A fully distributed scheme to run a network into a small world

Philippe Duchon, Nicolas Hanusse, Emmanuelle Lebhar, Nicolas Schabanel

▶ **To cite this version:**

Philippe Duchon, Nicolas Hanusse, Emmanuelle Lebhar, Nicolas Schabanel. A fully distributed scheme to run a network into a small world. [Research Report] LIP RR-2006-03, Laboratoire de l'informatique du parallélisme. 2006, 2+13p. hal-02102237

HAL Id: hal-02102237

https://hal-lara.archives-ouvertes.fr/hal-02102237

Submitted on 17 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

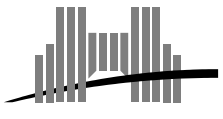
A fully distributed scheme to turn a network into a small world

Philippe Duchon
Nicolas Hanusse
Emmanuelle Lebhar
Nicolas Schabanel

Janvier 2006

Research Report N° 2006-03

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`



A fully distributed scheme to turn a network into a small world

Philippe Duchon
Nicolas Hanusse
Emmanuelle Lebhar
Nicolas Schabanel

Janvier 2006

Abstract

We investigate the problem of efficiently preprocessing a large network, in a fully distributed manner, so that the resulting network is a navigable small world. Namely, if the network has bounded growth, by adding a single entry in a distributed manner to each routing table (using $O(n)$ rounds and $O(\text{polylog}\, n)$ space), we obtain a network in which the greedy routing algorithm computes paths of polylogarithmic expected length between any pair of nodes. A bounded growth graph is a graph of constant expansion rate, *i.e.* the number of nodes within distance $2r$ from a given node is at most a constant times the number of nodes within distance r. These graphs are considered as a relevant framework for real networks as Peer-to-peer networks where our scheme could be used to considerably improve the routing speed over time. We also extends our algorithm to graphs of polylogarithmic expansion rate. Recent small world models provide augmentation processes of large graph classes into navigable small worlds via the addition of new random links to each node. However, the computation of the required random links distribution kept these processes unrealistic for large decentralized networks. Our algorithm, based on a careful sampling of a set of leader nodes, bypasses these limitations.

Keywords: Distributed algorithms, navigable small world, routing algorithms, bounded growth.

Résumé

Nous abordons le problème de l'augmentation d'un graphe en un petit monde navigable de façon efficace et distribuée. Précisément, nous montrons que si la métrique du graphe considéré est à croissance bornée, en ajoutant un seul arc (aléatoire) par noeud, on obtient un graphe où l'algorithme de routage glouton calcule des chemins de longueur d'espérance polylogarithmique en la taille du graphe, entre toutes les paires de sommets. Cette opération se fait de façon distribuée, en $O(n)$ rondes avec une mémoire polylogarithmique. Les métriques à croissances bornées sont les métriques telles que la taille d'une boule de rayon $2r$ est au plus une constante fois la taille de la boule de même centre et de rayon r. Elles sont actuellement considérées comme un cadre réaliste pour les grands réseaux décentralisés, comme les réseaux pair-à-pair. Les modèles de petits mondes proposés récemment donnent des processus d'augmentation de graphes en petits mondes navigables, par l'ajout d'arcs aléatoires supplémentaires. Toutefois, le calcul de la distribution de ces nouveaux liens aléatoire nécessite, grossièrement, la connaissance de la totalité de la métrique, rendant ces processus d'augmentations non réalistes pour un grand réseau distribué. En utilisant un échantillonnage aléatoire des sommets bien choisi, notre algorithme d'augmentation parvient à dépasser ces limitations.

Mots-clés: Algorithmes distribués, petits mondes navigables, algorithmes de routage, croissance bornée

1 Introduction

In this paper, we investigate the problem of efficiently preprocessing a large network, in a fully distributed manner, so that the resulting network is a navigable small world. Namely, by adding a single entry to each routing table, we obtain a network in which the greedy routing algorithm computes paths of polylogarithmic expected length between any pair of nodes. This problem arises as an application of recent investigations on the small world phenomenon in real interaction networks (e.g. social networks, or Peer-to-peer networks). The observation of the small world phenomenon in such networks consists in the combination of a low diameter *and* the ability, for each node, to discover short paths without the global knowledge of other nodes connections. This corresponds to the popular notion of the six degrees of separation, exhibited by the seminal experiment of the psychologist Milgram in social networks [17]. With the development of large decentralized networks, the understanding of the underlying causes of the small world phenomenon finds an area of promising applications.

The first graph model that reproduces the small world navigability was proposed by Kleinberg in 2000 [13] and consists in a 2-dimensional regular grid augmented by a constant number of random long-range links per node, distributed according to the 2-harmonic distribution. Kleinberg shows that, with the only knowledge of the grid, the greedy routing, which chooses the closest node to the target among its neighbor in each step, computes paths of polylogarithmic expected length between any two nodes. Further investigations pointed out the general characteristics of these models, by extending it to d-dimensional tori [3], suggesting a more general model. Recently, several augmentation processes have been proposed for larger graph classes [7, 19, 6] ; results are summarized in Figure 1. The greedy routing considered in these augmented graphs, only requires the knowledge, at each node, of its neighbors and of an oracle that, given two nodes, determines which one is the closest to some target node in the graph *before augmentation*. Thus, such processes shrink a network of high diameter into a network where greedy routing achieves polylogarithmic path lengths, by adding new links to each node. This application becomes of high interest when the augmentation process requires the addition of only one new link per node, which consists simply in the addition of one entry to the routing table in a virtual network. Formally, we tackle the following problem :

Problem 1 *Given a network, in which there exists an oracle answering the query "*$d(\mathbf{u}, \mathbf{w}) \geq d(\mathbf{v}, \mathbf{w})$ *?" in constant time, design an efficient decentralized algorithm, which adds 1 link per node, so that the greedy routing algorithm computes a path of polylogarithmic expected length between any pair of nodes, in the augmented graph.*

The hypothesis of such an oracle is realistic for many kinds of networks. Grids with discrete coordinates, hypercubes, Delaunay graphs and Yao graphs (also called Θ-graph) provide this oracle with their own node labeling ; the greedy routing has been well studied in these topologies [4]. Moreover, such an oracle can also be built through a preprocessing of the network. For instance, low cost sophisticated techniques, based on spanners [21], or distance-labeling [9], enable to compute quickly the exact and an approximate distance between nodes. Another approach consists in constructing compact routing tables (see [10, 21, 1] for results about arbitrary graphs). For bounded growth graphs [2] or bounded doubling dimension graphs [19], routing tables of polylogarithmic size can be designed at each node so that the routing path lengths are almost optimal (up to a factor $1 + \epsilon$ of the shortest path).

Shrinking the diameter and the routing path length of a network is an important task for several distributed efficient network constructions. Peer-to-peer networks are of special interest in this framework. Recent peer-to-peer overlay networks [18, 8, 16, 20] aim at directly designing a network guaranteeing a fast routing. However, dynamically maintaining a precise topology can be difficult. In a more general setting, one starts from an arbitrary topology, this is the case of unstructured peer-to-peer networks [5]. In this context, peers could broadcast a message to start an augmentation process to turn the network into a navigable small world, along the specifications of Problem 1.

The key of the small world augmentation processes given in [7, 19, 6], is that the random distribution of the new links fits the underlying graph so as to create shortcuts at all distance scales. However, as these link distributions depend on the graph structure properties, their computation roughly requires the exact knowledge of all the graph connections. In this paper, we propose a fully distributed algorithm which uses a random sampling of the graph to bypass this requirement of a centralized computation.

Ref.	Underlying structure	Out-degree	Expected path length	Scheme
[7]	Treewidth k	1	$O(k \log k \log^2 n)$	Centralized description
[14]	Group structure[1]	$O(\log^2 n)$	$O(\log^2 n)$	
[19]	2^α doubling dimension[2]	$O(2^{O(\alpha)} \log n \log \Delta)$	$O(\log n)$	
[6]	b-moderate growth[3]	1	$O((\log n)^{5/2+2b})$	
this paper	$O(1)$ expansion rate	1	$O((\log n)^2)$	Decentralized
this paper	$O((\log n)^\beta)$ expansion rate	1	$O((\log n)^{6\beta+2})$	Decentralized

FIG. 1 – Small world augmentation processes.

1.1 Model and main result

In the following, we refer to the set of nodes within distance r from some node $\mathbf{u}$ as the *ball* of center $\mathbf{u}$ and radius r, denoted by $\mathcal{B}_\mathbf{u}(r)$; we denote by $b_\mathbf{u}(r)$ its cardinality. A network is said to have an *expansion rate* $C(n)$ if there exists an uniform function $C(n)$ such that, for any node $\mathbf{u}$ and any radius $r > 0$, $b_\mathbf{u}(2r) \leq C(n) b_\mathbf{u}(r)$. In this paper, we consider unweighted networks of polylogarithmic expansion rate. We describe our results for networks of *bounded growth*, that are networks of constant expansion rate, and show that they still hold, with degraded performances, with a polylogarithmic expansion rate. Bounded growth networks have been studied recently in the area of object locations as a good representation of real graphs such as the Internet or peer-to-peer networks [12]. We say that an algorithm is a *small world augmentation process* if it consists in the addition of new random links to each node, so that there exists a decentralized algorithm that computes a path of polylogarithmic expected length between any pair of nodes in the resulting augmented network. A *long range contact* of a node is a new neighbor after the augmentation. We design a small world augmentation process, inspired from the random link distribution given in [19]. To bypass the centralized computations of ball sizes required to compute the new links in [19], we use a multi-layers sample scheme of the network, which is in charge of the link computations of other nodes. Moreover, our augmentation process only requires the addition of one long range contact per node.

Our distributed scheme applies to *synchronized* networks, *i.e.* a global clock is known by all the nodes, and, at each pulse of the clock, each node is allowed to run an action. The period of time between two pulses is called a *round*. We assume that nodes communicate by message-passing. Let Δ the maximal degree of the network. In the Δ-port model, a node can send, during each round, a constant number of messages to its neighbors. We assume that nodes are equipped by unique identifiers, and that each node knows the identifiers of its neighbors. The existence of an oracle fitting Problem 1 specifications is self-contained in the assumption that routing tables of the nodes enable to answer the distance comparison queries. We assume that each node has the knowledge of the size n of the network[4], and of the identifiers of its neighbors. The amount of memory per node is thus at least $\Omega(\Delta \log n)$ bits. The major criteria to evaluate the efficiency of distributed algorithms are the time complexity (number of rounds), the message complexity (total number of messages), and the amount of memory required per node, or per message. We do not take into account the time complexity due to the local computations between each round (these are $O(\Delta)$-time long and dominated by node-to-node communications).

The following Theorem is our main result. To our knowledge, this is the first *fully distributed* algorithm to augment any graph of bounded, or polylogarithmic, expansion rate into a small world, via the addition of one link per node.

Theorem 1 *Any synchronized n-node network of bounded growth, of diameter D, and maximum degree Δ, can be turned into a small world via the addition of one link per node,*

- *in $O(n)$ rounds, with an expected number of messages $O(nD \log n)$, and requiring $O(\Delta \log n \log D)$ memory size with high probability ;*

[1] A group structure is a framework which includes metrics of polynomial ball growth and hierarchies.

[2] A metric has 2^α doubling dimension if each ball of radius $2r$ can be covered by 2^α balls of radius r. Δ is the aspect ratio of the metric,*i.e.* the ratio of the largest distance over the smallest one.

[3] A graph has b-moderate growth if the ratio of a ball of radius $2r$ is at most $O(\log^b r)$ times the size of the ball of radius r and of same center, and the size of a sphere of radius r is at most $1/r$ times the size of the ball of same center and same radius.

[4] If the hypothesis is too restrictive, preprocesses can be used to obtain an approximation of n, see e.g. [11].

– or *in $O(D)$ rounds, with an expected number of messages $O(n \log n \log D)$ and requiring $O(n)$ bits of memory in each node with high probability.*

In the augmented network, the greedy routing algorithm computes paths of expected length $O(\log D \log \delta + \log n)$ between any pair of nodes at mutual distance δ in the original network.

Moreover, our process can be easily extended to produce any desired degree distribution, as long as one node has at least one new link with constant probability.

In [15] Liben-Nowell *et al.* recently experimentally demonstrated that, in a social network, the distribution of friendships that do not depend on the geography fits the inverse ball size distribution, which is close to the long range contact distribution produced by our small world process. In terms of social networks, our small world augmentation process offers the first decentralized scheme to turn a network into a small world. It may thus be a new perspective in the understanding of the creation of shortcuts which gives rise to the small world phenomenon in a social network.

1.2 Outline of the paper

Sections 2 and 3 focus on bounded growth networks, polylogarithmic expansion rate networks are treated in section 4. Section 2 gives a high level description of our augmentation process which includes a random sampling and a random choice of new links based on the sample. We show that the random sampling succeeds w.h.p., and that, under this condition, the greedy routing algorithm computes paths of polylogarithmic expected length in the augmented network. Section 3 provides the distributed implementations of both sampling and link choices processes, and analyzes the total cost under both high and low memory settings.

In all the following, we consider a network $G = (V, E)$ of expansion rate c, diameter D and maximum degree Δ. We will use the abbreviations *w.h.p.* standing for *with high probability* and *u.a.r.* standing for *uniformly at random.*

2 Sampling-based small world algorithm

One of the main issues with the recent small world processes given in the literature [14, 19, 7, 6] is that they require the complete knowledge of the underlying graph. More precisely, to compute the random link distribution, [7] requires to compute a tree decomposition and [14, 19, 6] require the size of all balls centered at any given node. In [6], the ball sizes centered on each node $\mathbf{u}$ are used to choose randomly the length ℓ of the random link to be drawn from $\mathbf{u}$: ℓ is chosen with probability proportional to $1/b_{\mathbf{u}}(\ell)$; the long-range contact of $\mathbf{u}$ is then picked u.a.r. among the nodes at distance exactly ℓ from $\mathbf{u}$. In [19], each node $\mathbf{u}$ chooses $\log n$ long range contacts, each u.a.r. in each ball $\mathcal{B}_{\mathbf{u}}(2^j)$, for $1 \leq j \leq \lceil \log D \rceil$. The principle of our scheme is to avoid the computation of all ball sizes for all nodes in the network, in such a way that each node only explores $O(\log n)$ nodes in the network on expectation.

2.1 Our random augmentation process

Description of the small world algorithm (Algorithm 1). The basic idea is that each node $\mathbf{u}$ will not compute its long range contact by itself but will only choose a random length scale $\ell = 2^i$, by picking i u.a.r. in $\{\lceil \log(2c \log n) \rceil, \ldots, \lceil \log D \rceil\}$, and asks another node $\mathbf{v}$, at distance $\leq \ell$ from $\mathbf{u}$, to pick u.a.r. $\mathbf{u}$'s long range contact in its own ball $\mathcal{B}_{\mathbf{v}}(3\ell)$. Such a node $\mathbf{v}$ is called the *ith level leader* for $\mathbf{u}$. Only the ith level leaders will have to explore their ball of radius $3 \cdot 2^i$; all other nodes will refer to them to compute their long range contact. On the one hand, ith level leaders should not be too far from each other so that each node is close enough to one of them to guarantee that this approximation is sufficiently precise. On the other hand, each ith level leader $\mathbf{v}$ will explore $b_{\mathbf{v}}(3 \cdot 2^i)$ nodes, and we must guarantee that the expected number of nodes visited by each nodes remains polylogarithmic. We will write S_i to describe the set of the ith level leaders. The selection of the ith level leader is completed by the SAMPLE algorithm detailed next. We first analyze the algorithm, its decentralized implementation will be described in section 3.

Lemma 1 *The probability that the sampling fails during step 1 of Algorithm 1 is less than* $\log n/n$.

Algorithme 1 SAMPLING-BASED SMALL WORLD ALGORITHM

In our scheme, the ith level leaders are in charge of drawing long range contacts at distance $O(2^i)$.

1. Let $k_0 = \lceil \log(2c \log n) \rceil$. For all $i \in \{k_0, \ldots, \lceil \log D \rceil\}$, each node $\mathbf{u}$ is chosen to be an ith level leader with probability $2c \log(n)/b_{\mathbf{u}}(2^i)$ independently ; this step is achieved by running for all $i \in \{k_0, \ldots, \lceil \log D \rceil\}$ the SAMPLE$(\mathbf{u}, i)$ algorithm (Algorithm 2) in parallel for all node $\mathbf{u} \in V$.
2. Then, each ith level leader informs all nodes at distance $\leq 2^i$ that it is their ith level leader (ties are arbitrarily broken).
3. If there exists a node $\mathbf{u}$ that did not get an ith leader for some i, go to step 1 (we say that the sampling phase *failed*).
4. Each ith level leader explores its ball of radius $3 \cdot 2^i$.
5. Each node $\mathbf{u}$ in the network then picks u.a.r. $i \in \{k_0, \ldots, \lceil \log D \rceil\}$ and asks a long range contact request to its ith level leader $\mathbf{v}$; $\mathbf{v}$ picks then $\mathbf{u}$'s long range contact, $L(\mathbf{u})$, u.a.r. in its ball $\mathcal{B}_{\mathbf{v}}(3 \cdot 2^i)$.

Proof. Consider a node $\mathbf{u}$ and let $i \in \{\lceil \log(2c \log n) \rceil, \ldots, \lceil \log D \rceil\}$. The probability that a node $\mathbf{v}$ belongs to S_i is $2c \log n / b_{\mathbf{v}}(2^i)$. By inclusion, any node $\mathbf{v}$ in $\mathcal{B}_{\mathbf{u}}(2^i)$ satisfies : $b_{\mathbf{v}}(2^i) \leq b_{\mathbf{u}}(2^{i+1})$. From the bounded growth, we have $b_{\mathbf{u}}(2^{i+1}) \leq c\, b_{\mathbf{u}}(2^i)$. The probability that no node of S_i lies into $\mathcal{B}_{\mathbf{u}}(2^i)$ is then less than :

$$\left(1 - \frac{2c \log n}{c\, b_{\mathbf{u}}(2^i)}\right)^{b_{\mathbf{u}}(2^i)} \leq \frac{1}{n^2}.$$

From the union bound, the probability that it happens for at least one level i is then less than $\log D / n^2$ (since there are less than $\log D$ levels). Finally, from the union bound again, the probability that this failure happens for at least one node is less than $\log D / n \leq \log n / n$, the result follows. □

***i*th level leaders selection (Algorithm 2).** Algorithm SAMPLE$(\mathbf{u}, i)$ decides parsimoniously whether $\mathbf{u}$ is an ith level leader and, once run over all nodes $\mathbf{u}$, ensures that w.h.p. all nodes $\mathbf{v}$ are at distance $\leq 2^i$ from some ith level leader (Lemme 1). The procedure explores bigger and bigger balls around each node. Each call to SAMPLE$(\mathbf{u}, i)$ launches the successive explorations of the balls of increasing sizes $\mathcal{B}_{\mathbf{u}}(2^{k_0}), \ldots, \mathcal{B}_{\mathbf{u}}(2^i)$. If $\mathbf{u}$ survives all exploration trials, it explores the ball $\mathcal{B}_{\mathbf{u}}(3 \cdot 2^i)$ that will be used later to pick u.a.r. the long range contacts.

Lemma 2 *For all $i \in \{k_0, \ldots, \lceil \log D \rceil\}$, and for every node $\mathbf{u} \in V$, $\Pr\{\mathbf{u} \in S_i\} = (2c \log n)/b_{\mathbf{u}}(2^i)$.*

Proof. Let $i \in \{k_0, \ldots, \lceil \log D \rceil\}$ and $\mathbf{u} \in V$. The node $\mathbf{u}$ is inserted in S_i (the set of the ith level leaders) if and only if it succeeded all the probabilistic trials up to $k = i$. $\mathbf{u}$ survives to the first trial with probability $2c \log n / b_{\mathbf{u}}(2^{k_0})$. It will then survive to next one with probability $b_{\mathbf{u}}(2^{k_0})/b_{\mathbf{u}}(2^{k_0+1})$, etc. The

Algorithme 2 SAMPLE$(\mathbf{u}, i)$

Algorithm SAMPLE$(\mathbf{u}, i)$ *proceeds by successive flooding phases of increasing radii.*

$k_0 := \lceil \log(2c \log n) \rceil$. $k := k_0 + 1$.
Explore $\mathcal{B}_{\mathbf{u}}(2^{k_0})$.
Die with probability $1 - 2c \log n / b_{\mathbf{u}}(2^{k_0})$.
tant que u is alive and $k \leq i$ **faire**
    Explore $\mathcal{B}_{\mathbf{u}}(2^k)$.
    Die with probability $1 - b_{\mathbf{u}}(2^{k-1})/b_{\mathbf{u}}(2^k)$.
    $k := k + 1$.
fin tant que
si u is alive **alors**
    u is a ith level leader.
fin si

node $\mathbf{u}$ is *alive* at the end of the process if it has succeeded all independent trials unto the value $k = i$, i.e., with total probability :

$$\frac{2c\log n}{b_{\mathbf{u}}(2^{k_0})} \times \frac{b_{\mathbf{u}}(2^{k_0})}{b_{\mathbf{u}}(2^{k_0+1})} \times \cdots \times \frac{b_{\mathbf{u}}(2^{i-1})}{b_{\mathbf{u}}(2^i)} = \frac{2c\log n}{b_{\mathbf{u}}(2^i)}.$$

□

Guessing the expansion rate. Note that the knowledge of the exact value of the expansion rate c is not required to run SAMPLE nor SAMPLING-BASED SMALL WORLD algorithms. Indeed, one can start with $c = 1$ and if the sampling fails, then rerun the algorithm with $c := 2c$ until the sampling succeeds. By Lemma 1 w.h.p., only $\lceil \log c \rceil$ runs are required before success. We will thus now assume that the expansion rate c is known.

2.2 Greedy routing analysis in the augmented graph

We now assume that the sampling succeeds. We show that the resulting augmented graph is a navigable small world by showing that greedy routing computes path of polylogarithmic expected length between any pair of nodes. We first lower bound the probability that a node $\mathbf{v}$ is the long range contact, $L(\mathbf{u})$, of a given node $\mathbf{u}$ in terms of the distance $d(\mathbf{u}, \mathbf{v})$ in the original graph. This lemma is the key to prove the polylogarithmic upper bound on the greedy routing path length (Theorem 2).

Lemma 3 *For all $\mathbf{v} \neq \mathbf{u}$ in V such that $d(\mathbf{u}, \mathbf{v}) \geq 2c\log n$, $\Pr\{L(\mathbf{u}) = \mathbf{v}\} \geq \frac{1}{c^2 \log D} \frac{1}{b_{\mathbf{u}}(d(\mathbf{u}, \mathbf{v}))}$.*

Proof. Since $d(\mathbf{u}, \mathbf{v}) \geq 2c\log n$, let $i \in \{k_0, \ldots, \lceil \log D \rceil\}$ such that $2^i \leq d(\mathbf{u}, \mathbf{v}) < 2^{i+1}$. Observe that $d(\mathbf{v}, \text{leader}_{\mathbf{u}}(i)) \leq 3 \cdot 2^i$ by the triangle inequality. The probability that $L(\mathbf{u}) = \mathbf{v}$ is thus greater than the probability that $\mathbf{u}$ has chosen the level i and that $\mathbf{v}$ has been chosen uniformly among the nodes of $\mathcal{B}_{\text{leader}_{\mathbf{u}}(i)}(3 \cdot 2^i)$. The level i is chosen with probability at least $1/\log D$. Since, by definition of the ith level leader, $d(\mathbf{u}, \text{leader}_{\mathbf{u}}(i)) \leq 2^i$, we have $b_{\text{leader}_{\mathbf{u}}(i)}(3 \cdot 2^i) \leq b_{\mathbf{u}}(4 \cdot 2^i) \leq b_{\mathbf{u}}(4d(\mathbf{u}, \mathbf{v}))$. By bounded growth, $b_{\mathbf{u}}(4d(\mathbf{u}, \mathbf{v})) \leq c^2 b_{\mathbf{u}}(d(\mathbf{u}, \mathbf{v}))$. The result follows. □

Theorem 2 *Greedy routing computes, in the augmented graph, a path of expected length $O(\log D \log \delta + \log n)$, between any pair of nodes at mutual distance δ in the original graph.*

Proof. Let $\mathbf{s}$ and $\mathbf{t}$ the source and target, at mutual distance δ. Divide the execution of the greedy algorithm into $\log \delta$ phases : the algorithm is in phase j, $1 \leq j \leq \lceil \log \delta \rceil$, as long as the current distance to $\mathbf{t}$ (in the original graph) belongs to $(2^{j-1}, 2^j]$.

Assume the algorithm is in phase j and that $\mathbf{u}$ is the current message holder. Assume that $j \geq k_0 + 2$, any node $\mathbf{v}$ in $\mathcal{B}_{\mathbf{t}}(2^{j-2})$ is at distance at least $2^{j-2} \geq 2c\log n$ from $\mathbf{u}$. By Lemma 3, any node $\mathbf{v}$ in $\mathcal{B}_{\mathbf{t}}(2^{j-2})$ is the long range contact of $\mathbf{u}$ with probability at least $1/(c^2 \log D \cdot b_{\mathbf{u}}(d(\mathbf{u}, \mathbf{v})))$; moreover $d(\mathbf{u}, \mathbf{v}) \leq 5 \cdot 2^{j-2}$. Thus, the long range contact of $\mathbf{u}$ belongs to $\mathcal{B}_{\mathbf{t}}(2^{j-2})$ with probability at least :

$$\frac{1}{c^2 \log D} \frac{b_{\mathbf{t}}(2^{j-2})}{b_{\mathbf{u}}(5 \cdot 2^{j-2})},$$

By inclusion, $b_{\mathbf{u}}(5 \cdot 2^{j-2}) \leq b_{\mathbf{t}}(10 \cdot 2^{j-2})$, and by bounded growth hypothesis, $b_{\mathbf{t}}(10 \cdot 2^{j-2}) \leq c^4 b_{\mathbf{t}}(2^{j-2})$. The probability to exit phase j is thus at least $1/(c^6 \log D)$ at each step during this phase. The expected number of steps in each phase $j \geq k_0 + 2$ is then $O(\log D)$. As soon as $j < k_0 + 2$, we can simply upper bound the remaining expected number of steps by $2^{k_0+1} = O(\log n)$. Finally, summing over the $\lceil \log \delta \rceil$ phases, the total expected path length is $O(\log D \log \delta + \log n)$. □

Note that the proof of Theorem 2 can be lead similarly if we choose to add k long range contact per node, k being determined by some probability distribution, as long as each node has a constant probability to have at least one long range contact. Indeed, it suffices to multiply the lower bound given in Lemma 3 by this constant, and the proof of the theorem is identical.

3 Decentralized implementations and performances

The only two steps in SAMPLING-BASED SMALL WORLD algorithm that require a careful decentralized implementation are the exploration step in SAMPLE and the random generation of the long range contacts (which consists in routing each request to the appropriate leader, picking a random node in the leader's ball, and sending the contact back).

3.1 The sampling step

In the sampling step, balls are explored in breadth first search order. In order to simplify the message passing within each ball, we grow a shortest path tree $\mathcal{T}_{\mathbf{u},i}$ during the exploration. The tree structure is maintained by storing in each explored node its parent and children (ties are arbitrarily broken). If $\mathbf{u}$ dies at trial k, $\mathcal{T}_{\mathbf{u},i}$ spans the ball $\mathcal{B}_{\mathbf{u}}(2^k)$. If $\mathbf{u}$ survives all exploration trials, $\mathbf{u}$ informs all nodes in $\mathcal{B}_{\mathbf{u}}(2^i)$, using tree structure $\mathcal{T}_{\mathbf{u},i}$, that it is their ith level leader ; $\mathcal{T}_{\mathbf{u},i}$ spans then the ball $\mathcal{B}_{\mathbf{u}}(3 \cdot 2^i)$ and $\mathbf{u}$ requests each of its children to compute and store recursively the size of their subtree in $\mathcal{T}_{\mathbf{u},i}$. These sizes will be used later to pick a node u.a.r. in $\mathcal{B}_{\mathbf{u}}(3 \cdot 2^i)$ upon request.

The next lemma tighten our sampling analysis by showing that the set of leaders is sparse enough. This will be useful in our implementations analysis.

Lemma 4 *W.h.p., any node* $\mathbf{u}$ *belongs to* $O(\log n \log D)$ *trees* $\{\mathcal{T}_{\mathbf{v},i}\}_{\mathbf{v} \in V, i \leq \log D}$.

Proof. Let $\mathbf{u} \in V$ and $k \geq k_0$. For any node $\mathbf{v}$, , and $j \geq k$, the probability that $\mathbf{v}$ explores, during the selection of the ith leader, at least upto the radius 2^k is $2c \log n / b_{\mathbf{v}}(2^k)$.Let $X_{\mathbf{v},j}$ be the random variable equals to 1 if $\mathbf{v}$ explores a radius at least 2^k during the selection of the ith leader, and 0 otherwise. Thus $\mathbb{E}[X_{\mathbf{v},j}] = 2c \log n / b_{\mathbf{v}}(2^k)$, if $j \geq k$ and 0 otherwise.

$\mathbf{u}$ belongs to all trees rooted in $\mathbf{v} \in \mathcal{B}_{\mathbf{u}}(2^k)$ that have explored unto the radius 2^k. For all nodes $\mathbf{v} \in \mathcal{B}_{\mathbf{u}}(2^k)$, we have $b_{\mathbf{v}}(2^k) \leq b_{\mathbf{u}}(2^{k+1}) \leq c\, b_{\mathbf{u}}(2^k)$ by the bounded growth. Moreover, $b_{\mathbf{v}}(2^k) \geq b_{\mathbf{v}}(2^{k+1})/c$ by the bounded growth. Thus, for all nodes $\mathbf{v}$ in $\mathcal{B}_{\mathbf{u}}(2^k)$, $\mathbb{E}[X_{\mathbf{v},j}] \in [(2 \log n)/b_{\mathbf{u}}(2^k), 2c^2 \log n / b_{\mathbf{u}}(2^k)]$. Let $X = \sum_{\mathbf{v} \in \mathcal{B}_{\mathbf{u}}(2^k)} \sum_j X_{\mathbf{v},j}$, we have $\mathbb{E}[\sum_j X_{\mathbf{v},j}] \in [2 \log n, 2c^2 \log n \log D]$ by linearity of expectation. Since variables $X_{\mathbf{v},j}$ are independent, we can use a Chernoff bound to upper bound the probability that their sum is greater than $4\mathbb{E}[X]$:

$$\Pr\{X > 4\mathbb{E}[X]\} \leq [\frac{1}{e}(e/4)^4]^{\mathbb{E}[X]} \leq \frac{1}{e^{2 \log n}} \leq \frac{1}{n^2}.$$

Thus, with probability greater than $1 - 1/n^2$, there are less than $O(\log n \log D)$ trees rooted in $\mathcal{B}_{\mathbf{u}}(2^k)$ that traverses $\mathbf{u}$. Note that we can conduct the same argument at all scale $k \leq \log D$. From the union bound, the probability that there exists a ball $\mathcal{B}_{\mathbf{u}}(2^k)$, with more than $\log n$ trees that traverses $\mathbf{u}$ is less than $\log D / n^2$. Thus, there are less than $O(\log n (\log D))$ trees that traverses $\mathbf{u}$ with probability greater than $1 - \log D / n^2$. Applying once more the union bound yields that this is true for all nodes $\mathbf{u} \in V$ with probability greater than $1 - \log D / n$.□

Proposition 1 *The sampling step in* SAMPLE-BASED SMALL WORLD *(Algorithm 1) requires* $O(D)$ *rounds, the exchange of* $O(n\Delta \log n (\log D)^2)$ *messages on expectation, and* $O(\Delta \log n \log D)$ *memory size in each node w.h.p..*

Proof. The number of rounds that requires SAMPLE$(\mathbf{u}, i)$ is at most $6 \cdot 2^i + 1$ in the case where the exploration succeeds and reach the radius $3 \cdot 2^i$, because messages have to go down to the leaves along the radius $3 \cdot 2^i$, and then then sent back to the root. A given node $\mathbf{u}$ executes SAMPLE$(\mathbf{u}, i)$ for i from k_0 to $\lceil \log D \rceil$, which thus requires at most $O(D)$ rounds. Since the executions are run in parallel, the total number of rounds is $O(D)$.

In SAMPLE$(\mathbf{u}, i)$, each inner node of the tree exploration sends at most Δ messages at most three times. With probability $2c \log n / b_{\mathbf{u}}(2^k)$, the tree exploration ends at radius 2^k and contains $b_{\mathbf{u}}(2^k)$ nodes, yielding a number of messages $O(\Delta b_{\mathbf{u}}(2^k))$. The expected number of messages of for the level i is then $O(i\Delta)$, and $O(\Delta (\log D)^2)$ for the total execution on the node $\mathbf{u}$. By linearity of expectation, the total number of messages $O(n\Delta (\log D)^2)$ in expectation.

As for the memory, a node $\mathbf{u}$ has to store enough information for the execution of a DFS on each tree that traverses it during the execution of SAMPLE on all nodes and all levels. This requires a memory size $O(\Delta)$ for each tree. From lemma 4, any node is traversed by at most $O(\log n \log D)$ trees w.h.p., the result follows. □

3.2 The long range contact request steps

During this step, each node $\mathbf{u}$ chooses u.a.r. $i \in \{k_0, \ldots, \lceil \log D \rceil\}$ and requests a long range contact to its ith level leader. We propose two implementations depending on the memory available at the nodes.

Large memory setting. In this first implementation, we aim at minimizing the overall number of messages though the network, assuming that a large memory is available in the nodes. The requests addressed to each ith level leader $\mathbf{u}$ flow synchronously from leaves to $\mathbf{u}$ along $\mathcal{T}_{\mathbf{u},i}$. Each inner node merges the requests received from its children into a single message, and forwards it to its parent with its own request (if needed). Once $\mathbf{u}$ has received all its requests (after 2^i rounds), say x requests, $\mathbf{u}$ picks x integers $q_1, \ldots, q_x$ u.a.r. in $\{1, \ldots, b_{\mathbf{u}}(3 \cdot 2^i)\}$, and launches the prefix searches in $\mathcal{T}_{\mathbf{u},i}$ for the q_1th, $\ldots$, q_xth nodes $\mathbf{v}_1, \ldots, \mathbf{v}_x$ (using the subtree sizes stored in each inner node). The long range contacts $\mathbf{v}_1, \ldots, \mathbf{v}_x$ are then sent back to the leader (using the same merging scheme as before) and then to their destination (using either the oracle or the rank of the destination in the prefix order in $\mathcal{T}_{\mathbf{u},i}$ as above) .

Proposition 2 *The large memory request scheme requires : $O(D)$ rounds, $O(n \log n \log D)$ messages in expectation and a memory size $O(n)$.*

Proof. In this scheme, sample nodes of level $\lceil \log D \rceil$ may have to record up to n requests of links in the worst case where every node have chosen the $\lceil \log D \rceil$-th level, and then requires a linear memory size for these nodes.

The number of rounds required to execute the long-range link request phase is at most four times the number of rounds required by the largest DFS, *i.e.* $O(D)$.

Concerning the number of messages, each tree $\mathcal{T}_{\mathbf{s},i}$ rooted at a ith level leader $\mathbf{s}$ produces at most $3b_{\mathbf{s}_i}(2^i)$ messages. To count the number of messages, one can consider that each sample node of level i induces at most $3b_{\mathbf{s}_i(2^i)}$ messages, and the other nodes do not induce messages. The overall expected number of messages through the execution is then less than $\sum_{\mathbf{u} \in V} \sum_{i=k_0+2}^{\lceil \log D \rceil} 3b_{\mathbf{u}}(2^i) \Pr\{\mathbf{u} \in S_i\} = O(n \log n \log D)$. □

Low memory setting. In the previous scheme, nodes of high levels are the bottlenecks of the network, and the amount of memory they require can be unrealistic for some applications. Note first that a polylogarithmic memory size for every node cannot be achieved if the number of rounds is $o(n)$. Indeed, assume the process runs in R rounds, even if requests to the leaders are properly scheduled, there is at least one round for which a sample node of the highest level may receive a link request from n/R nodes. In order to match an amount of memory of size $O(\log^a n)$, at *any node*, for some constant $a > 0$, R has then to be of order $\Omega(n/\log^a n)$. We present here an alternative implementation of the long range contacts request phase, which achieves the augmentation process in $O(n)$ rounds with only polylogarithmic memory size for all nodes w.h.p..

In fact, the long range contacts requests within a leader tree can be scheduled in such a way that there is at most one new request per round received by the leader. Each ith leader node $\mathbf{u}$ creates a token $\mathrm{OK}_{\mathbf{u},i}$ that traverses $\mathcal{B}_{\mathbf{u}}(2^i)$ along the tree $\mathcal{T}_{\mathbf{u},i}$, i.e., at each round, the token is forwarded to the next node in prefix order. Whenever a node $\mathbf{v}$ receives the token, it sends its long range contact request to its parent $\mathcal{T}_{\mathbf{u},i}$ (if needed). Any node receiving a long range contact request for $\mathbf{u}$ forwards it to its parent in $\mathcal{T}_{\mathbf{u},i}$. Whenever $\mathbf{u}$ receives a long range contact request, it chooses a random integer x in $\{1, \ldots, b_{\mathbf{u}}(3 \cdot 2^i)\}$, launches a prefix search for the corresponding node $\mathbf{v}_x$ in $\mathcal{B}_{\mathbf{u}}(3 \cdot 2^i)$ along $\mathcal{T}_{\mathbf{u},i}$. $\mathbf{v}_x$ stores all nodes that want itself as a long range contact. When token $\mathrm{OK}_{\mathbf{u},i}$ is back to $\mathbf{u}$, $\mathbf{u}$ creates a new token $\mathrm{GO}_{\mathbf{u},i}$ that traverses $\mathcal{B}_{\mathbf{u}}(3 \cdot 2^i)$ in prefix order along $\mathcal{T}_{\mathbf{u},i}$. Whenever a node $\mathbf{v}$ receives the new token, it answers to each of its stored requests one by one at each round until no request are left, and then forwards the token to next node in prefix order. This schedule is illustrated in Figure 2.

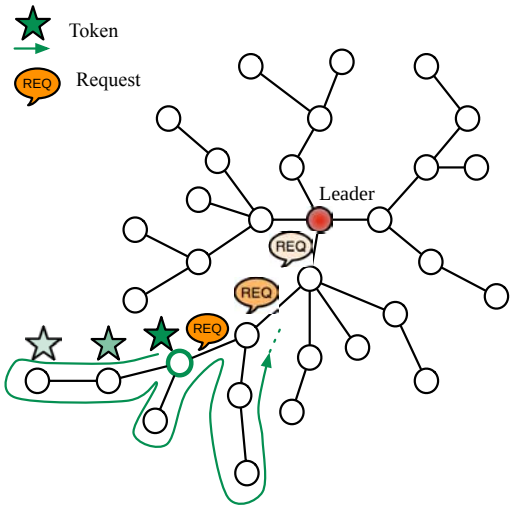


FIG. 2 – Illustration of long range contact requests schedulling in the low memory setting.

This algorithm guarantees that, in $\mathcal{T}_{\mathbf{u},i}$, the number of messages simultaneously sent by a node is at most a constant. From Lemma 4, any node belongs to at most $O(\log n \log D)$ such trees, w.h.p., one can then allow a node to merge $O(\log n \log D)$ potential messages into one message for the same neighbor, at no extra cost.

The following lemma guarantees that the number of requests stored in a node before being sent back to a leader is at most $O(\log n)$, and thus fits our low memory setting. This lemma relies on a standard probability urn result, since picking of long range contacts consists in picking independently random numbers in a set of numbers.

Lemma 5 *A node* $\mathbf{v}$ *is chosen as the long-range contact of at most* $8c^2 \log n$ *nodes with high probability.*

Proof. Let $\mathbf{v}$ a node. Let $X_{\text{ask}}(\mathbf{v})$ the random variable which states for the number of nodes that choose $\mathbf{v}$ as their long-range contact, in the whole network. Denote such nodes as *asking* nodes. The total number of asking nodes is the sum of the number of asking nodes in each tree $\mathcal{T}_{\mathbf{u},i}$ such that $\delta(\mathbf{u},\mathbf{v}) \leq 3 \cdot 2^i$, and $\mathbf{u}$ is a ith level leader. We are going to use an extra random variable $X^\star_{\text{ask}}(\mathbf{v})$ which stochastically dominates $X_{\text{ask}}(\mathbf{v})$. We consider the virtual setting where, for each node $\mathbf{u}$ and each level i, all the nodes $\mathbf{w}$ in $\mathcal{B}_{\mathbf{u}}(2^i)$ prick two numbers uniformly at random : one between k_0+2 and $\lceil \log D \rceil$, and the other one between 1 and $b_{\mathbf{u}}(3 \cdot 2^i)$. Given a node $\mathbf{u}$ and a level i, we denote by $\mathbb{1}_{\mathbf{u},i,\mathbf{w}}$, for all $\mathbf{w} \in \mathcal{B}_{\mathbf{u}}(2^i)$, the random variable which is equal to 1 if $\mathbf{u}$ is a ith level leader and $\mathbf{w}$ has picked the couple of numbers $(i, v_{\mathbf{u},i})$, $v_{\mathbf{u},i}$ being the rank of $\mathbf{v}$ in the prefix order search in $\mathcal{B}_{\mathbf{u}}(3 \cdot 2^i)$ (for instance). Then, the random variable $X^\star_{\text{ask}}(\mathbf{v})$ defined by :

$$X^\star_{\text{ask}}(\mathbf{v}) = \sum_{i=k_0+2}^{\lceil \log D \rceil} \sum_{\mathbf{u} \in \mathcal{B}_{\mathbf{v}}(3 \cdot 2^i)} \sum_{\mathbf{w} \in \mathcal{B}_{\mathbf{u}}(2^i)} \mathbb{1}_{\mathbf{u},i,\mathbf{w}},$$

stochastically dominates $X_{\text{ask}}(\mathbf{v})$. Indeed, in the real framework, a node chooses its long-range contact through a single leader, while in our virtual setting, by making all nodes pick a couple of numbers, we artificially increase the probability to pick the number $v_{\mathbf{u},i}$, which corresponds to picking the node $\mathbf{v}$ uniformly at random.

Note that :

$$\Pr\{\mathbb{1}_{\mathbf{u},i,\mathbf{w}} = 1\} = \frac{2c \log n}{b_{\mathbf{u}}(2^i)} \; \frac{1}{\lceil \log D \rceil - k_0 - 1} \; \frac{1}{b_{\mathbf{u}}(3 \cdot 2^i)}.$$

By the way, for all $\mathbf{u} \in \mathcal{B}_{\mathbf{v}}(3 \cdot 2^i)$, $b_{\mathbf{v}}(6 \cdot 2^i)/c \leq b_{\mathbf{u}}(3 \cdot 2^i) \leq b_{\mathbf{v}}(6 \cdot 2^i)$, by inclusion and bounded growth.

Consequently, by linearity of expectation,

$$2c\log n\frac{b_{\mathbf{v}}(3\cdot 2^i)}{b_{\mathbf{v}}(6\cdot 2^i)} \leq \mathbb{E}[X^\star_{\text{ask}}(\mathbf{v})] \leq 2c^2\log n\frac{b_{\mathbf{v}}(3\cdot 2^i)}{b_{\mathbf{v}}(6\cdot 2^i)},$$

that is : $\mathbb{E}[X^\star_{\text{ask}}(\mathbf{v})] \in [2\log n, 2c^2\log n]$, by using the moderate growth. Since the indicative variables are bounded and independent, we can use a Chernoff bound :

$$\Pr\{X^\star_{\text{ask}}(\mathbf{v}) > 4\mathbb{E}[X^\star_{\text{ask}}(\mathbf{v})]\} \leq [\frac{1}{e}(e/4)^4]^{\mathbb{E}[X^\star_{\text{ask}}(\mathbf{v})]} \leq \frac{1}{e^{2\log n}} \leq \frac{1}{n^2}.$$

Thus, node $\mathbf{v}$ has to record less than $8c^2\log n$ asking nodes with a probability greater than $1-1/n^2$, since $X^\star_{\text{ask}}(\mathbf{v})$ stochastically dominates $X_{\text{ask}}(\mathbf{v})$. By the union bound over all nodes $\mathbf{v}$, each node has to record at most $8c^2\log n$ asking nodes with high probability (greater than $1-1/n$). □

Lemma 6 *Let* $\mathbf{u}$ *be a* i*th level leader. For any round of the low memory request scheme, every node* $\mathbf{v} \in \mathcal{T}_{\mathbf{u},i}$ *sends or receives at most* 3 *messages in* $\mathcal{T}_{\mathbf{u},i}$*.*

Proof. Message passing is organized by means of the tokens $\text{OK}_{\mathbf{u},i}$ and $\text{GO}_{\mathbf{u},i}$, which are located in at most one node The knowledge of message $\text{OK}_{\mathbf{u},i}$ indicates that the current node is allowed to immediately send or forward its request towards $\mathbf{u}$ in the next rounds.

Let $\mathbf{v}$ and $\mathbf{w}$ two nodes in $\mathcal{T}_{\mathbf{u},i}$ and assume that $\mathbf{w}$ receives the message $\text{OK}_{\mathbf{u},i}$ before $\mathbf{v}$. If $\mathbf{w}$ is an ancestor of $\mathbf{v}$ in $\mathcal{T}_{\mathbf{u},i}$, $\mathbf{w}$ will be allowed to send its request before $\mathbf{v}$ and the two requests cannot be located in the same node. As soon as $\text{OK}_{\mathbf{u},i}$ and the link requests are no more located in the same node $\mathbf{z}$, it means that the message $\text{OK}_{\mathbf{u},i}$ visits a new subtree of $\mathcal{T}_{\mathbf{u},i}$ *and* that any node of this subtree has $\mathbf{z}$ as ancestor. The request forwarded by $\mathbf{z}$ cannot be caught up with a link request coming from this subtree.

As long as the token $\text{OK}_{\mathbf{u},i}$ is not back to $\mathbf{u}$, the long range contact request of a node $\mathbf{w}$ can be decomposed into two steps : (1) the routing toward the leader , and (2) the routing from $\mathbf{u}$ to a random node $\mathbf{v}_x$. Since $\mathbf{u}$ receives at most one request per round, and as the routing tasks are done along shortest paths, every node of $\mathcal{T}_{\mathbf{u},i}$ receives at most one request per round from its parent. Finally, each node of $\mathcal{T}_{\mathbf{u},i}$ receives an overall of at most 3 messages per round. The same analysis holds for the way back of the long-range contacts, with the use of token $\text{GO}_{\mathbf{u},i}$. □

Proposition 3 *The low memory request scheme requires at most* $4n$ *rounds,* $O(\Delta\log n\log D)$ *memory size in all nodes w.h.p., and* $O(nD\log n)$ *messages on expectation.*

Proof. The number of rounds required by the new implementation is the time needed to schedule the requests. This algorithm is based upon a DFS traversal. For a tree $\mathcal{T}_{\mathbf{u},i}$, the traversal is done in $b_{\mathbf{u}}(2^i)$ rounds and the first phase of the link requests are done within $b_{\mathbf{u}}(2^i)+2^i$ rounds, which is at most $2b_{\mathbf{u}}(2^i)$. As for the second phase, we have to add the extra delay due to the number of nodes of $\mathcal{T}_{\mathbf{u},i}$ that have chosen $\mathbf{u}$ as their leader. This phase is done in at most $2b_{\mathbf{u}}(3\cdot 2^i)+2b_{\mathbf{u}}(2^i)$ rounds, and the scheduling is achieved within $4n$ rounds for any node $\mathbf{u}$ and any level i.

We now deal with the total number of messages. Since a token makes twice a DFS traversal, there are $2b_{\mathbf{u}}(3\cdot 2^i)$ messages dedicated to the token. The number of requests from $\mathcal{T}_{\mathbf{u}}(2^i)$ that are sent to $\mathbf{u}$ is at most $b_{\mathbf{u}}(2^i)$. Each request produces at most $4\cdot 3\cdot 2^i$ messages. To count the number of messages, one can consider that each ith leader node of level i induces at most $O(2^i b_{\mathbf{u}}(2^i))$ messages, and the other nodes do not induce messages. Let $\mathbf{v} \in V$. The probability that $\mathbf{v}$ is a ith level leader is $2c\log n/b_{\mathbf{v}}(2^i)$, the expected number of messages per node is thus $\sum_{i\leq\lceil\log D\rceil} O(2^i\log n)) = O(D\log n)$, yielding an expected overall number of messages $O(nD\log n)$.

Concerning the memory size, from Lemma 4, each node requires $O(\Delta\log n\log D)$ to store all the information for the trees that traverse it. From Lemma 5, the memory required to store the requests before sending them is at most $8c^2\log n$ requests w.h.p., the result follows. □

4 Polylogarithmic expansion rate

In this section, we extend our distributed small world algorithm to networks of polylogarithmic expansion rate $O((\log n)^\beta)$, $\beta \geq 0$. For a network of expansion rate $c(\log n)^\beta$, $c > 0, \beta \geq 0$, the only important change consists in setting the value of k_0 to $\lceil \log(2c(\log n)^{\beta+1} \rceil$, and the initial trial probability of SAMPLE$(\mathbf{u}, i)$ to $2c(\log n)^{\beta+1}/b_{\mathbf{u}}(2^{k_0})$. The probability of success of the sample remains then unchanged. Since the value of k_0 is required to run the algorithm SAMPLE, one could think that the knowledge of the exact constants c, β is required. However, since SAMPLE succeeds w.h.p., one can guess the values of c and β as follows : start with $c = 1$, and $\beta = 0$; restart with $c := 2c$, until $c \geq \log n$; then, restart with $\beta := 1$ and $\beta := \beta + 1$, until the sampling phase succeeds. This guarantees, that within $O(\log \log n)$ executions, the sample succeeds, and the error on the true constant β cannot be larger than $O(1)$.

The following theorem states our extended results to networks of polylogarithmic expansion rates.

Theorem 3 *Any synchronized n-node network of expansion rate $O((\log n)^\beta)$, diameter D, and maximum degree Δ, can be turned into a small world via the addition of one link per node,*

- *in $O(n)$ rounds, with an expected number of messages $O(nD(\log n)^{\beta+1})$, and requiring $O(\Delta(\log n)^{3\beta+1})$ memory size with high probability ;*
- or *in $O(D)$ rounds, with an expected number of messages $O(n(\log n)^{\beta+1}(\log D)^3)$ and requiring $O(n)$ bits of memory in each node with high probability.*

In the augmented network, the greedy routing algorithm computes paths of expected length $O\left((\log D)^{6\beta} \log \delta + (\log n)^{\beta+1}\right)$ between any pair of nodes at mutual distance δ in the original network.

Using the low memory scheme, any synchronized n-node network, of expansion rate $O((\log n)^\beta)$ and diameter D, can be turned into a small world in at most $7n$ rounds, requiring $O(\Delta(\log n)^{3\beta+2} \log D)$ bits of memory in each node w.h.p., and with $O(nD)$ messages on expectation.

The proof of Theorem 3 is similar to the proof of Theorem 1, however, the expansion rate plays an important role in the variations of the main functions of the process (sample density, path length, memory required) and requires a precise new analysis.

From now, we consider a graph $G = (V, E)$ of expansion rate $c(\log n)^\beta$.

The next two lemma guarantee that the sampling phase has an analogous behaviour in a polylogarithmic expansion rate network than in a bounded growth network. This new analysis of the random sampling provides us the dependencies of the constants in terms of the expansion rate exponent β.

Lemma 7 *For all $i \in \{\lceil \log(2c(\log n)^{\beta+1}) \rceil + 2, \ldots, \lceil \log D \rceil\}$, we have, for all node $\mathbf{u} \in V$, $\Pr\{\mathbf{u} \in S_i\} = (2c(\log n)^{\beta+1})/b_{\mathbf{u}}(2^i)$.*

Proof. Setting $k_0 = \lceil \log(2c(\log n)^{\beta+1}) \rceil$ suffices to prove the result : the proof is the identical to Lemma 2 proof. □

Lemma 8 *The probability that each node $\mathbf{u}$ has a ith level leader for all levels $i \in \{\lceil \log(2c(\log n)^{\beta+1}) \rceil + 2, \ldots, \lceil \log D \rceil\}$ is greater than $1 - \frac{\log n}{n}$.*

Proof. Consider a node $\mathbf{u}$ and let $i \in \{\lceil \log(2c(\log n)^{\beta+1}) \rceil + 2, \ldots, \lceil \log D \rceil\}$. From Lemma 7, the probability that a node $\mathbf{v}$ belongs to S_i is $2c(\log n)^{\beta+1}/b_{\mathbf{v}}(2^i)$. By inclusion, any node $\mathbf{v}$ in $\mathcal{B}_{\mathbf{u}}(2^i)$ satisfies : $b_{\mathbf{v}}(2^i) \leq b_{\mathbf{u}}(2^{i+1})$. From the given expansion rate, we have $b_{\mathbf{u}}(2^{i+1}) \leq c(\log n)^\beta b_{\mathbf{u}}(2^i)$. The probability that no node of S_i lies in $\mathcal{B}_{\mathbf{u}}(2^i)$ is thus less than :

$$\left(1 - \frac{2 \log n}{b_{\mathbf{u}}(2^i)}\right)^{b_{\mathbf{u}}(2^i)} \leq \frac{1}{n^2}.$$

By the union bound, since there are n nodes and less than $\lceil \log D \rceil$ levels, the probability that this failure happens for at least one node is less than $\log D/n \leq \log n/n$, the result follows. □

Lemmas 7 and 8 enable us to analyze the greedy routing in the augmented network, and thus to prove its navigable small world status.

Theorem 4 *In any network of n nodes and of expansion rate $O((\log n)^\beta)$, augmented by our small world algorithm, the greedy algorithm computes, between any pair of nodes at mutual distance δ, a path of expected length $O\left((\log n)^{6\beta} \log D \log \delta\right)$.*

Proof. Let $\mathbf{s}$ and $\mathbf{t}$ the source and target, at mutual distance δ. Divide the execution of the greedy algorithm into $\log m$ phases : the algorithm is in phase j while the current distance to $\mathbf{t}$ belongs to $(2^{j-1}, 2^j]$, for all $j \in \{1, \log m\}$.

Assume the algorithm is in phase j and $\mathbf{u}$ is the current message holder. From lemma 10, for any node $\mathbf{v}$ at distance less than 2^{j-1} from $\mathbf{t}$, and at distance $X \geq 2c(\log n)^{\beta+1}$ from $\mathbf{u}$, $\Pr\{L(\mathbf{u}) = \mathbf{v}\}$ is greater than : $1/(c^2(\log n)^{2\beta+1} \log D \cdot b_{\mathbf{u}}(X))$. Note that, while $j > k_0 + 1$, the distance hypothesis are satisfied for all nodes $\mathbf{v}$ in $\mathcal{B}_{\mathbf{t}}(2^{j-2})$. Assume now $j > j_0$. For all nodes $\mathbf{v}$ in $\mathcal{B}_{\mathbf{t}}(2^{j-2})$, we have $X \leq 5 \cdot 2^{j-2}$. From the union bound on nodes of $\mathcal{B}_{\mathbf{t}}(2^{j-2})$, we obtain :

$$\Pr\{L(\mathbf{u}) \in \mathcal{B}_{\mathbf{t}}(2^{j-2})\} \geq \frac{1}{c^2(\log n)^{2\beta}} \frac{1}{\log D} \frac{b_{\mathbf{t}}(2^{j-2})}{b_{\mathbf{u}}(5 \cdot 2^{j-2})},$$

By inclusion, $b_{\mathbf{u}}(5 \cdot 2^{j-2}) \leq b_{\mathbf{t}}(10 \cdot 2^{j-2})$, and from the given expansion rate, $b_{\mathbf{t}}(10 \cdot 2^{j-2}) \leq c^4(\log n)^{4\beta} b_{\mathbf{t}}(2^{j-2})$. The probability to end the jth phase of the execution is thus greater than : $1/(c^6(\log n)^{6\beta} \log D)$ in each node during this phase. The expected number of steps in each phase $j > j_0$ is then less than $\left(c^6(\log n)^{6\beta}\right) \log D$. As soon as $j \leq j_0$, we can simply lower bound the remaining expected number of steps by 2^{j_0}. Finally, summing over the $\log \delta$ phases, the total expected path length is less than : $O\left((\log n)^{\beta+1}\right) + O\left((\log D)^{6\beta} \log \delta\right)$. □

It remains to analyze the consequences of the polylogarithmic expansion rate in terms of memory space and number of messages in the large and low memory settings of our algorithm. Note that the two implementation described in section 3 are unchanged, we only study the changes in performances.

Lemma 9 *W.h.p, each node $\mathbf{u}$ belongs to $O((\log n)^{3\beta+1})$ trees $\mathcal{T}$ rooted in sample nodes.*

Proof. Trees $\mathcal{T}$ to which $\mathbf{u}$ belongs are all the trees rooted in samples nodes of level i that belong to $\mathcal{B}_{\mathbf{u}}(2^{i+1})$, for all levels i. Consider a node $\mathbf{v} \in \mathcal{B}_{\mathbf{u}}(2^{i+1})$. From Lemma 7, $\Pr\{\mathbf{v} \in S_i\} = 2c(\log n)^{\beta+1}/b_{\mathbf{v}}(2^i)$. Since $\mathbf{v} \in \mathcal{B}_{\mathbf{u}}(2^{i+1})$, by inclusion, we have $b_{\mathbf{v}}(2^i) \leq b_{\mathbf{u}}(2^{i+2}) \leq c(\log n)^{\beta} b_{\mathbf{u}}(2^{i+1})$. From the expansion rate, $b_{\mathbf{v}}(2^i) \geq b_{\mathbf{v}}(2^{i+2})/(c^2(\log n)^{2\beta}) \geq b_{\mathbf{u}}(2^{i+1})/(c^2(\log n)^{2\beta})$. For each node $\mathbf{v}$ in $\mathcal{B}_{\mathbf{u}}(2^{i+1})$, let $X_{\mathbf{v}}$ the random variable equals to 1 if $\mathbf{v} \in S_i$, and 0 otherwise, and let $X = \sum_{\mathbf{v} \in \mathcal{B}_{\mathbf{u}}(2^{i+1})} X_{\mathbf{v}}$. From the above discussion, for all $\mathbf{v}$ in $\mathcal{B}_{\mathbf{u}}(2^{i+1})$, $\mathbb{E}[X_{\mathbf{v}}]$ belongs to $[2\log n/b_{\mathbf{u}}(2^{i+1}), 2c^3(\log n)^{3\beta+1}/b_{\mathbf{u}}(2^{i+1})]$. Thus $\mathbb{E}[X] \in [2\log n, 2c^3(\log n)^{3\beta+1}]$. Since variables $X_{\mathbf{v}}$ are independent, we can use Chernoff bound to upper bound the probability that X is greater than $4\mathbb{E}[X]$, we get :

$$\Pr\{X > 4\mathbb{E}[X]\} \leq [\frac{1}{e}(e/4)^4]^{\mathbb{E}[X]} \leq \frac{1}{e^{2\log n}} \leq \frac{1}{n^2}.$$

Since $\Pr\{X > 4\mathbb{E}[X]\} \geq \Pr\{X > 8c^3(\log n)^{3\beta+1}\}$, with probability greater than $1 - 1/n^2$, $\mathbf{u}$ is traversed by at most $8c^3(\log n)^{3\beta+1}$ trees rooted in sample nodes of level i. By the union bound, it occurs for all levels i with probability greater than $1 - \log D/n^2$, which gives a number of trees $O((\log n)^{3\beta+1})$. By the union bound again, this occurs for all nodes $\mathbf{u}$ with probability greater than $1 - \log n/n$. □

Lemma 10 *For all $\mathbf{v} \neq \mathbf{u}$ in V such that $d(\mathbf{u}, \mathbf{v}) \geq 2c(\log n)^{\beta+1}$, the probability that $\mathbf{v}$ is the long range contact of $\mathbf{u}$ is greater than*

$$\frac{1}{c^2(\log n)^{2\beta+1} \log D} \frac{1}{b_{\mathbf{u}}(d(\mathbf{u}, \mathbf{v}))}.$$

Proof. It suffices to replace the factor c^2 by $c^2(\log n)^\beta$ in Lemma 3 proof. □

Lemma 11 *The expected overall number of messages induced by the algorithm* Sample *is $O\left(n(\log n)^{\beta+1}(\log D)^3\right)$.*

Proof. Compared to the bounded growth networks version, the probability of being a ith level node is simply multiplied by $O((\log n)^{\beta+1})$, the result is immediate. □

Lemma 12 *W.h.p., the memory required by algorithm* Sample *in each node is* $O\left(\Delta(\log n)^{3\beta+1}\log D\right)$.

Proof. A node $\mathbf{u}$ has to store enough information for the execution of a DFS on each tree $\mathcal{T}_r$ that traverses it during the execution of Sample on r, which is $O(\Delta)$ for each tree. From lemma 9, w.h.p., there are at most $O((\log n)^{3\beta+1}\log D)$ such trees for each node $\mathbf{u}$, the result follows. □

We are now able to state our extended results in the high memory scheme.

Proposition 4 *The high memory scheme requires* $O(D)$ *rounds,* $O(n(\log n)^{\beta+1}(\log D)^3)$ *messages and a memory size* $O(n)$.

Proof. In this scheme, sample nodes of level $\lceil \log D \rceil$ may have to record up to n requests of links in the worst case where every node have chosen the $\lceil \log D \rceil$-th level, and then requires a linear memory size for these nodes. The time required to execute the long-range link request phase is at most four times the time required by the largest DFS, *i.e.* $O(D)$. As for the number of messages, each tree $\mathcal{T}_{\mathbf{s}_i}$ rooted in a sample node of level i induces at most $3b_{\mathbf{s}_i}(2^i)$ messages. To count the number of messages, one can consider that each sample node of level i induces at most $3b_{\mathbf{s}_i}(2^i)$ messages, and the other nodes do not induce messages. We deduce the overall expected number of messages through the execution : $\sum_{\mathbf{u}\in V}\sum_{i=k_0+2}^{\log D} 3b_{\mathbf{u}}(2^i)\Pr\{\mathbf{u}\in S_i\}$, which is $O(n(\log n)^{\beta+1}\log D)$, and is dominated by the number of messages due to Sample. □

To complete the proof of Theorem 3, we need to analyse the schedulling part of the low memory setting.

Proof of Theorem 3. Proposition 4 partially proves the Theorem. to complete the proof, we study the low memory setting.

The only real change concerns the addition of the scheduling. However, this algorithm is based upon a DFS traversal. For a tree $\mathcal{T}_r$, the traversal is done in $|\mathcal{T}_r|$ rounds and the first phase of the link requests are done within $|\mathcal{T}_r|$ plus the height of $\mathcal{T}_r$ rounds, which is at most $2|\mathcal{T}_r|$. As for the second phase, we have to add the extra delay due to the number of nodes of $\mathcal{T}_r$ that have chosen r as their leader. This phase is done in at most $3|\mathcal{T}_r|$ rounds, and the scheduling is done within $5|\mathcal{T}_r| \leq 5n$ rounds for any r.

We now deal with the total number of messages. Since a token makes twice a DFS traversal, there are $2b_{\mathbf{u}}(3\cdot 2^i)$ messages dedicated to the token. The number of requests from $\mathcal{T}_{\mathbf{u}}(2^i)$ that are sent to $\mathbf{u}$ is at most $b_{\mathbf{u}}(2^i)$. Each request produces at most $4\cdot 3\cdot 2^i$ messages. To count the number of messages, one can consider that each ith leader node of level i induces at most $O(2^i b_{\mathbf{u}}(2^i))$ messages, and the other nodes do not induce messages. Let $\mathbf{v}\in V$. The probability that $\mathbf{v}$ is a ith level leader is $2c(\log n)^{\beta+1}/b_{\mathbf{v}}(2^i)$, the expected number of messages per node is thus $\sum_{i\leq\lceil\log D\rceil} O(2^i)(\log n)^{\beta+1}) = O(D(\log n)^{\beta+1})$, yielding an expected overall number of messages $O(nD(\log n)^{\beta+1})$.

Concerning the memory size, from Lemma 9, each nodes can store all the informations on trees that traverse it with a memory of size $O(\Delta(\log n)^{3\beta+1})$. The memory required to store the requests before sending them is dominated by this latter memory size. □

5 Conclusion

We have introduced a fully distributed scheme which augments graphs of low expansion into navigable small world via the addition of one single link to each nodes. This scheme can be used in distributed virtual networks such as Peer-to-peer networks, where the addition of one link per node corresponds to the addition of one entry in each routing table. Our analysis focused on graphs of constant or polylogarithmic expansion rate, it can be extended to higher expansion rates, however, the performances will degrade.

An interesting extension of this work would be to capture the minimal dynamical mechanisms underlying the navigable small world property, and to separate it from the structural properties of the graph. Indeed, it is still an open question to determine whether any graph class can be augmented into a navigable small world.

Références

[1] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. In 16^{th} *Annual ACM Symposium on Parallel Algorithms and Architecture (SPAA)*, pages 20–24, 2004.

[2] Ittai Abraham and Dahlia Malkhi. Name independent routing for growth bounded networks. In 17^{th} *Annual ACM Symposium on Parallel Algorithms and Architecture (SPAA)*, pages 49–55, jul 2005.

[3] L. Barrière, P. Fraigniaud, E. Kranakis, and D. Krizanc. Efficient routing in networks with long range contacts. In *Proceedings of 15th International Symposium on Distributed Computing (DISC '01)*, volume 2180, pages 270–284, 2001.

[4] P. Bose and P. Morin. Online routing in triangulations. *SIAM J. Comput.*, 33(4) :937–951, 2004.

[5] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet : A distributed anonymous information storage and retrieval system. In *Proceedings of Designing Privacy Enhancing Technologies : Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, July 2000.

[6] P. Duchon, N. Hanusse, E. Lebhar, and N. Schabanel. Could any graph be turned into a small world ? *Theoretical Computer Science*, 2005.

[7] P. Fraigniaud. Greedy routing in tree-decomposed graphs : a new perspective on the small-world phenomenon. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA)*, 2005. To appear.

[8] P. Fraigniaud and P. Gauron. Brief announcement : an overview of the content-addressable network d2b. In *Proceedings of the ACM 22st Symposium on Principles of Distributed Computing (PODC)*, page 151, 2003.

[9] C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 210–219, 2001.

[10] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Journal of Distributed Computing*, 16 :111–120, May 2003. PODC 20-Year Special Issue.

[11] K. Horowitz and D. Malkhi. Estimating network size from local information. *Inf. Process. Lett.*, 88(5) :237–243, 2003.

[12] D. R. Karger and Matthias Ruhl. Finding nearest-neighbors in growth-restricted metrics. In *Proceedings of the 34th annual ACM symposium on Theory of computing (STOC)*, pages 741–750, 2002.

[13] J. Kleinberg. The Small-World Phenomenon : An Algorithmic Perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC)*, pages 163–170, 2000.

[14] J. Kleinberg. Small-world phenomena and the dynamics of information. In *Advances in Neural Information Processing Systems (NIPS) 14*, 2002.

[15] D. Liben-Nowell, J.Novak, R. Kumar, R. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Science of the USA*, 102(33) :11623–11628, 2005.

[16] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy : a scalable and dynamic emulation of the butterfly. In *Proceedings of the ACM 21st Symposium on Principles of Distributed Computing (PODC)*, pages 183–192, 2002.

[17] S. Milgram. The small world problem. *Psychology Today*, 61(1), 1967.

[18] A. Rowstron and P. Druschel. Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Int. Conf. on Distr. Syst. Platf.*, pages 329–350, 2001.

[19] A. Slivkins. Distance estimation and object location via rings of neighbors. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2005. To appear.

[20] 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 *SIGCOMM*, pages 149–160, 2001.

[21] M. Thorup and U. Zwick. Approximate distance oracles. *J. ACM*, 52(1) :1–24, 2005.