bird almost surely? Or versions where the hunter is catching the bird almost surely, and the question is how fast the hunter can do it, this can be studied for finite chains as well. Other aspects to consider is the amount of information available for the hunter. Does the hunter get to see the time the bird visited the vertex or just if it ever visited there before, or did the bird visit again since the last time the hunter inspected the vertex. Further limitations on the hunter can involve the size of the memory of the hunter. Benjamini and Kozma (2010) (unpublished), showed that if the underling graph is a d tree and the bird is performing a uniform directed walk away from the root, then there is a hunter that will almost surely catch the bird iff $d < 4$.

In this paper we show an algorithm for the hunter of a SRW on a torus of dimension 2 or 3, and analyze its expected hunting time complexity, comparing it to a simple lower bound on the complexity for any algorithm. The algorithm will need to know a little more than whether the walk visited the vertex, namely it will need to know whether it visited the vertex since the previous examination of this vertex. This is the most limited information in which we found a non trivial solution. The SRW can be replaced by any symmetric walk with steps distribution with a second moment.

2 Results

2.1 Hunting a Random Walker

2.1.1 Definitions

Let us define the problem more formally. Let $d \in \{2,3\}$, and $n \in \mathbb{N}$. Let $G = (V, E)$ be a d dimensional torus of side-length n. Let $\{Z_i\}$ be an SRW on G, Z_0 uniform on G. Also let f_m be a function of $V^m \times \mathbb{N}^{<m}$ which we will call the information function. f_m will always be invoked on some $((v_1, v_2, ..., v_m), i_1, i_2, ...i_l)$ where the i_j are the set of times s.t. $Z_{i_j} = v_m$ and $i_j \leq m$. We will use the information function in the definition of a hunter algorithm, and there those v_i will always be a list of queries the hunting algorithm performed, and the information will simply be the information it receives from the last query (the m query). We will work with $f_m((v_1, v_2, ..., v_m), i_1, i_2, ...i_l)$ being 1 iff $\forall j < m, (\forall k \leq lj \geq i_k) \rightarrow v_j \neq v_m$ and 0 otherwise. This means that f_m is 1 iff the walk visited v_m after the last query, except the m, that indicated v_m as a vertex. This is a very limited notion of information, but it continues to generate new information all the time. A hunter is an algorithm A with access to an oracle H. The oracle has a state, it receives a single vertex of V as Input. Let v_j be the input on the j invocation of H, then H returns $f((v_1, v_2, ..., v_m), i_1, i_2, ...i_l)$ on the m invocation. The hunting time T_A is $min\{i | v_i = Z_i\}$.

2.1.2 results

2.2 the 2-dimensional case

In this section, $d = 2$.

Proposition 2.1. *For any hunting algorithm A, $\mathbf{E}[T_A] = \Omega(n\sqrt{\log n})$.*

Proof. The proof is a simple counting argument. Define T, $T = min(t | \exists i \leq t, v_t = Z_i)$. Clearly $T \leq T_A$, so we will show $\mathbf{E}[T] = \Omega(n\sqrt{\log n})$, by showing that for some c and sufficiently large n, $\mathbf{P}[T > cn\sqrt{\log n}] > \frac{1}{2}$. To estimate $\mathbf{P}[T > k]$ we define v'_i to be the queries of A with the oracle that returns 0 on all invocations. Note that for $i \leq T$, $v_i = v'_i$. Therefore $\mathbf{P}[T > k] = \mathbf{P}[T' > k]$, where T' is defined exactly as T with v_t replaced by v'_t. Now

$$\mathbf{P}[T' > k] > \mathbf{P}[\{v'_i\}_{i \leq k} \cap \{Z_i\}_{i \leq k} = \phi],$$

and since both sets are chosen independently, and $\{Z_i\}_{i \leq k}$ is translation invariant we get

$$\mathbf{P}[\{v'_i\}_{i \leq k} \cap \{Z_i\}_{i \leq k} = \phi] = 1 - \frac{\mathbf{E}[\#\{v'_i\}_{i \leq k}] \cdot \mathbf{E}[\#\{Z_i\}_{i \leq k}]}{n^2}.$$

Now, $\#\{v'_i\}_{i \leq k} = k$ and $\mathbf{E}[\#\{Z_i\}_{i \leq k}] = \Theta(\frac{k}{\log k})$, so for some $c > 0$ we get

$$\mathbf{P}[T > cn\sqrt{\log n}] > \frac{1}{2}.$$

□

Now let us describe a hunting algorithm. We will define the algorithm in terms of four constants $C_1, C_2, i_0, \epsilon_1$, then show that these can be chosen to achieve a desired hunting time. We will describe it using two internal variables, the level i and vertex v, they will always satisfy that the walk was at v at most $2t_i$ before the present, where t_i is 2^{2^i},or $i = \log_2 \log_2(n(\log n)^{\frac{1}{2}-\epsilon_1}) + 1$, it should be obvious from the definition that this hold. First we will define the 'Grid Search' operation, we generate $\{u_j\}_{j<C_1^2\sqrt{t_i}}$, a square grid centred in v with distances $t_i^{\frac{1}{4}}$ between adjacent grid points and $C_1 t_i^{\frac{1}{4}}$ points along each cardinal direction. The whole grid is of side length $C_1\sqrt{t_i}$. We would like to test all nodes of the grid for a point the walk was in $\sqrt{t_i}$ before it was tested. Finding such a point is simply a matter of splitting the grid to sets of size $\sqrt{t_i}$, and checking each one once (ignoring the information, this is just 'resetting' the points) than checking it again. The number of points in each set is equal to the time window we are looking at, as it is the temporal distance between two queries for any point. If we find any such point (visited up to $\sqrt{t_i}$ before its second query) we will move to the $(i-1)$'th level, setting v as this point. If none was found, the 'Grid Search' failed. The algorithm on the i'th level (for $i \leq \log_2 \log_2(n(\log n)^{\frac{1}{2}-\epsilon_1})$) is simply invoking 'Grid Search' $C_2 \log t_i$ times or until successful (on success, moving to the $(i-1)$'th level as noted above). If all the grid searches failed, move to level $\log_2 \log_2(n(\log n)^{\frac{1}{2}-\epsilon_1}) + 1$ around the origin. The algorithm starts on the top level, $\log_2 \log_2(n(\log n)^{\frac{1}{2}-\epsilon_1}) + 1$. The behaviour on the top level is different, looking for a point visited $n(\log n)^{\frac{1}{2}-\epsilon_1}$ time ago in the whole torus. This is done by choosing a random uniform set of size $n(\log n)^{\frac{1}{2}-\epsilon_1}$ under the constraint that no two points in the set are closer than $\sqrt{\frac{n}{10(\log n)^{\frac{1}{2}-\epsilon_1}}}$, and checking it in the same manner (resetting it, then checking). i_0 is the minimal level. Once we reach this level, we make a query to a random uniform vertex within distance t_{i_0} from v then return to the top level. (Note: the algorithm have no stopping condition, but we are only interested in the hunting time)

Proposition 2.2. *For any $\epsilon > 0$ there are $C_1, C_2, i_0, \epsilon_1$ such that the hunting algorithm $A_{C_1,C_2,i_0,\epsilon_1}$ described above has $\mathbf{E}[T_{A_{C_1,C_2,i_0,\epsilon_1}}] = O(n(\log n)^{\frac{1}{2}+\epsilon})$.*

Proof. We will get an upper bound on $T_{A_{C_1,C_2,i_0,\epsilon_1}}$ by estimating the time until a query done on level i_0 will hit the walk. Clearly, whatever i_0 is, hitting the walk have a probability bounded away from 0 every time we get to level i_0, so we will have to calculate two things, the probability that the algorithm will get there starting from level $\log_2 \log_2(n(\log n)^{\frac{1}{2}-\epsilon_1})$, and the time it takes to get to level $\log_2 \log_2(n(\log n)^{\frac{1}{2}-\epsilon_1})$ from the top level (the time for all the levels below it is $O(\sqrt{(n(\log n)^{\frac{1}{2}-\epsilon_1}) \log n})$, lower order than the time spent on the top level). Our bound will then be the time on the top level divided by the probability to get to level i_0. i_0 itself will be chosen to have behaviour close to the asymptotic bounds of the SRW.

we will take ϵ_1 to be $\frac{\epsilon}{2}$, and show that there are C_1, C_2, i_0 such that on any level $i \geq i_0$ the probability of moving to a lower level is $p_i > 2^{-\epsilon_1} - O(2^{-i})$, conditioned on the entire history of the process. Given this, the probability of moving all the way from level $\log_2 \log_2(n(\log n)^{\frac{1}{2}-\epsilon_1})$ to i_0 will be the product of the p_is, and is $\Omega((\log n)^{-\epsilon_1})$. We will also show that the total time every such attempt takes is $O(n(\log n)^{\frac{1}{2}+\epsilon_1})$, and so we get the required total expected hunting time.

The main observation that will allow us to handle this conditioning on the entire history

is that on entering level i we know the walk visited v at some time window, from $t-t_i$ to t. If we can show that we get this bound on p_i for every time in this window, we could use the markovian behaviour and ignore the history up to the visit to v. That is for every event A, if we define F_j for the event $Z_j = v$ and $\forall k, t-t_i \leq k < j \rightarrow Z_k \neq v$ (j is the first visit to v in the window) we get

$$\begin{aligned}\mathbf{P}[A|\text{entire history}] &= \sum_{j=t-t_i}^{t} \mathbf{P}[F_j|\text{entire history}] \cdot \mathbf{P}[A|F_j \cap \text{entire history}] \\ &= \sum_{j=t-t_i}^{t} \mathbf{P}[F_j|\text{entire history}] \cdot \mathbf{P}[A|Z_j = v \cap \text{history after time } j]\end{aligned}$$

and since $\sum_{j=t-t_i}^{t} \mathbf{P}[F_j|\text{entire history}] = 1$, we get

$$\mathbf{P}[A|\text{entire history}] \geq \min_{j \in [t-t_i, t]} \mathbf{P}[A|Z_j = v \cap \text{history after time } j].$$

There is also a corresponding upper bound using the maximum, of course. We will also use a similar argument using a space-window. That is conditioning on the walk to be in some space at a given point in time, and using the same argument we get that bounds that can be shown for every point in this space-window and without any history before will hold when conditioning on the space-window with any history.

So now let us start in a 'Grid Search' with a specific vertex u of the grid and an upper bound on the probability of the walk visiting any vertex in the grid on its query-window, conditioned on visiting u in its query-window (conditioning on the history here is irrelevant, we could condition on the walk being somewhere in the torus at the beginning of the 'Grid Search' to justify ignoring the history). We will use this estimate to show that the conditioning on history after time j mentioned above is negligible. To estimate we will first use an upper bound for two specific vertices, visiting u_2, conditioned on visiting u_1. First, we take for an upper bound twice the probability to travel from u_1 to u_2 in the diameter of the union of query-window (twice, for order of visits). This we will bound again by the product of reaching to distance $d(u_1, u_2) - t_i^{\frac{1}{4}}$ from u_1 in time $2C_1^2\sqrt{t_i}$, times the probability to hit u_2 from distance $t_i^{\frac{1}{4}}$ in time $2C_1^2\sqrt{t_i}$ (taking the point in such distance with maximal probability to hit u_2). The second part is clearly independent of actual choices of u_1,u_2, and it is $O(2^{-i})$, since in that time only $O(\frac{\sqrt{t_i}}{\log t_i})$ vertices will be visited and there are $\Theta(\sqrt{t_i})$ that are at least as likely to be hit as u_2. The first part is decaying exponentially in $d(u_1, u_2)$, and so we could sum on all possible choices of u_2 and get $O(\frac{1}{\log t_i})$, with no dependence on C_1 even for the constant hidden in the O notation (that is, there is a bound independent of C_1, which we get by summing the infinite sum, ignoring the fact that the grid is actually finite). Now, when entering level i, the only queries after the start of the time-window are those done in the same 'Grid Search'. All points tested before v were not visited in their query-window, otherwise v would not have been tested, but the estimate above tells us that the conditioning on this will only change probabilities by $O(\frac{1}{\log t_i}) = O(2^{-i})$, which is the reason this appear in the definition of p_i. From now on we will therefore ignore the conditioning on the history, to get the rest of p_i.

Let us now look at the probability for success in one of the C_2 iterations of 'Grid Search', conditioned on the walk being in an arbitrary point v' in a box of side length $C_1\sqrt{t_i}$ centered in v (a box that matches the grid), at the beginning of the 'Grid Search'. There is a point u in the grid with $d(u,v') \leq t_i^{\frac{1}{4}}$, and the probability of the walk visiting it in its query-window is $\Omega(\frac{1}{\log t_i})$, simply because the window is of size $\sqrt{t_i}$ and starts $O(\sqrt{t_i})$ after the walk was in v'. We will use this as a lower bound on the probability of the walk hitting at least one query-window and call it q_i.

To estimate the probability of successfully moving from level i to level $i-1$ we need to do one last thing. Since we perform multiple 'Grid Search' attempts (if not successful), we will estimate the probability of a 'Grid Search' operation to succeed, conditioned on the failure of the previous ones. This we can bound by the probability for success conditioned on being in a box of side length $C_1\sqrt{t_i}$ centred in v at the beginning of the 'Grid Search' (this conditioning 'masks' the conditioning on previous failures), times the probability of being there conditioned on the failure of previous iterations. The conditioned probability of being in the box is easily bounded by the unconditioned probability of being in the box at the correct time minus the probability of succeeding in at least one of the previous iterations. The unconditioned probability of being in the box can be bounded for all iterations, call this bound α. So if we call the probability to succeed in at least one of the first j iterations x_j we get the following form:$x_{j+1} \geq x_j + q_i(\alpha - x_j)$. Using $y_j := \alpha - x_j$ we get $y_{j+1} \leq y_j(1-q_i)$, and $y_j \leq \alpha(1-q_i)^j$. so we get

$$\mathbf{P}[\text{successfully moving from level } i \text{ to level } i-1] = \alpha(1-(1-q_i)^{C_2 \log t_i})$$

and by a proper choice of C_2 we can get it as close to α as we would like. α itself is defined by C_1, and we can get it as close as we like to 1 by choosing a sufficiently large C_1.

This completes the proof for the probability of going all the way to level i_0, and we still need to show that it takes only $n(\log n)^{\frac{1}{2}+\epsilon_1}$ to get to level $\log_2 \log_2(n(\log n)^{\frac{1}{2}-\epsilon_1})$. We look at the probability that a set chosen in the top level will lead us to the next. First we estimate the probability, conditioned on one point being visited in its query-window, that any other point in visited in its query-window. This is done in the same way we estimated the probability for such a thing in 'Grid Search', and we get the probability $O((\log n)^{-2\epsilon_1})$. The probability for a single point, uniformly chosen on the torus, to be visited in its query-window is $O(\frac{1}{n(\log n)^{\frac{1}{2}+\epsilon_1}})$. If we mark A_j the event of the j'th point visited in it's query window we get:

$$\mathbf{P}[\bigcup A_j] \leq n(\log n)^{\frac{1}{2}-\epsilon_1}\mathbf{P}[A_1](1-\mathbf{P}[\bigcup_{j\neq 1} A_j|A_1]) = O((\log n)^{-2\epsilon_1})$$

. The expected time to get to the next level is therefore $n(\log n)^{\frac{1}{2}+\epsilon_1}$ □

2.3 the 3-dimensional case

In this section, $d = 3$. The result is very similar to the previous section.

Proposition 2.3. *For any A hunting algorithm* $\mathbf{E}[T_A] = \Omega(n^{\frac{3}{2}})$.

Proof. The proof is a simple counting argument. Define T, $T = min(t|\exists i \leq t, v_t = Z_i)$. Clearly $T \leq T_A$, so we will show $\mathbf{E}[T] = \Omega(n^{\frac{3}{2}})$, by showing that for sufficiently large n, $\mathbf{P}[T > \sqrt{\frac{n^3}{2}}] > \frac{1}{2}$. To estimate $\mathbf{P}[T > k]$ we define v'_i to be the queries of A with the oracle that returns 0 on all invocations. Note that for $i \leq T$, $v_i = v'_i$. Therefore $\mathbf{P}[T > k] = \mathbf{P}[T' > k]$, where T' is defined exactly as T with v_t replaced by v'_t. Now

$$\mathbf{P}[T' > k] > \mathbf{P}[\{v'_i\}_{i \leq k} \cap \{Z_i\}_{i \leq k} = \phi],$$

and since both sets are chosen independently, and $\{Z_i\}_{i \leq k}$ is translation invariant we get

$$\mathbf{P}[\{v'_i\}_{i \leq k} \cap \{Z_i\}_{i \leq k} = \phi] = 1 - \frac{\mathbf{E}[\#\{v'_i\}_{i \leq k}] \cdot \mathbf{E}[\#\{Z_i\}_{i \leq k}]}{n^3}.$$

Now, $\#\{v'_i\}_{i \leq k} = k$ and $\mathbf{E}[\#\{Z_i\}_{i \leq k}] \leq k + 1$, so we get

$$\mathbf{P}[T > \sqrt{\frac{n^3}{2}}] > \frac{1}{2}.$$

□

Now let us describe a hunting algorithm. We will define the algorithm in terms a constant i_0, then show that it can be chosen to achieve a desired hunting time. We will describe it using two internal variables, the level i and vertex v, they will always satisfy that the walk was at v at most $2t_i$ before the present, where t_i is 2^{2^i}, or $i = \log_2 \log_2(n^3)$, it should be obvious from the definition that this hold. First we will define the 'Grid Search' operation, we generate $\{u_j\}_{j < t_i^{\frac{3}{4}}}$, a square grid centred in v with distances $t_i^{\frac{1}{4}}$ between adjacent grid points and $t_i^{\frac{1}{4}}$ points along each cardinal direction. The whole grid is of side length $\sqrt{t_i}$. We would like to test all nodes of the grid for a point the walk was in $\sqrt{t_i}$ before it was tested. Finding such a point is simply a matter of splitting the grid to sets of size $\sqrt{t_i}$, and checking each one once (ignoring the information, this is just 'resetting' the points) than checking it again. The number of points in each set is equal to the time window we are looking at, as it is the temporal distance between two queries for any point. If we find any such point (visited up to $\sqrt{t_i}$ before its second query) we will move to the $(i-1)$'th level, setting v as this point. If none was found, the 'Grid Search' failed. The algorithm on the i'th level (for $i < \log_2 \log_2(n^3)$) is simply invoking 'Grid Search' $t_i^{\frac{1}{4}}$ times or until successful (on success, moving to the $(i-1)$'th level as noted above). If all the grid searches failed, move to level $\log_2 \log_2(n^3)$ around the origin. The algorithm starts on the top level, $\log_2 \log_2(n^3)$. The behaviour on the top level is different, looking for a point visited $n^{\frac{3}{2}}$ time ago in the whole torus. This is done by choosing a random uniform set of size $n^{\frac{3}{2}}(\log n)^{-3}$ under the constraint that no two points in the set are closer than $\frac{1}{10}\sqrt{n} \log n$, and checking it in the same manner (resetting it, waiting some time, then checking). i_0 is the minimal level. Once we reach this level, we make a query to a random uniform vertex within distance t_{i_0} from v then return to the top level. (Note: the algorithm have no stopping condition, but we are only interested in the hunting time)

Proposition 2.4. *There are d, i_0 such that the hunting algorithm A_{i_0} described above has $\mathbf{E}[T_{A_{i_0}}] = O(n^{\frac{3}{2}}(\log n)^d)$.*

Proof. We will get an upper bound on $T_{A_{i_0}}$ by estimating the time until a query done on level i_0 will hit the walk. Clearly, whatever i_0 is, hitting the walk have a probability bounded away from 0 every time we get to level i_0, so we will have to calculate two things, the probability that the algorithm will get there starting from level $\log_2 \log_2(n^3) - 1$, and the time it takes to get to level $\log_2 \log_2(n^3) - 1$ from the top level (the time for all the levels below it together is $O(n^{\frac{3}{2}})$, lower order than the time spent on the top level). Our bound will then be the time on the top level divided by the probability to get to level i_0. i_0 itself will be chosen to have behaviour close to the asymptotic bounds of the SRW.

we will show that on any level $i \geq i_0$ the probability of moving to a lower level is bounded away from 0 by some constant c, conditioned on the entire history of the process. Given this, the probability of moving all the way from level $\log_2 \log_2(n^3) - 1$ to i_0 will be the product of those probabilities, and is $\Omega((\log n)^(3 - d))$ (this is our definition of d). We will also show that the total time every such attempt takes is $O(n^{\frac{3}{2}}(\log n)^3)$, and so we get the required total expected hunting time.

The main observation that will allow us to handle this conditioning on the entire history is that on entering level i we know the walk visited v at some time window, from $t - t_i$ to t. If we can show that the probability to move to the next level is more than c for every time in this window, we could use the markovian behaviour and ignore the history up to the visit to v. That is for every event A, if we define F_j for the event $Z_j = v$ and $\forall k, t - t_i \leq k < j \rightarrow Z_k \neq v$ (j is the first visit to v in the window) we get

$$\begin{aligned}\mathbf{P}[A|\text{entire history}] &= \sum_{j=t-t_i}^{t} \mathbf{P}[F_j|\text{entire history}] \cdot \mathbf{P}[A|F_j \cap \text{entire history}] \\ &= \sum_{j=t-t_i}^{t} \mathbf{P}[F_j|\text{entire history}] \cdot \mathbf{P}[A|Z_j = v \cap \text{history after time } j]\end{aligned}$$

and since $\sum_{j=t-t_i}^{t} \mathbf{P}[F_j|\text{entire history}] = 1$, we get

$$\mathbf{P}[A|\text{entire history}] \geq \min_{j \in [t-t_i, t]} \mathbf{P}[A|Z_j = v \cap \text{history after time } j].$$

There is also a corresponding upper bound using the maximum, of course. We will also use a similar argument using a space-window. That is conditioning on the walk to be in some space at a given point in time, and using the same argument we get that bounds that can be shown for every point in this space-window and without any history before will hold when conditioning on the space-window with any history.

So now let us start in a 'Grid Search' with a specific vertex u of the grid and an upper bound on the probability of the walk visiting any vertex in the grid on its query-window, conditioned on visiting u in its query-window (conditioning on the history here is irrelevant, we could condition on the walk being somewhere in the torus at the beginning of the 'Grid Search' to justify ignoring the history). We will use this estimate to show that the conditioning on history after time j mentioned above is negligible. To estimate we will first use an upper bound for two specific vertices, visiting u_2, conditioned on visiting u_1. We bound this simply by the number of points the walk visits in the query-window of u_2, divided by the number of points closer to u_1 than u_2. Using l for the distance between u_1 and u_2 in grid

units we get $O(t_i^{\frac{-1}{4}} l^{-3})$. Summing over all possible u_2 gives us $O(\frac{\log t_i}{t_i^{\frac{1}{4}}})$. Now, when entering level i, the only queries after the start of the time-window are those done in the same 'Grid Search'. All points tested before v were not visited in their query-window, otherwise v would not have been tested, but the estimate above tells us that the conditioning on this will only change probabilities by $O(\frac{\log t_i}{t_i^{\frac{1}{4}}})$, which is negligible compared to a constant. From now on we will therefore ignore the conditioning on the history, to get that the advancement probability is indeed greater than some c.

Let us now look at the probability for success in one of the iterations of 'Grid Search', conditioned on the walk being in an arbitrary point v' in a box of side length $\sqrt{t_i}$ centered in v (a box that matches the grid), at the beginning of the 'Grid Search'. There are $t_i^{\frac{3}{8}}$ grid points within distance $\leq t_i^{\frac{3}{8}}$ of v', and the probability of the walk visiting a specific one of them in its query-window is $\Omega(t_i^{\frac{-5}{8}})$, simply because the window is of size $\sqrt{t_i}$ and starts $O(t_i^{\frac{3}{4}})$ after the walk was in v'. We already know that the probability of hitting another query-window conditioned on hitting one is low, so if A_j are the events of hitting any of those points' query-window we get

$$\mathbf{P}[\bigcup A_j] \geq t_i^{\frac{3}{8}} \mathbf{P}[A_1](1 - \mathbf{P}[\bigcup_{j \neq 1} A_j | A_1]) = O(t_i^{\frac{-1}{4}}),$$

call this q_i.

To estimate the probability of successfully moving from level i to level $i-1$ we need to do one last thing. Since we perform multiple 'Grid Search' attempts (if not successful), we will estimate the probability of a 'Grid Search' operation to succeed, conditioned on the failure of the previous ones. This we can bound by the probability for success conditioned on being in a box of side length $t_i^{\frac{1}{4}}$ centered in v at the beginning of the 'Grid Search' (this conditioning 'masks' the conditioning on previous failures), times the probability of being there conditioned on the failure of previous iterations. The conditioned probability of being in the box is easily bounded by the unconditioned probability of being in the box at the correct time minus the probability of succeeding in at least one of the previous iterations. The unconditioned probability of being in the box can be bounded for all iterations, call this bound α. So if we call the probability to succeed in at least one of the first j iterations x_j we get the following form:$x_{j+1} \geq x_j + q_i(\alpha - x_j)$. Using $y_j := \alpha - x_j$ we get $y_{j+1} \leq y_j(1-q_i)$, and $y_j \leq \alpha(1-q_i)^j$. so we get

$$\mathbf{P}[\text{successfully moving from level } i \text{ to level } i-1] = \alpha(1 - (1-q_i)^{t_i^{\frac{1}{4}}}).$$

This is clearly bounded away from zero. We can't control the constant here in the same way we did in 2 dimensions, because multiplying by a constant will change α, as the time spent in level i is of the same order as the time window the actions in level i is based on.

This completes the proof for the probability of going all the way to level i_0, and we still need to show that it takes only $O(n^{\frac{3}{2}})$ to get to level $\log_2 \log_2(n^3) - 1$. We look at the probability that a set chosen in the top level will lead us to the next. First we estimate the probability, conditioned on one point being visited in its query-window, that any other point in visited in its query-window. This is done in the same way we estimated the probability for

such a thing in 'Grid Search', and we get the probability $O((\log n)^{-2})$. The probability for a single point, uniformly chosen on the torus, to be visited in its query-window is $O(n^{-\frac{3}{2}})$. If we mark A_j the event of the j'th point visited in it's query window we get:

$$\mathbf{P}[\bigcup A_j] \leq n^{\frac{3}{2}}(\log n)^{-3}\mathbf{P}[A_1](1 - \mathbf{P}[\bigcup_{j\neq 1} A_j|A_1]) = O(1)$$

. The expected time to get to the next level is therefore $n^{\frac{3}{2}}(\log n)^3$ □

2.3.1 Open Problems

First of all, there is still a gap between the solutions presented and the lower bounds, and a natural question is for the optimal solution. Then there are related problems. One might consider other notions of information, we would describe the simplest two. First, full information. due to the markovian nature of the walk, this is equivalent to just getting the last time the walk visited this vertex. In this case we expect the optimal solution to achieve the lower bounds on $2, 3$ dimensional torii. Second, the existence of visits to this vertex. In this case the walk ceases to generate information at some point, we suspect that any algorithm will have a bounded probability to catch the walk only in $O(n^d)$ time. Another question, maybe even more natural one, is the behaviour on other graphs. On a torus of dimension 5 or more, we expect no algorithm to do better (up to a constant ratio) than independent uniform selection, even with full information.