Optimal Path Search in Small Worlds: Dimension Matters

[Extended Abstract] *

George Giakkoupis†
LIAFA – Université Paris Diderot, Paris, France
ggiak@liafa.jussieu.fr

Nicolas Schabanel‡
CNRS – Université Paris Diderot, Paris, France
IXXI – Université de Lyon, Lyon, France
http://www.liafa.jussieu.fr/∼nschaban

ABSTRACT

We consider Kleinberg's celebrated small-world model (2000). This model is based on a d-dimensional grid graph of n nodes, augmented by a constant number of "long-range links" per node. It is known that this graph has diameter $\Theta(\log n)$, and that a simple greedy search algorithm visits an expected number of $O(\log^2 n)$ nodes, which is asymptotically optimal over all decentralized search algorithms. Besides the number of nodes visited, a relevant measure is the length of the path constructed by the search algorithm. A decentralized algorithm by Lebhar and Schabanel (2003) constructs paths of expected length $O(\log n(\log \log n)^2)$ by visiting the same number of nodes as greedy search. A natural question, posed by Kleinberg (2006), is whether there are decentralized algorithms that construct paths of length $O(\log n)$ while visiting only a poly-logarithmic number of nodes.

In this paper we resolve this question. For grid dimension $d = 1$, we answer the question in the negative, by showing that any decentralized algorithm that visits a poly-logarithmic number of nodes constructs paths of expected length $\Omega(\log n \log \log n)$. Further we show that this bound is tight; a simple variant of the algorithm by Lebhar and Schabanel matches this bound. For dimension $d \geqslant 2$, however, we answer the question in the affirmative; the bound is achieved by essentially the same algorithm we used for $d = 1$. This is the first time that such a dichotomy, based on the dimension d, has been observed for an aspect of this model. Our results may be applicable to the design of peer-to-peer networks, where the length of the path along which data are transferred is critical for the network's performance.

†Supported by the ANR project PROSE, the European project EULER, and the INRIA project GANG.
‡Supported by ANR project DYNAMO, CNRS PEPS DynaDraw, and LIAFA's CNRS recurrent funding.
*A full version of this paper is available at `http://www.liafa.jussieu.fr/∼nschaban/`.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Routing and layout; E.1 [**Data Structures**]: Graphs and networks; G.2.2 [**Graph Theory**]: Graph algorithms, Network problems; C.2.2 [**Network Protocols**]: Routing protocols

General Terms

Algorithms, Performance, Theory

Keywords

small worlds, social networks, peer-to-peer networks, decentralized search, optimal paths

1. INTRODUCTION

The famous "six-degrees-of-separation" experiments conducted in the 1960s by Milgram [16] revealed that not only individuals are a few handshakes away from each other, but they are also able to find such short paths between them, in spite of their extremely local view of the world-wide social network. In 2000, Kleinberg [8] proposed a simple random network model exhibiting this surprising *small-world* property of social networks. Besides the sociological aspect, this model had an important impact on the design on several peer-to-peer protocols [12, 1, 19, 17, 18], because it addresses the general question of how decentralized algorithms can find short paths in a partially unknown network. Kleinberg's small-world model consists of an augmented d-dimensional grid with n^d nodes: besides its $2d$ *local* neighbors in the grid (representing its *local* acquaintances, such as geographic or professional), each node is given one *long-range* directed link pointing to a random node at distance r from it chosen with probability proportional to $1/r^s$, where s is a parameter of the model (this long-range contact represents, e.g., a random acquaintance met in the past). Kleinberg defined a *decentralized search algorithm* as an algorithm that has to deliver a message from a given source node to a given target without knowing in advance the long-range contacts of the nodes it has not visited yet. He showed that if $s \neq d$ then no decentralized algorithm can find *short* paths (i.e., of length poly-logarithmic in the size of the grid), even if the diameter of the augmented graph is short (as in the cases $s < 2d$ [14, 15]). Only when $d = s$, may a decentralized algorithm find short paths for arbitrary source–target pairs. In fact, the simple greedy algorithm that just forwards the message to the neighbor (local or long-range) of the current message holder that is the closest to the target node in the grid, computes paths of expected length $O(\log^2 n)$ [8]. Several decentralized algorithms [11, 5, 3, 13] have been proposed to compute shorter paths when $s = d$, *efficiently* (i.e., by visiting no more than a poly-logarithmic number of nodes). The best algorithm so far ([11]) for Kleinberg's original model computes path of length $O(\log n(\log\log n)^2)$ for arbitrary source–target pairs, which is still significantly larger than the diameter of the graph that is $\Theta(\log n)$. The question whether there exists efficient decentralized algorithms that can find optimal paths was highlighted by Kleinberg in 2006 (open problem n°3 in [9]).

We answer this question: first, by showing that if $d = 1$ no efficient decentralized algorithm can find paths shorter than $\Omega(\log n \log\log n)$ in expectation; and second, by providing an efficient decentralized algorithm (largely based on the work of [11]) which computes paths of expected length $O(\log n \log\log n)$ for $d = 1$, and $O(\log n)$ for $d \geqslant 2$. To our knowledge, this is the first time that such a transition in performance is observed in Kleinberg's model when $s = d$.

Path length is a critical indicator of performance in routing protocols (e.g., with respect to speed and fault tolerance). Several peer-to-peer networks (e.g., Chord [17] and Symphony [12]) are based on an augmented ring ($d = 1$). Our paper shows that using a 2-dimensional torus instead may improve considerably the performances while paying only a modest overhead cost (the degree of each node remains constant: 5 for $d = 2$).

A relevant question is if other networks (such as the one in [6]) would present similar transitions where, although the diameter is small and short paths can be computed by efficient decentralized algorithms, diameter may not be obtained by any such algorithm.

2. DEFINITIONS AND MAIN RESULTS

The network.

We consider Kleinberg's d-dimensional small-world network $\mathcal{K}_n^d$ with one long-range link per node [8, 2, 4].[1] This network consists of the d-dimensional toric lattice $\mathcal{L}_n^d = \{-n, \ldots, 0, \ldots, n\}^d$ with $N = (2n+1)^d$ nodes, where each node u has links to its $2d$ neighbors in the lattice (its *local contacts*), and one extra directed link to a random node $v \neq u$ (its *long-range contact*). Node v is chosen independently with probability $1/(Z_d \cdot \beta_{||u-v||})$, where $||u - v||$ denotes the (ℓ_1-)distance in $\mathcal{L}_n^d$ (i.e., $||u-v|| = \sum_{i=1}^d \min\{|u_i - v_i|, 2n+1-|u_i - v_i|\}$), β_r denotes the size of a ball of radius r in $\mathcal{L}_n^d$ (i.e., $\beta_r = |\{u : ||u|| \leqslant r\}|$), and $Z_d = \sum_{v \neq 0} 1/\beta_{||v||}$ is the normalizing constant. We denote by ℓ_u the long-range contact of u.

In all the following, we denote by 'ln' the natural logarithm (to the base e), and by 'log' the logarithm to the base 2. We denote by $H_n = \sum_{i=1}^n 1/i$ the harmonic sum; note that $\ln(n+1) < H_n \leqslant \ln n + 1$.

Efficient decentralized search algorithms.

We study algorithms that compute a path to transmit a message from a source node to a target node, along the local and (directed) long-range links of the network. Following Kleinberg's definition, such an algorithm is *decentralized* if it navigates through the network using only local information to compute the path. Precisely, it has knowledge of 1) the underlying lattice structure (i.e., the d-dimensional torus), 2) the coordinates of the target in the lattice, and 3) the nodes it has previously visited as well as their long-range contacts. But, crucially, 4) it can *only* visit nodes that are local or long-range contacts of previously visited nodes, and 5) it does not know the long-range contacts of any node that has not yet been visited. However, 6) the algorithm (but *not the path it computes*) is allowed to travel backwards along any directed links it has *already* followed. As remarked in [7], Point 6 is a crucial component of the human ability to find short paths: one can interpret it as a web user pushing the back button, or an individual in Milgram's experiment returning the letter to its previous holder (who wrote his address on the envelope before sending it). Further, we require that the algorithm be *efficient*, in the sense that the number of intermediate nodes it visits when computing a path between a pair of nodes should be at most poly-logarithmic in the size of the network with high probability. This is a natural requirement (at least) in the context of social networks, where the size of the network is typically in the order of millions or even billions. Efficient decentralized algorithms are thus likely to model the capacity of individuals to find short paths in a social network.

Since each node in $\mathcal{K}_n^d$ has a constant degree (equal to $2d+1$), the diameter of $\mathcal{K}_n^d$ is at least $\Omega(\log n)$. It was shown in [14] that the diameter of $\mathcal{K}_n^d$ is indeed $\Theta(\log n)$ for any fixed d. The analysis of [14] suggests a decentralized algorithm that would have to visit $\Omega(\sqrt{n})$ nodes to compute a path of length $O(\log n)$ for each pair of nodes, which is clearly unrealistic in the framework of social networks. In [8] it was shown that *greedy search* (which passes the message to the neighbor of the current message holder that is the closest to the target) computes paths of expected length $O(\log^2 n)$ while visiting $O(\log^2 n)$ nodes (since it does not visit any additional nodes other than those in the path it constructs). Then, in [13] it was shown that any decentralized search algorithm must visit at least $\Omega(\log^2 n)$ nodes in expectation to compute a path between a random pair of nodes. In [10, 11] an efficient decentralized algorithm was proposed that computes "near-optimal" paths of expected length $O(\log n(\log\log n)^2)$ between each pair of nodes, while visiting an optimal expected number of nodes $O(\log^2 n)$.

Our contribution.

In [9] Kleinberg posed the question of finding efficient decentralized algorithms that compute optimal paths of length $O(\log n)$ between any pair of nodes. The following theorem, which summarizes our main results, answers this question.

THEOREM 1.

(a) *For dimension $d = 1$ and any fixed $\varepsilon > 0$, there is an efficient decentralized search algorithm for $\mathcal{K}_n^d$ that computes a path of expected length $O(\log n \log\log n)$ for any source–target pair, while visiting $O(\log^{2+\varepsilon} n)$ nodes with high probability.*

(b) *For any fixed $d \geqslant 2$ and any fixed $\varepsilon > 0$, there is an efficient decentralized algorithm for $\mathcal{K}_n^d$ that computes a path of optimal expected length $O(\log n)$ for any source–target pair, while visiting $O(\log^{2+\varepsilon} n)$ nodes with high probability.*

[1]Note that our results are still valid for $O(1)$ long-range links per node instead of one.

(c) *For $d = 1$, for any efficient decentralized search algorithm for $\mathcal{K}_n^d$, the expected length of the path computed by the algorithm is $\Omega(\log n \log\log n)$ for almost all source-target pairs.*[2]
(Thus, (a) is asymptotically optimal.)

Intuition behind our results.

Our optimal decentralized search algorithms (Theorem 1(a) and 1(b)) rely on simplifying and improving the approach of [10, 11]. The key steps of the results in [10, 11] were obtained through rather complicated calculations. We obtain here a simple geometric rereading of these results that allows us to improve them and get optimal search algorithms. If we forget about the technical details, the algorithms in [10, 11] rely on the fact that in Kleinberg's network, the long-range contact of any node u has an equal probability $\approx \frac{1}{\log n}$ to be in each of the $\log n$ rings centered at u and whose distance from u ranges in $(2^{i-1}, 2^i]$, for each $i \in \{1, \ldots, \log n\}$. It follows that if u is at distance $r \in (2^{i-1}, 2^i]$ from the target t, then its long-range contact has probability $\approx \frac{i}{\log n} \approx \frac{\log r}{\log n}$ to be closer to t than u. If one neglects the possible overlapping (which is fine for large enough r), the BFS tree rooted at u expands at a rate of at least $\approx 1 + \frac{i}{\log n}$ towards the target (the 1 is because there is always at least one local contact which is closer to t, and the $\frac{i}{\log n}$ is the probability that the long-range contact of the current node is closer to t). It follows that after roughly $h = \frac{\log n \, \log\log n}{i}$ steps, the BFS tree contains at least $\approx (1 + \frac{i}{\log n})^{\frac{\log n \, \log\log n}{i}} \approx \log n$ leaves, among which one has a positive constant probability to fall in the ith ring (the one that contains t) and thus has a positive constant probability to be at distance at most $r/2$ from t. Routing the message from u to this leaf and repeating the process until t is reached yields a path of expected length at most $\approx \sum_{i=1}^{\log n} \frac{\log n \, \log\log n}{i} = \log n \cdot \log\log n \cdot H_{\log n} \approx \log n \, (\log\log n)^2$.

In order to obtain Theorem 1(a), we fix some $\varepsilon > 0$ and consider concentric rings of radius $\log^{\varepsilon j} n$, for $j \in \{1, \ldots, \log_{\log^\varepsilon n} n\}$, centered at the target. Suppose that the message has reached a node u at distance $r \in (\log^{\varepsilon(i-1)} n, \log^{\varepsilon i} n]$. As in [10, 11], we explore the BFS tree rooted at u but $(1+\varepsilon)$ times deeper, up to depth $h = \frac{\log n \, (1+\varepsilon) \, \log\log n}{\log r}$. This ensures that we get, with constant probability, at least $\approx (1 + \frac{\log r}{\log n})^{\frac{\log n \, (1+\varepsilon) \, \log\log n}{\log r}} \approx \log^{1+\varepsilon} n$ leaves, among which one has its long-range contact $\log^\varepsilon n$ times closer to the target with constant positive probability. A constant number of BFS are thus explored in each ring around the target on expectation. It follows that the expected length of the computed path between any pair of nodes is now at most:

$$\begin{aligned}
&\approx \sum_{i=1}^{\log_{\log^\varepsilon n} n} \frac{(1+\varepsilon) \, \log n \, \log\log n}{\log^{\varepsilon i} n} \\
&\lesssim \log n \cdot \log\log n \cdot \frac{1}{1 - \frac{1}{\log^\varepsilon n}} \\
&\approx \log n \, \log\log n.
\end{aligned}$$

[2]i.e., for a fraction $1 - o(1)$ of the source-target pairs.

The details of our algorithm (including how we handle the cases where overlapping cannot be neglected because we are close to the target) are presented in Section 3.3.

To obtain Theorem 1(b), we improve the analysis of the algorithm above by taking into account that the cardinal of a sphere grows at least linearly with its radius when $d \geqslant 2$. From our calculations, it follows that the expansion factor of the BFS tree rooted at a node at distance $r \in (\log^{\varepsilon(i-1)} n, \log^{\varepsilon i} n]$ is now at least $\approx 1 + \sqrt{\frac{\log r}{\log n}}$. Thus, we just need to explore each BFS tree up to depth $h = (1+\varepsilon) \, \log\log n \sqrt{\frac{\log n}{\log r}}$ to gather, with constant probability, $\log^{1+\varepsilon} n$ leaves among which one has its long-range contact at least $\log^\varepsilon n$ times closer to the target. It follows that the expected length of the computed path between any pair of nodes is at most

$$\begin{aligned}
&\approx (1+\varepsilon) \, \log\log n \sum_{i=1}^{\log_{\log^\varepsilon n} n} \sqrt{\frac{\log n}{\log\log^{\varepsilon i} n}} \\
&\approx \log\log n \sqrt{\log n} \sum_{i=1}^{\log_{\log^\varepsilon n} n} \frac{1}{\sqrt{\varepsilon i \log\log n}} \\
&\approx \frac{1}{\sqrt{\varepsilon}} \sqrt{\log\log n \log n} \sqrt{\log_{\log^\varepsilon n} n} \\
&= \frac{1}{\varepsilon} \log n.
\end{aligned}$$

It follows that this exploration-based efficient decentralized algorithm computes asymptotically optimal paths in expectation for any pair of nodes. The complete description and analysis of this algorithm are given in Section 3.4.

Our last result, Theorem 1(c), states that for the one-dimensional case, no efficient decentralized algorithm can achieve paths of diameter-length, and that, in fact, the upper bound of Theorem 1(a) is tight. Consider an efficient decentralized search algorithm that is bounded to visit at most $m = O(\log^c n)$ nodes, for some constant $c > 0$. Note that, with high enough probability, none of the long-range contacts of a set S of m nodes can be more than $O(\log^{c-1} n)$ times closer to the target than any node in S. It follows that the algorithm has to visit at least one node in each ring of radius m^i centered at the target. The first step consists in partitioning the underlying grid into concentric rings of radius m^{9i}, for $i \in \{1, \ldots, \log_{m^9} n\}$, centered at the target. According to the above, no matter how large the exponent c is, the algorithm will need to go through $\Omega(\log_{m^9} n) = \Omega(\log n / \log\log n)$ rings. The second step consists in proving by a coupling argument that the algorithm has almost no control over the first nodes it will visit each time it enters a ring. Indeed, we prove that since the algorithm visits at most m nodes in total, we can force the algorithm to enter each ring through a set of at most $O(m)$ nodes (the *entry points*), which are close to the farthest border of the ring from the target, and chosen randomly and *independently of the algorithm.* The third and last step consists then in bounding the extent of the BFS trees rooted at these entry points. Since the entry points are independent of the algorithm, we are left with a purely geometric analysis of these trees. This requires a finer analysis than for the search algorithm we proposed before, since we need to bound precisely the total cumulated length of the possibly explored long-range links. By bounding the cumulated length of the long-range links used in these BFS trees, we are able to prove that, with high enough probability, none of these BFS trees reaches the next ring, closer to the target, before depth $\gtrsim \frac{\log n \log\log n}{\log r}$, at least for the range of distances $r \in [2^{\log^{0.1} n}, 2^{\log^{0.9} n}]$. We conclude that the expected length of the computed path is at least:

$$\begin{aligned}
&\sum_{i=\log_{m^9}(2^{\log^{0.1} n})}^{\log_{m^9}(2^{\log^{0.9} n})} \frac{\log n \log\log n}{\log m^{9i}} \\
&\qquad \gtrsim \frac{\log n \log\log n}{9 \log m}\left(H_{\frac{\log^{0.9} n}{2\log m}} - H_{\frac{\log^{0.1} n}{2\log m}}\right) \\
&\qquad \gtrsim \frac{\log n \log\log n}{9c \log\log n} 0.8 \log\log n \\
&\qquad = \Omega(\log n \log\log n).
\end{aligned}$$

It follows that no efficient decentralized search algorithm can do asymptotically better than our algorithm in Theorem 1(a). The detailed proof is given in Section 4.

3. OPTIMAL SEARCH ALGORITHM

3.1 Exploration-based search

Our algorithm takes its inspiration in the work of [10, 11] with three important differences. First, as opposed to their work, each phase consists of a real BFS which allows to explore more nodes together with a shorter path length when $d \geqslant 2$. Second, at the end of each such BFS phase, with constant positive probability, the distance to the target is divided by some poly-logarithmic factor in n (instead of a constant factor). Third, our proofs are almost purely geometric (as opposed to direct calculations) which allows a finer understanding of the underlying principles.

Our exploration-based algorithm.

As in [10, 11], our algorithm is parameterized by three functions:

- $h_{\max}(r)$ is the maximum depth of the BFS performed from a node at distance r from the target,
- $b_{\max}$ is the maximum number of nodes that the BFS will have on one level, and
- *now-go-greedy* is the distance to the target from which we switch from exploration-based search to greedy search.

Algorithm 1 proceeds as follows (see Figure 1 and 2). As long as the current message holder u is at distance $r >$ *now-go-greedy*, it starts a BFS from u of maximum depth $h_{\max}(r)$ including only the long-range contacts which are at distance $< r$ from the target, and that are at least $2h_{\max}(r)$ away from any other previously seen long-range contact. The long-range contacts meeting these two conditions are stored in a set F allowing to perform this test easily. We denote by B_h the nodes visited at the h-th level of the BFS. The BFS is stoped prematurely if at some point $|B_h| > b_{\max}$, else it continues up to $h = h_{\max}(r)$. This guarantees that not too many nodes are visited. Once the BFS is completed, the message is forwarded along the links of the BFS to the node y which is the closest to the target among the contacts (long-range or local) of the nodes explored in the BFS. It is then forwarded again to a node z chosen $2h_{\max}(r)$ local steps closer to the target These extra steps guarantee that the next BFS will not overlap the previous ones (see Figure 1). Then, a new BFS is started from z if z is at distance $>$ *now-go-greedy* from the target. Once the message reaches a node at distance $\leqslant$ *now-go-greedy*, the algorithm switches to the "greedy mode" and routes the message using the improved greedy search algorithm of [5]: the current message holder looks at the $\log n$ closer nodes (which are within a radius of $\log^{1/d} n$ in $\mathcal{L}_n^d$) and routes the message to the closest node to the target among the contacts (local or long-range) of theses nodes; this algorithm computes a path of length $O(\log^{1/d} n \log r)$ between nodes at distance r from each other in $\mathcal{L}_n^d$ [5].

Christmas trees.

The nodes visited during each BFS during the exploration phase of the algorithm are structured as a tree of non-overlapping local balls of $\mathcal{L}_n^d$: its edges are the long-range links that are followed during the BFS, and its vertices are the balls of local links that grow from the extremities of the edges as the BFS goes deeper (see Figure 1). We will call such a structure a *Christmas tree*. Our main concern will be to evaluate the growth rate of a Christmas tree in $\mathcal{K}_n^d$.

3.2 Preliminary geometrical facts

We adopt a geometrical approach to the problem based on simple properties of the metric of the underlying grid. We denote by $B_u(r) = \{v : ||u - v|| \leqslant r\}$ the ball of radius r centered at u, and by $\beta_r = |B_u(r)|$ its cardinal. We have $\frac{2^d}{d!} r^d \leqslant \beta_r \leqslant \frac{2^d}{d!}(r+1)^d$. A useful fact is that grids have bounded doubling dimension: $\beta_{2r} \leqslant 2^d \beta_r$ for any integer r. It follows that: $Z_d \leqslant 2^d \log n$. We denote by ℓ_u the long-range contact of u. We can now bound the probability that a long-range link is followed during each exploration phase.

LEMMA 2. *Consider two integers f and h, a node u at distance $r > f^{2/d} \cdot h^2$ from the target t and a set of f nodes $F \subset B_t(r-1)$. Then:*

$$\Pr\{||\ell_u - t|| < r \text{ and } (\forall w \in F)\ ||\ell_u - w|| \geqslant h\} \geqslant \frac{1}{2^{5d+2}} \frac{\log r}{\log n}.$$

PROOF SKETCH. The geometrical intuition behind this result is based on Figure 3. Indeed, one can bound the probability to get closer to the target t by the sum of the probabilities of falling into one of the $\approx \log r$ disjoint balls with radius 2^{i-2} and centered at distance 2^i from u on a

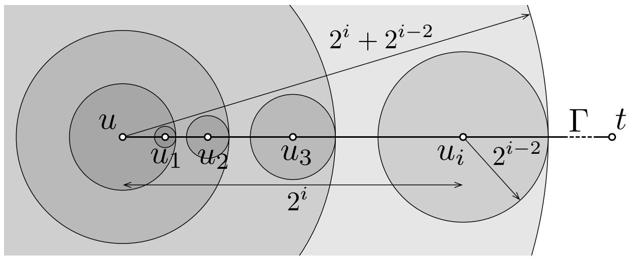

Figure 3: Bounding the probability of following a link.

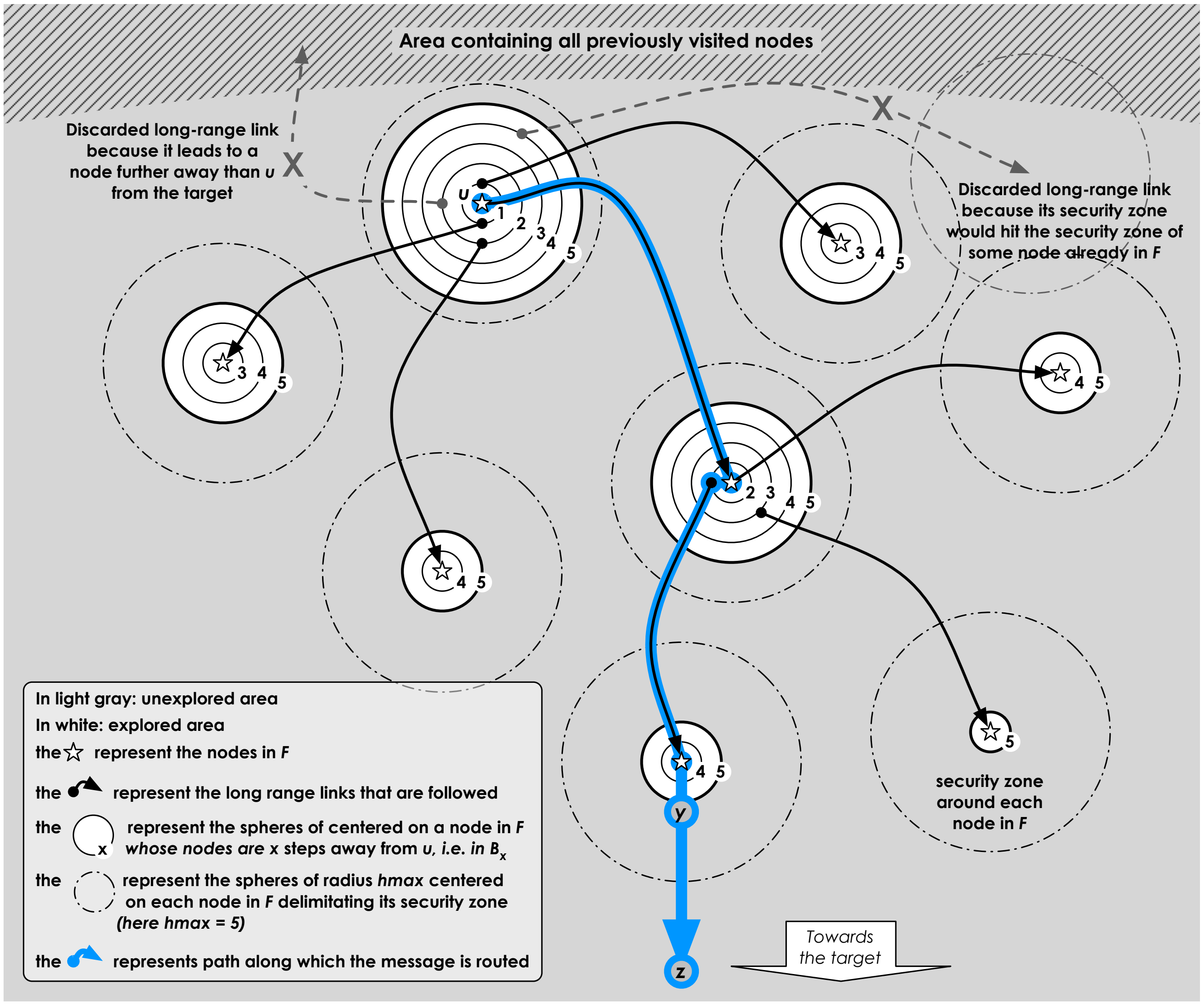
Area containing all previously visited nodes
Discarded long-range link because it leads to a node further away than u from the target
X
X
Discarded long-range link because its security zone would hit the security zone of some node already in F
u
1
2
3
4
5
security zone around each node in F
y
z
Towards the target
In light gray: unexplored area
In white: explored area
the ☆ represent the nodes in F
the represent the long range links that are followed
the represent the spheres of centered on a node in F whose nodes are x steps away from u, i.e. in B_x
the represent the spheres of radius hmax centered on each node in F delimitating its security zone (here hmax = 5)
the represents path along which the message is routed

Figure 1: The Christmas tree structure of the nodes visited in each exploration phase.

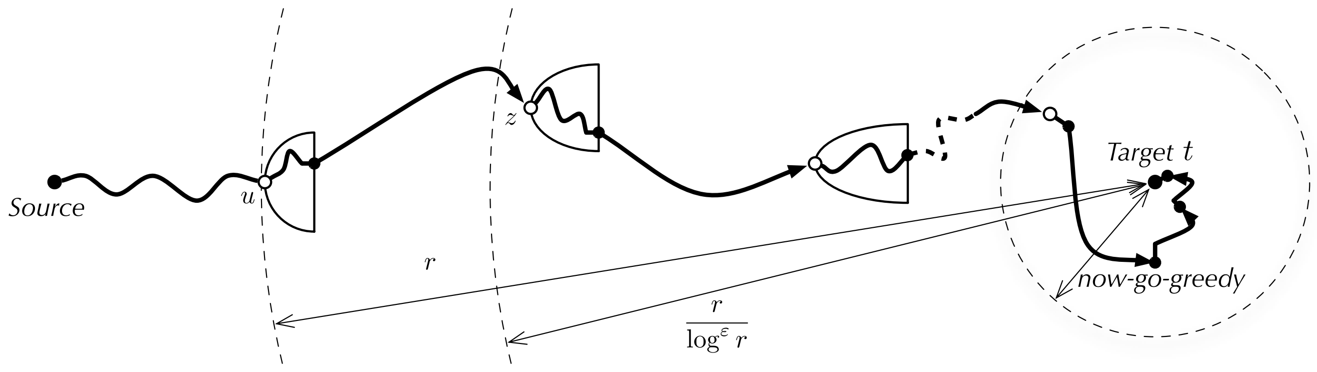

Figure 2: Progression of the algorithm: each exploration phase divides the distance to the target by $\log^{\varepsilon} r$ with constant probability as long as $r >$ *now-go-greedy*; as soon as $r \leqslant$ *now-go-greedy*, the routing switches to the improved greedy search algorithm of [5].

Algorithm 1 Exploration-based efficient decentralized search

```
input: a source $s$ and a target $t$ in $\mathcal{K}_n^d$.
parameters: $h_{\max}(r)$, $b_{\max}$, and now-go-greedy. // their values are given in Equations (3.1) for $d = 1$, and in (3.2) for $d \geqslant 2$ .
initialization: $u := s$.
1) Exploration
while $||u - t|| >$ now-go-greedy do
    1.a) BFS exploration
    Init sets $B_0$ and $C$ to $\{u\}$, and set $h := 1$, and $r := ||u - t||$.
    try
        while $h \leqslant h_{\max}(r)$ do
            Init set $B_h$ to an empty set of maximal size $b_{\max}$, throwing the exception is_full! if the size of $B_h$ reaches $b_{\max}$.
            for each $v \in B_{h-1}$ do
                Add all not yet visited local contacts of $v$ to $B_h$.
                if $||\ell_v - t|| < r$ and $(\forall w \in F)\, ||\ell_v - w|| \geqslant 2 \cdot h_{\max}(r)$ then
                    Add $\ell_v$ to $F$ and $B_h$.
            $h := h + 1$.
    catch is_full! $h := h + 1$ and goto line 14.          // exit the while if $|B_h| \geqslant b_{\max}$
    Let $h_{\text{stop}} := h - 1$ and $A := \bigcup_{h=0}^{h_{\text{stop}}} B_h$.          // By construction, $|B_{h_{stop}}| \leqslant b_{\max}$
    1.b) Message passing
    Let $y$ be the closest node to the target among the local and long-range contacts of the nodes in $B_{h_{\text{stop}}}$. Let $z$ be a node $2h_{\max}(r)$ local steps closer to the target than $y$. Route the message along the path from $u$ to $z$ passing through $y$ along the Christmas tree $A$. Set $u := z$.
2) Final steps (improved greedy search algorithm of [5])
Visit the $\log n$ closest local unexplored contacts of the current message holder and pass the message to the local or long-range contact of them which is the closest to the target until it reaches the target.
```

shortest path from u to t, for $i = 1, ..., \log r$. The doubling dimension property ensures that we have a uniform probability $\approx \frac{1}{\log n}$ to fall into each of these balls. We conclude that by showing that excluding a constant fraction of the balls is enough to subtract the probability of falling too close to F. □

A last useful lemma bounds the probability that one of the long-range contacts of a set of nodes is at least $\log^\varepsilon n$ closer to the target t. The proof is omitted.

Lemma 3. *Given a set B of at least $\frac{\log^{1+\varepsilon d} n}{2}$ nodes at distance at most r from the target in $\mathcal{L}_n^d$, the probability that at least one long-range contact of a node in B is a distance at most $\frac{r}{\log^\varepsilon n}$ from the target is at least $1 - e^{2^{-(2d+1)}} > 0$.*

3.3 Analysis of the search algorithm in dimension $d = 1$

Let us now fix the values of the parameters $h_{\max}(r)$, $b_{\max}$ and *now-go-greedy* for $d = 1$. For some $\varepsilon > 0$, we choose:

$$\begin{aligned} b_{\max} &= \log^{1+\varepsilon} n, \\ h_{\max}(r) &= \log_{1+\varrho_r}(b_{\max}), \\ \textit{now-go-greedy} &= \big((2h_{\max}(\log n) + 1) \cdot b_{\max}\big)^2 \cdot \log^\varepsilon n, \end{aligned} \tag{3.1}$$

where $\varrho_r = \frac{\log(r/\log^\varepsilon n)}{2^{5d+2}\log n} = \frac{\log r - \varepsilon \log\log n}{2^{5d+2}\log n}$. Note that:

$$h_{\max}(r) = \Theta\left(\frac{(1+\varepsilon)\ \log n\ \log\log n}{\log r}\right)$$

and *now-go-greedy* $= \Theta((1+\varepsilon)^2 \log^{2+3\varepsilon} n)$.

(Further, note that *now-go-greedy* $\geqslant \log n$ for all n).

Now, consider any exploration phase starting at a node u at distance $r >$ *now-go-greedy* from the target t. The following lemma is a direct corollary of Lemma 2.

Lemma 4. *For all h and $v \in B_h$, if $||v - t|| > \frac{r}{\log^\varepsilon n}$, then when ℓ_v is tested (Line 10 of Algorithm 1), we have:*

$$\Pr\big\{||v - t|| < r \text{ and } (\forall w \in F)\, ||v - w|| > h_{\max}(r)\big\} \geqslant \varrho_r.$$

It follows that the long-range contact of every node v in B_h, located at distance at least $r/\log^\varepsilon n$ from the target, is added to F independently at step h with probability at least ϱ_r.

We say that an exploration phase is a *success* if (i) a node at distance at most $\frac{r}{\log^\varepsilon n}$ from t is reached during the exploration, or (ii) $|B_{h_{\text{stop}}}| \geqslant \frac{b_{\max}}{2}$, i.e., if the last level of the Christmas tree contains at least $\frac{b_{\max}}{2}$ nodes (each of them having its long-range link still unobserved). We will prove the following lemma.

Lemma 5. *Every exploration phase starting at distance $r \geqslant$ now-go-greedy is a success with probability at least $\frac{1}{5}$.*

Proof Sketch. The proof follows mainly the steps of [11]. It simply explores a little bit deeper which allows to gather $\log^{1+\varepsilon} n$ nodes at the deepest level with constant probability. □

We can now conclude the analysis of Algorithm 1 for $d = 1$.

THEOREM 6. *For $d = 1$, Algorithm 1 computes for any source–target pair a path of expected length $O(\frac{1}{\varepsilon}\log n \log\log n)$ while visiting $O(\frac{1}{\varepsilon}\log^{2+2\varepsilon} n)$ nodes with high probability.*

PROOF SKETCH. Partition the grid into rings of radius $\log^{i\varepsilon} n$ centered at the target, for $i = \log_{\log^\varepsilon n}(\textit{now-go-greedy}) \dots \log_{\log^\varepsilon n}(n)$. Each exploration in the ith ring is a success with constant probability and if so, a contact in one of the next closer rings is found with constant probability. Each exploration in the i-th ring adds $O(h_{\max}(\log^{i\varepsilon} n)) = O(\frac{1}{i}\log n)$ nodes to the path. It follows that the computed path until distance *now-go-greedy* to the target has expected length $O(\log n \sum_{i \leqslant \log_{\log^\varepsilon n} n} \frac{1}{i}) = O(\log n \log\log n)$. The expected length of the second part of the path is $O(\log n \log(\textit{now-go-greedy})) = O(\log n \log\log n)$ by [8]. $\square$

3.4 Analysis of the search algorithm in dimension $d \geqslant 2$

In dimension $d \geqslant 2$, the size of the spheres grow at least linearly with their radius. It follows that the expansion rate of the Christmas tree is higher than in dimension 1, so we do not need to explore the Christmas tree as deep to find the $b_{\max}$ nodes necessary to decrease the distance to the target by some poly-logarithmic factor in n. In fact, we will show that the expansion rate of the Christmas tree is at least $(1 + \Theta(\sqrt{\varrho_r}))$ (instead of $(1 + \varrho_r)$ in dimension 1), which will allow us to a $\log\log n$ factor on the length of the path, obtaining an asymptotically optimal path.

Precisely, for $d \geqslant 2$ we choose:

$$\begin{aligned} b_{\max} &= \log^{1+\varepsilon d} n \\ h_{\max}(r) &= \log_\lambda(4 b_{\max}) \\ \textit{now-go-greedy} &= 2^{\log^{1-1/d} n} \end{aligned} \tag{3.2}$$

where $\varrho_r = \frac{\log r - \varepsilon \log\log n}{2^{5d+2}\log n}$ as before, and $\lambda = 1 + \frac{2}{\sqrt{1+\frac{4}{\varrho_r}}-1} = 1 + \Theta(\sqrt{\varrho_r})$. The enigmatic (for now) choice of λ comes from Lemma 8: λ is a natural lower bound on the expansion rate of the Christmas tree computed in each exploration phase.

Consider an exploration starting at a node u at distance $r > \textit{now-go-greedy}$ from the target. As in dimension 1, the following lemma is a direct corollary of Lemma 2.

LEMMA 7. *For all h and $v \in B_h$, if $||v - t|| > \frac{r}{\log^\varepsilon n}$, then when ℓ_v is tested (Line 10 of Algorithm 1), we have:*

$$\Pr\{||v - t|| < r \text{ and } (\forall w \in F)\, ||v - w|| > h_{\max}(r)\} \geqslant \varrho_r.$$

As in dimension 1, we say that the exploration phase *succeeds* if either (i) a node at distance at most $\frac{r}{\log^\varepsilon n}$ from t is found during the exploration, or (ii) $|B_{h_{\text{stop}}}| \geqslant \frac{b_{\max}}{2}$, i.e., if the last level of the Christmas tree contains at least $\frac{b_{\max}}{2}$ nodes (each having a still unobserved long-range link). The following lemma shows that each exploration succeeds with constant probability.

LEMMA 8. *Every exploration phase starting at distance $r \geqslant$ now-go-greedy succeeds with probability at least $\frac{1}{17}$.*

PROOF SKETCH. The proof relies on demonstrating that the number of nodes at distance $< r$ from the target which are at most h local or long-range contacts away from u grows exponentially at a rate λ^h with constant probability. The evaluation of this growth is complicated by the fact that the growth of the Christmas tree in dimension $d \geqslant 2$ is heterogenous: a polynomial component of the growth comes from the growth of the already present balls; and an exponential component comes from the discovery of new good long-range contacts that will creates new balls; and both polynomial and exponential growth are required to obtain the result. We thus use two random variables: one which counts the number of nodes h contacts away from u, while the other counts the current number of balls in the Christmas tree. We then solve the recursive system of equations that relate these two variables to show that their expected value indeed grows as λ^h and that their variance is just a constant factor times the square of their expected value, which ensures by the second moment method (see [11]) that they remain close to their expected value with constant probability. $\square$

We can now conclude the analysis of our algorithm for dimension $d \geqslant 2$, showing that it computes paths of optimal expected length $O(\log n)$, that is, at most asymptotically equal to the diameter of the graph $\mathcal{K}_n^d$.

THEOREM 9. *For dimension $d \geqslant 2$, Algorithm 1 computes, for all source-target pairs, a path of expected length $O(\frac{1}{\varepsilon}\log n)$ while visiting $O(\frac{1}{\varepsilon}\log^{2+O(\varepsilon)} n)$ nodes with high probability.*

PROOF SKETCH. As before, we partition the underlying grid into rings of radii $\log^{i\varepsilon} n$ centered at the target. The key observation is that now each exploration in the ith ring adds only $\log_\lambda(\log^{1+d\varepsilon} n) = O(\sqrt{\frac{1}{i}\log n \log\log n})$ nodes to the path. It follows that the expected length of the computed path during the exploration part is at most $O(\sqrt{\log n \log\log n}\sum_{i \leqslant \log_{\log^\varepsilon n} n}\frac{1}{\sqrt{i}}) = O(\log n)$. For the second part, we use the fact that the search algorithm of [5] computes a path of expected length at most $O(\log^{1/d} n \log(\textit{now-go-greedy})) = O(\log n)$. $\square$

4. LOWER BOUND FOR THE ONE-DIMENSIONAL CASE

We consider Kleinberg's ring-based graph $\mathcal{K}_n^1$ on $2n+1$ nodes $\{-n, \dots, n\}$ and an efficient decentralized search algorithm A that computes paths between any source-target pair while visiting at most $m = \log^{O(1)} n$ nodes — w.l.o.g. and for technical reasons, we assume that $m \geqslant \log^2 n$. The *distance* between two nodes u and v, denoted $||u - v|| \in \{0, \dots, n\}$, is the length of the shortest path along the ring between u and v. The long-range link ℓ_u of each node u is chosen independently at random to be $\ell_u = v\ (\neq u)$ with probability $p_{||u-v||}$, where $p_d = \frac{1}{2H_n}\cdot\frac{1}{d}$ and $H_n = \sum_{i=1}^n \frac{1}{i} = \ln n + O(1)$. We denote by $A(r_1, r_2] = B_t(r_2) \smallsetminus B_t(r_1)$ the annulus of radii $(r_1, r_2]$ around the target t.

4.1 The Lower Bound

Theorem 1(c) is a direct corollary of the following result which will be proved in this section.

THEOREM 10. *For any efficient decentralized search algorithm A, there exists two constants $p_A > 0$ and $c_A > 0$ (independent of n), such that for all large enough n, for all* *source–target pairs (s,t) in $\mathcal{K}_n^1$ with $|s-t| \geqslant 2^{(\log n)^{0.9}} = o(n)$, the length of the computed path from s to t is at least $c_A \cdot \log n \log\log n$ with probability at least p_A.*

This theorem follows from the following lemma which reduces the analysis to the study of the length of the computed path restricted to each annulus of radii $(\frac{d}{m^9}, d]$ around the target for $2^{\log^{0.1} n} \leqslant d \leqslant 2^{\log^{0.9} n}$.

LEMMA 11. *Consider an arbitrary decentralized algorithm that visits at most $m = (\log n)^{O(1)}$ nodes (w.l.o.g. $m \geqslant \log^2 n$). There exists a constant $c > 0$ such that: for all distances d with $2^{\log^{0.1} n} \leqslant d \leqslant 2^{\log^{0.9} n}$, and all source–target pairs (s,t) with $||s-t|| \geqslant d$, the number K of nodes in the path output by the algorithm which lie inside the annulus $A(\frac{d}{m^9}, d]$ satisfies*

$$\Pr\left\{K \geqslant c \cdot \frac{\log n \cdot \log m}{\log d}\right\} = 1 - O\left(\frac{1}{\log n}\right).$$

PROOF OF THEOREM 10. Consider some source-target pair (s,t) such that $||s-t|| \geqslant 2^{\log^{0.9} n}$. Let $d_i = \frac{2^{\log^{0.9} n}}{m^{9i}}$, for $i = 0, \ldots, \log^{0.9}_{m^9} n - \log^{0.1}_{m^9} n$. According to Lemma 11, the intersection of the path with each annulus $A(d_{i+1}, d_i]$ counts at least $k_i = c \cdot \frac{\log n \cdot \log m}{\log d_i}$ nodes with probability $1 - O(\frac{1}{\log n})$. It follows by the union bound that with probability at least $1 - O(\frac{\log^{0.9} n - \log^{0.1} n}{9 \log m} \cdot \frac{1}{\log n}) = 1 - o(1)$, the length of this path is at least:

$$\begin{aligned}
\sum_{i=1}^{\log^{0.9}_{m^9} n - \log^{0.1}_{m^9} n} k_i &= \sum_{i=1}^{\log^{0.9}_{m^9} n - \log^{0.1}_{m^9} n} \frac{c \cdot \log n \log m}{\log^{0.9} n - 9i \cdot \log m} \\
&= \frac{c}{9} \cdot \log n \sum_{i=1}^{\log^{0.9}_{m^9} n - \log^{0.1}_{m^9} n} \frac{1}{\frac{\log^{0.9} n}{9 \log m} - i} \\
&= \frac{c}{9} \cdot \log n \cdot \left(\ln \frac{\log^{0.9} n}{9 \log m} - \ln \frac{\log^{0.1} n}{9 \log m} + O(1) \right) \\
&= \frac{c}{9} \cdot \log n \cdot (0.8 \ln\log n + O(1))
\end{aligned}$$

It follows that the expect length of the path computed from s to t is at least $(1 - o(1)) \cdot \frac{0.8 c \ln 2}{9} \log n \log\log n$. $\square$

The following sections are devoted to the proof of Lemma 11. Let us now fix some source-target pair (s,t) such that $||s-t|| \geqslant d$ for some distance d with $2^{\log^{0.1} n} \leqslant d \leqslant 2^{\log^{0.9} n}$. W.l.o.g., $t = 0$ and from now on, we denote by $B(d) = B_0(d)$ the ball of radius d centered on the target (note that $||s-t|| = |s|$).

4.2 Getting rid of the algorithm: reduction to geometrical analysis

Consider the (random) set of nodes visited by the algorithm while computing a path from s to 0 with $|s| \geqslant 2^{\log^{0.9} n}$. We say that a node $u \in B(d)$ is an *entry point* in $B(d)$ if it is a (local or long-range) contact of a node not in $B(d)$ visited by the algorithm. The following lemma shows by a coupling argument that the set of entry points is included in a random set, of size at most $m+2$, which is independent of the algorithm.

LEMMA 12. *There exists a random set $R \subset B(d)$ of size at most $m+2$, which is independent of the algorithm and of the long-range links originating from $B(d)$, such that for every run of the algorithm, all of its entry points belong to R with probability 1.*

PROOF SKETCH. We use the principle of differed decision: the long-range links originating outside $B_t(d)$ in $\mathcal{K}_n^1$ are revealed only when their origin will be visited by the algorithm. In order to control their destination when they fall inside $B(d)$, we generate beforehand m random numbers $r_1, \ldots, r_m \in \{-n, \ldots, -1, 1, \ldots, n\}$ chosen independently with probability $\Pr\{r_i = r\} = p_{|r|}$. We set:

$$\begin{aligned}
R = &\{-d-1+r_i : 1 \leqslant r_i \leqslant 2d+1\} \\
&\cup \{d+1+r_i : -2d-1 \leqslant r_i \leqslant -1\} \cup \{-d, d\}
\end{aligned}$$

to be our random set of "authorized" entry points. Let us now show we can force the algorithm to enter $B(d)$ through these points only without corrupting the distribution of $\mathcal{K}_n^1$.

Let $u_1, \ldots, u_{m'}$ $(m' \leqslant m)$ be the sequence of the nodes visited by the algorithm ordered by time of first visit. We set the long-range contact ℓ_{u_i} of each u_i at the time it is visited as follows. If $u_i \in B(d)$, then set its long-range contact $\ell_{u_i} = v$ for some v drawn independently with probability $p_{||u-v||}$. Consider now the case where $u_i \notin B_t(d)$. We use the fact that the probability distribution of each long-range contact decreases with the distance and then use a reject-based strategy to select the long-range contact of u_i. Let us denote by $u \curvearrowright v$ and $v \curvearrowleft u$ in $\{0, \ldots, 2n\}$ the clockwise and counterclockwise distances from u to v. We extend the probability distribution p_r to $\{1, \ldots, 2n\}$ simply by setting $p_r = 0$ for $r > n$. Now, we have two cases: either r_i is positive or negative. If $r_i > 0$, then the long-range contact of u_i will be chosen clockwise, among $u_i + 1, \ldots, u_i + n$ as follows:

- if $1 \leqslant r_i \leqslant 2d+1$, let $\lambda_i = -d-1+r_i \in B(d)$. Note that since $r_i = (-d-1) \curvearrowright \lambda_i \leqslant u_i \curvearrowright \lambda_i$, we have $p_{r_i} \geqslant p_{(u_i \curvearrowright \lambda_i)}$. We then choose $\ell_{u_i} := \lambda_i$ with probability $\frac{p_{(u_i \curvearrowright \lambda_i)}}{p_{r_i}}$; otherwise we reject λ_i and choose $\ell_{u_i} := u_i + k$ with $k \in \{1, \ldots, n\} \smallsetminus \{u_i \curvearrowright -d, \ldots, u_i \curvearrowright d\}$ with probability $p'_k = \frac{p_k}{\frac{1}{2} - \sum_{v \in B(d)} p_{(u_i \curvearrowright v)}}$. ($p'_k$ is the conditional probability that the long-range contact of u_i is at distance k given that it is chosen clockwise and outside the ball $B(d)$)

- otherwise, i.e. if $r_i > 2d+1$, choose $\ell_{u_i} = u_i + k$ with $k \in \{1, \ldots, n\} \smallsetminus \{u_i \curvearrowright -d, \ldots, u_i \curvearrowright d\}$ chosen according to distribution p'_k.

The case $r_i < 0$ is obtained by clockwise/counterclockwise symmetry and is omitted. One can easily check that the distribution of the long-range links of any visited node u_i is identical to the one in $\mathcal{K}_n^1$ and ensures that the long-range contact of any node visited by the algorithm outside $B(d)$ either lies outside $B(d)$ or belongs to R, a random set of at most $m+2$ nodes which is independent of the algorithm. $\square$

The following lemma shows that we can furthermore assume that the entry points are fairly close to the boundary of $B(d)$.

LEMMA 13. $\Pr\{R \subset A(\frac{d}{m}, d]\} = 1 - O(\frac{1}{\log n})$.

PROOF SKETCH. $\Pr\{R \subset A(\frac{d}{m}, d]\}$

$$= \left(1 - \Pr\{d(1-\tfrac{1}{m}) \leqslant |r_i| \leqslant d(1+\tfrac{1}{m})\}\right)^m$$
$$\sim \exp(m \cdot \ln(1 - \tfrac{2}{m \ln n})) = 1 - O(\tfrac{1}{\log n}).$$

□

We now want to bound from bellow the number K of nodes of the path from s to $t = 0$ computed by the algorithm inside the annulus $A(\frac{d}{m^9}, d]$. We know from above that this path necessarily enters $B(d)$ through nodes in the random set R which is independent of the algorithm and of the long-range links originating from$B(d)$. For any set of $m+2$ nodes $S \subset A(\frac{m}{d}, d]$, let us denote by $D(S)$ the minimum depth of the BFS forest *restricted to* $B(d)$ which starts from the nodes in S and reaches some node in $B(\frac{d}{m})$. Clearly,

LEMMA 14. *If there exists some k and q such that for all sets $S \subset A(\frac{m}{d}, d]$ with $|S| = m+2$, $\Pr\{D(S) \geqslant k\} \geqslant q$, then:*

$$\Pr\{K \geqslant k\} \geqslant q \cdot (1 - O(\tfrac{1}{\log n})).$$

PROOF SKETCH. With probability $1 - O(\frac{1}{\log n})$, $R \subset A(\frac{m}{d}, d]$, independently of the long-range links originating from $B(d)$. But since with probability at least q (taken over the long-range links originating from $B(d)$ only and thus independent from R) for $S = R$ we have $D(R) \geqslant k$, it follows that with probability at least $q \cdot (1 - O(\frac{1}{\log n}))$, $K \geqslant D(R) \geqslant k$. □

4.3 Geometrical analysis

We are now left with answering a purely geometrical question about $\mathcal{K}_n^1$: provide a lower bound on the minimum depth $D(S)$ of a BFS forest restricted to $B(d)$, starting from nodes in a set $S \subset A(\frac{d}{m}, d]$ of size $m + 2$ and reaching some node in $B(\frac{d}{m^9})$. We know from [8] that with high probability $D(S) = O(\log^2 n)$. Given that for our purposes $d \geqslant 2^{\log^{0.1} n}$, any progress made by local links is negligible. We will just ignore them and focus only on the progresses made by long-range links. In order to lower-bound $D(S)$ we will have to watch two parameters: (1) the frequency of the long-range links in the forest; and (2) their total length along each path of the forest. Lemma 15 handles (1) and Lemma 19 treats (2). We will also need to get rid of the dependencies between the paths inside the forest, which is achieved by a sequence of reductions of the original BFS forest from S, to a random abstract path embedded in $B(d)$.

Let us first bound the frequency of the long-range links in the BFS forest restricted inside $B(d)$. Clearly,

LEMMA 15. *For all $u \in B(d)$,*

$$\Pr\{\ell_u \in B(d)\} \leqslant \varrho =_{\text{def}} \tfrac{H_d}{H_n} = \tfrac{\log d}{\log n} + O(\tfrac{1}{\log n}).$$

Since none of the long-range links are allowed to exit $B(d)$ in the BFS, the long-range link of any node in the BFS belongs to the BFS with probability at most ϱ independently.

Reduction to the analysis of an abstract path embedded in $B(d)$.

Step 1. Each node in S and each long-range contact in the BFS forest gives birth to two chains of local neighbors. In order to get an homogeneous representation, we duplicate each of these nodes (Step 1 in Figure 4) so that every node in the BFS has exactly one local son and, with probability ϱ, two long-range sons, embedded at the same location (for now) in $B(d)$.

Step 2. A simple probabilistic domination argument shows that if one draws, with probability ϱ, for each node in the forest two long-range links pointing to two (possibly different) long-range contacts (instead of one) according to distribution p_r restricted to be in $B(d)$ (one for each copy of the original long-range contact), then the length of the overall path can only increase stochastically and thus the depth of the forest can only decrease stochastically. We call such a structure the (random) *embedded enhanced BFS forest F_S* of S (Step 2 in Figure 4): starting from the (duplicated) nodes in S at level 0, each node at level $i - 1$ has one local son at level i (embedded at the same location—recall we neglect the local steps in the analysis) plus, with probability ϱ, two long-range sons drawn independently at random according to the distribution p_r restricted to be inside $B(d)$. We denote by $D(F_S)$ the minimum length of a path in F_S reaching a node in $B(\frac{d}{m^9})$. Clearly,

LEMMA 16. *$D(S)$ dominates stochastically $D(F_S)$.*

Step 3. We would now like to study how close each path connecting a node in S to a node at level i in F_S can get to the target. The lengths of these paths are highly correlated because each pair of paths share the same long-range links in their common part. We then seek for a bound on how close each of these paths can get to the target with high enough probability so that we get a satisfying lower bound for all of them together. In order to do so, we need to evaluate: (1) the number of nodes at level i in F_S and (2) the frequency of the long-range links along each path. Obtaining a crude bound on the number of nodes at level i is easy (and similar to the analysis in Section 3.3):

LEMMA 17. *The number of nodes N_i at level i in F_S is at most $2(m + 2)(1 + 3\varrho)^i$ for all levels i in F_S with probability $1 - n^{-\omega(1)}$.*

Counterintuitively, because long-range links appears together in an embedded enhanced forest, the frequency of long-range links along each path in F_S is higher than ϱ, but still no larger than 3ϱ. This fact is established by studying how paths merge together from bottom to top.

LEMMA 18. *The number of long-range links in a path joining a node in S to a node at level i in F_S is stochastically dominated by* $\text{Binomial}(i, 3\varrho)$.

We have thus reduced the problem to evaluating how close a path of size i including $\text{Binomial}(i, 3\varrho)$ long-range links restricted to land in $B(d)$, starting from a location in $A(\frac{d}{m}, d]$ can get to $B(\frac{d}{m^9})$. This is done by the following lemma:

LEMMA 19. *The shortest distance L to the target from a node in an embedded path inside $B(d)$ starting from a location in $A(\frac{d}{m}, d]$ and containing at most b long-range links verifies:* $\Pr\{L \geqslant \frac{d}{3m^2 7^b \log n}\} \geqslant 1 - O(\frac{1}{m^2 \log n})$.

Taking $i = \frac{\ln m}{6\varrho}$, and using the Union Bound over all $O(m\sqrt{m})$ resulting paths w.h.p., yields the following corollary that concludes the proof of Lemma 11:

COROLLARY 20. $\Pr\{D(F_S) \geqslant \frac{\ln m}{6\varrho}\} \geqslant 1 - O(\frac{1}{\log n})$.

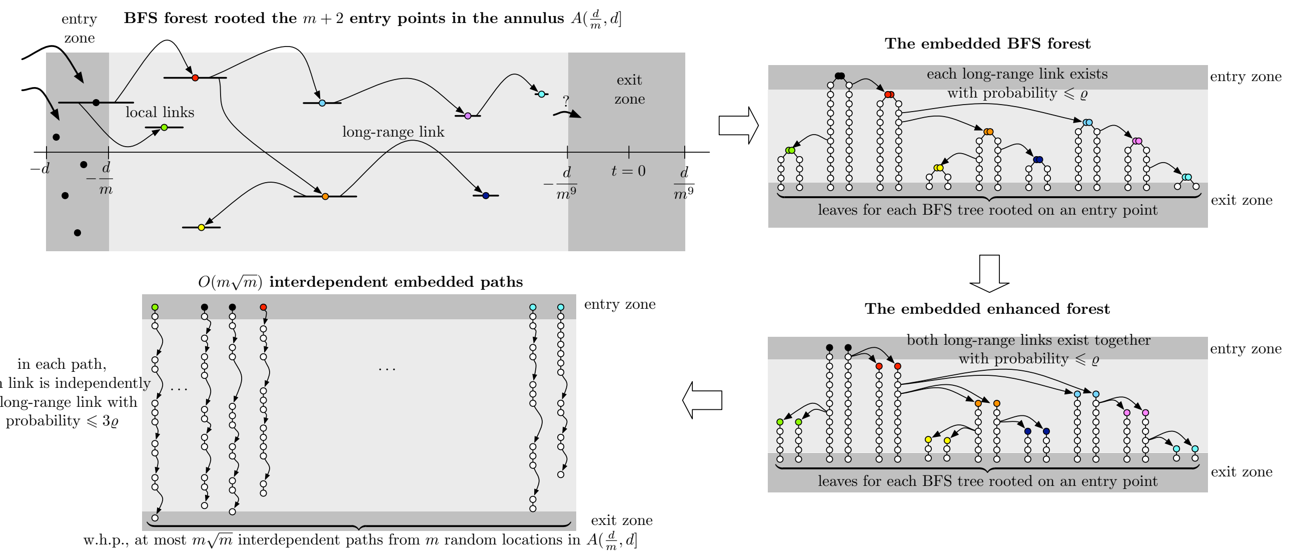
entry zone
BFS forest rooted the $m+2$ entry points in the annulus $A(\frac{d}{m}, d]$
local links
long-range link
exit zone
?
$-d$
$-\frac{d}{m}$
$-\frac{d}{m^9}$
$t=0$
$\frac{d}{m^9}$
The embedded BFS forest
each long-range link exists with probability $\leqslant \varrho$
entry zone
exit zone
leaves for each BFS tree rooted on an entry point
The embedded enhanced forest
both long-range links exist together with probability $\leqslant \varrho$
entry zone
exit zone
leaves for each BFS tree rooted on an entry point
$O(m\sqrt{m})$ interdependent embedded paths
entry zone
exit zone
in each path, each link is independently a long-range link with probability $\leqslant 3\varrho$
...
w.h.p., at most $m\sqrt{m}$ interdependent paths from m random locations in $A(\frac{d}{m}, d]$

Figure 4: Outline of the reduction to abstract paths embedded in $B(d)$.

5. REFERENCES

[1] J. Aspnes, Z. Diamadi, and G. Shah. Fault-tolerant routing in peer-to-peer systems. In *Proc. 21st ACM Symp. on Principles of Distributed Computing (PODC)*, pages 223–232, 2002.

[2] L. Barrière, P. Fraigniaud, E. Kranakis, and D. Krizanc. Efficient routing in networks with long range contacts. In *Proc.of the Int. Conf. on Distributed Computing (DISC)*, pages 270–284, 2001.

[3] D. Coppersmith, D. Gamarnik, and M. Sviridenko. The diameter of a long-range percolation graph. *Random Structures and Algorithms*, 21(1):1–13, 2002.

[4] P. Duchon, N. Hanusse, E. Lebhar, and N. Schabanel. Could any graph be turned into a small world? *Theoretical Computer Science, Special edition on complex networks*, 355(1):96–103, 2006.

[5] P. Fraigniaud, C. Gavoille, and C. Paul. Eclecticism shrinks even small worlds. In *Proc. of ACM Symp. on Principles of Distributed Computing (PODC)*, pages 169–178, 2004.

[6] P. Fraigniaud and G. Giakkoupis. On the searchability of small-world networks with arbitrary underlying structure. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 389–398, 2010.

[7] J. Kleinberg. Small-world phenomena and the dynamics of information. *Advances in Neural Information Processing Systems, MIT Press.*, 14, 2002.

[8] J. M. Kleinberg. The small-world phenomenon: an algorithmic perspective. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 163–170, 2000.

[9] J. M. Kleinberg. Complex networks and decentralized search algorithms. In *Proc. of the Int. Congress of Mathematicians (ICM)*, pages 1–26, 2006. Nevanlinna prize recipient presentation.

[10] E. Lebhar and N. Schabanel. Almost optimal decentralized routing in long-range contact networks. In *Proc. of the Int. Colloquium on Automata, Languages and Programming (ICALP)*, volume LNCS 3142, pages 894–905, July 2004.

[11] E. Lebhar and N. Schabanel. Close to optimal decentralized routing in long-range contact networks. *Theoretical Computer Science special issue on ICALP 2004*, 348(294-310), 2005.

[12] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *Proc. 4th USENIX Symp. on Internet Technologies and Systems (USITS)*, pages 127–140, 2003.

[13] G. S. Manku, M. Naor, and U. Wieder. Know thy neighbor's neighbor: the power of lookahead in randomized P2P networks. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, 2004.

[14] C. Martel and V. Nguyen. Analyzing kleinberg's (and other) small-world models. In *Proc. of ACM Symp. on Principles of Distributed Computing (PODC)*, pages 179–188, 2004.

[15] C. Martel and V. Nguyen. Analyzing and characterizing small-world graphs. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, pages 311–320, 2005.

[16] S. Milgram. The small world problem. *Psych. Today*, 2:60–67, 1967.

[17] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord : A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM Special Interest Group on Data Communications (SIGCOMM)*, pages 149–160, 2001.

[18] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Proc. of ACM Special Interest Group on Data Communications (SIGCOMM)*, pages 85–96, 2005.

[19] H. Zhang, A. Goel, and R. Govindan. Using the small-world model to improve Freenet performance. In *Proc. 21st IEEE INFOCOM*, pages 1228–1237, 2002.