Distributed Social Graph Embedding

Anne-Marie Kermarrec, Vincent Leroy, Gilles Tredan

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Distributed Social Graph Embedding

Anne-Marie Kermarrec — Vincent Leroy — Gilles Trédan

N° 7327

Juin 2010

Domaine 1

ISSN 0249-6399 ISRN INRIA/RR--7327--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

Distributed Social Graph Embedding

Anne-Marie Kermarrec* , Vincent Leroy† , Gilles Trédan‡

Domaine : Mathématiques appliquées, calcul et simulation
Équipe-Projet ASAP

Rapport de recherche n° 7327 — Juin 2010 — 23 pages

Abstract: Content recommendation is becoming central in the Web 2.0 to leverage the growing information on users available today.

In this paper we propose a decentralized gossip-based algorithm called SoCS (Social Coordinate Systems) that achieves efficient *distributed social graph embedding* for content recommendation purposes. Social graph embedding embeds a graph into a d-dimensional Euclidean coordinate space. SoCS relies on a force-based graph embedding technique to extract communities from a graph. We explore here a distributed algorithm for it *(i)* scales to large dynamic graph, aggregating the computing power of individual nodes and, *(ii)* avoids a central entity controlling users sensitive data such as relations and preferences. We evaluate SoCS using two different force-based models and compare them in the context of a generated Kleinberg small-world topology. More specifically, we show that the SoCS graph embedding enables to clearly distinguish between short and long-range links. We also evaluate SoCS against a real DBLP data set, showing that removed links are correctly predicted. Finally, we show that our gossip-based algorithm is extremely resilient to dynamics.

Key-words: peer-to-peer, gossip, social networks

* INRIA Rennes - Bretagne Atlantique
† INSA de Rennes, UEB
‡ Deutsche Telekom

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Plongement distribué de graphe social

Résumé : La recommandation de contenu est devenue un élément central de l'Internet moderne pour permettre aux utilisateurs de naviguer parmi la masse d'information à leur disposition.

Dans ce papier, nous proposons un algorithme décentralisé appelé SoCS (Social Coordinate Systems) qui permet de plonger un graphe social dans un espace Euclidien afin d'effectuer des recommandations. Nous utilisons un algorithme de dessin de graphe basé sur un modèle de forces afin de mettre en évidence les communautés présentes dans un graphe. Nous présentons ici une solution décentralisée car cela *(i)* passe à l'échelle dans des systèmes dynamiques rassemblant plusieurs milliers d'utilisateurs, et *(ii)* permet d'éviter qu'une entité centrale ait accès à toutes les données privées des utilisateurs. Nous évaluons SoCS à travers deux modèles de force différents et les comparons dans le contexte d'une topologie petit-monde issue du modèle de Kleinberg. En particulier, nous montrons que SoCS permet de clairement différencier les liens courts des liens longs. Nous évaluons également SoCS sur des données réelles issues de DBLP, et montrons que des liens retirés peuvent être prédits grâce à notre algorithme. Enfin, nous montrons que notre algorithme est extrêmement résistant aux perturbations.

Mots-clés : pair-à-pair, algorithmes épidémiques, réseaux sociaux

1 Introduction

There has recently been an explosion of user-generated content in the Web 2.0. Users are now active players: they generate content, share it with others, annotate it, etc. This represents a huge opportunity to exploit the relations between users and content for content recommendation. For instance, detecting communities in a social network, can be used to suggest friends to users. At the era of the Web 2.0, the applications that could benefit from such recommendation systems are numerous. Yet, this requires to be able to extract relationships between users in a large and growing information universe. While some recommendation systems exist, they are mostly characterized by ad-hoc systems targeting specific applications (Amazon, NetFlix). In addition, these approaches are mostly centralized. In this work, we are interested in fully distributed approaches to provide effective tools for content recommendation.

Decentralized approaches have started to be considered to implement such recommendation systems for they are appealing for scalability and privacy reasons: *(i)* if each user (machine) is in charge of a part of the computation, the system is inherently scalable. This is crucial in systems where the number of users and data to handle increases by the minute; As a matter of fact, dealing with such a huge amount of information can be afforded today only by a handful of huge and powerful companies; *(ii)* In addition, user-centric recommendation systems do not necessarily need to have a global knowledge of the system, the data which is relevant to each user is in fact sufficient to provide good results, this typically fits well a distributed approach; *(iii)* Finally, at the time where the "Big brother is watching syndrome" is prominent and users are more and more reluctant to hand off their personal information (taste, request history, annotation) to large single companies for obvious privacy reasons [8], fully decentralized solutions represent a way out, where each user is in charge of her own information.

Graphs are the most natural way to represent relationships between pretty much everything on the Web today: users, computers, data, devices, etc. It is a well-known fact that most of social networks exhibit *small-world* properties. A small-world graph is characterized by a low diameter and significantly higher clustering coefficient than random graphs [15]). A graph may represent relations between users as in social networks, between pages as in the Web, between items as books in Amazon, between users and items as in NetFlix, etc. These graphs often exhibit, along with their small world properties, a community structure. The knowledge of the community structure is of prime interest for a recommendation systems, since links (and thus, mutual interests) within a community are far more likely to appear than links between two communities [4].

Yet, extracting the community structure from a sole graph is not straightforward. Detecting communities is a computationally expensive task that is hardly compatible with dynamic graphs. Moreover, the traditional definition of community, still under discussion, is very rigid: a node either belongs or does not belong to a community. Overlapping communities have only been only recently considered (see *e.g.* [22]). In this paper, we substitute the (tree-shaped) traditional community structure by a notion of community space, that we call *social* space. In this space, communities are no longer strictly defined as sets, but nodes from the same community belong to the same geographical area in

that space. This definition allows nodes in the social space, to be in the core or at the border of a community, as well as in multiple communities.

Following Noack's pioneering work [21], we believe that this notion of communities in a graph can be successfully extracted using graph embedding techniques preserving such graph community structure. This would enable to extract the relevant information from any graph to implement a recommendation system.

Contributions In this paper, we propose SoCS (Social Coordinate System), a fully distributed graph embedding system, which we believe has numerous applications for content recommendation and search. SoCS is based on a gossip protocol and does not require any node to have a global knowledge of the system. The problem can then be stated as follow: considering a graph G, we aim at embedding the graph in a d dimensional Euclidean host space: each node of the graph should be assigned coordinates in the host space, such that nodes belonging to the same community are mapped close to each other. The resulting host space is *social* as nodes not directly linked in the initial graph may appear next to each other in the resulting host space, reflecting hidden relationships, typically communities. Similarly, this defines a *social distance*, namely the Euclidean distance over the social space. Consider for example a social network graph, users (nodes) may then discover close neighbors in the host space and consider those as suggestions.

SoCS can be parametrized with any force-based model and we explore in this paper the application of several well-known energy models to provide a social graph embedding. Force-based techniques have been shown equivalent (under some conditions) to local non linear dimensionality reduction algorithms [3] that are commonly used in machine learning or proximity analysis. The importance of locality in these algorithms makes them good candidates for distribution. Therefore, the algorithm we propose in this paper can be seen as a distributed dimensionality reduction algorithm.

Our contributions are the following:

1. We propose a distributed graph embedding algorithm: SoCS. This algorithm is fully gossip-based and thus suited to dynamic networks. The embedding achieved by this algorithm respects the community structure of the input graph. We show how the output of this algorithm can be considered as a social space. For instance, when applied to a small world topology, it allows to differentiate between short and long-range links, which is known as a non-trivial problem [9].

2. We evaluate SoCS in the context of *(i)* a synthetic small-world Kleinberg topology and *(ii)* the DBLP data set and evaluate the relevance of SoCS for link prediction [18]. Secondly, our results show that SoCS is able to achieve social embedding: *(i)* it enables to clearly distinguish between short and long-range links in a small-world network and *(ii)* removed links are correctly predicted in a real DBLP data set. We believe that many content recommendation systems may leverage such a coordinate system.

3. Force based embedding algorithms rely on a force model. We inspect several variants of force models and provide empiric evidence that the state-of-the-art force model as presented by theoreticians (the LinLog model

[20]) is not always the model providing the best results in a practical setting. Additionally, we obtain the surprising result that distributed versions of our protocol, that solely rely on local knowledge, often exhibit better performance than their centralized counterparts. This demonstrates a connection with local non-linear reduction algorithms.

The rest of the paper is organized as follows. In Section 2, we present background knowledge about graph embedding and force-based layout. We present SoCS in Section 3 and show experimental results in Section 4. Finally, we review related work in Section 5 and conclude in Section 6.

2 Background: Force-based Graph Embedding

In this section, we provide the background on graph embedding required to understand our decentralized protocol. We first define graph embedding and present the two force-based models that we are considering in this work.

Graph Embedding Consider $G(V, E)$, an undirected graph of n nodes representing a distributed system ($n = |V|$). Let $\mathcal{P}$ be the *host* space and let $\dim \mathcal{P} = d$. A graph embedding is a function:

$$\begin{array}{rcl} f : V & \rightarrow & \mathcal{P} \\ i & \rightarrow & P_i \end{array}$$

We denote $d_G(i, j)$ the graph shortest path distance between nodes i and j, and $||\overrightarrow{P_iP_j}||$ the (Euclidean) distance between images for nodes i and j in $\mathcal{P}$ by the function f. In other words, $||\overrightarrow{P_iP_j}||$ is the distance between the projections of i and j in the host space.

Most of the works on graph embedding concentrate on minimizing the graph distance between two nodes and the distance of their respective images in the host space. Formally, the criterion with respect to this goal is captured by the so-called *distortion*: let f be an embedding of G into $\mathcal{P}$, and let $c \in]0, 1]$ be the distortion. We have:

$$\forall i, j \in V^2, d_G(i, j) \geq ||\overrightarrow{P_iP_j}|| \geq \frac{1}{c} d_G(i, j).$$

If $c = 1$, the embedding is isometric. It is important to note that there always exists an isometric embedding in a $(n - 1)$-dimensional host space [27]. The dimension d of the host space is an important parameter to consider for distortion. Minimizing the distortion is an important goal when the embedding aims at reflecting as closely as possible the graph distances. Yet, our goal to achieve a social embedding is to precisely have some distortion to distinguish edges between community. Therefore, in SoCS, we use low-dimensionnal host spaces ($d << n$): our goal is not to minimize the distortion, since an isometric embedding would not respect the community structure.

Force-based graph embedding There are several ways of achieving graph embedding. In this paper, we explore a subfamily of these techniques, namely the force-based embeddings (FBE), introduced by Eades [7]. Several FBE models have been proposed since. Force-based models achieve the embedding by

assigning forces to edges and nodes. Intuitively, edges can be seen as springs and nodes as electrically equally charged particles so that the graph simulates a physical system. We consider these models for they match well the social embedding we are targeting in this work.

FBEs are iterative algorithms and rely on two forces that define the attractions and repulsions that each node is subject to in the host space. Initially, each node is placed at random in the host space. At each iteration of the algorithm, the forces are applied to the nodes, pulling them closer to each other or pushing them further apart in the host space. The algorithm computes the sum of attraction and repulsion forces that applies to each node, and derives each node's next position. The algorithm stops when the system reaches an equilibrium. In practice, it stops when each iteration's impact is small enough (nodes moves are negligible). FBEs thus iteratively reaches an energy minimum.

In FBEs, *attractive* forces are always applied to a node by its *graph neighbors* whereas *repulsive* forces are applied by *all* other nodes. Figure 1 illustrates this principle: node a is attracted by b and c and repulsed by b, c, d. Moreover, following Noack's formalism [21], we define forces as proportional to some power of the distance between two images. More formally, the attraction forces applied to a node $i \in V$ are defined as

$$\overrightarrow{A_i} = \sum_{j \in V_i} ||\overrightarrow{P_iP_j}||^{fa} . \frac{\overrightarrow{P_iP_j}}{||\overrightarrow{P_iP_j}||},$$

and the repulsion forces applied to i as:

$$\overrightarrow{R_i} = -\sum_{j \in V} ||\overrightarrow{P_iP_j}||^{fr} . \frac{\overrightarrow{P_iP_j}}{||\overrightarrow{P_iP_j}||}.$$

Thus each couple (fa, fr), as respective parameters of the attraction and the repulsion force, defines a new force model. Note that only systems with $(fa > fr)$ produce finite distances (provided the graph is connected).

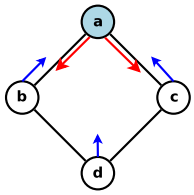


Figure 1: Illustration of FBE

In this paper, we study two common FBE models. The first one is Noack's *LinLog* [20], known as producing layouts that take into account the similarity of nodes. This is considered as the best force model to preserve the community structure of a graph. The second one is the carbon copy of physical Hooke's spring attraction and Coulomb's electrostatic repulsion forces (such as presented in [25]). We denote *HC* this model hereafter. In this widespread model, repulsion strength quickly decreases with distance. This fits the approximation we present in the next section. The following table describes the two forces parameters:

Force Name	*fa*	*fr*
LinLog	0	−1
HC	1	−2

3 Social Coordinate System (SoCS): distributed graph embedding

In this section, we describe the SoCS decentralized protocol, achieving a force-based embedding in a fully distributed manner. The novelty of SoCS is twofold: *(i)* Firstly we are considering a social embedding in which the distance in the host space should reflect some social proximity; *(ii)* SoCS is a fully decentralized system and relies on a gossip protocol to implement the embedding without relying on a central entity having the global graph knowledge. Instead, we assume that each node knows only a limited set of other nodes in the graph.

In this section we describe SoCS in detail. We first provide the rationale behind social embedding (Section 3.1) and present our system model (Section 3.2). We then describe how to compute repulsions in FBEs without any node having the global knowledge of the system (Section 3.3). Finally, we describe the SoCS gossip-based algorithm (Section 3.4).

3.1 Rationale

In order to illustrate the intuition behind social embedding, let us consider a graph example as shown on Algorithm 2. Assume that this graph expresses the *friendship* relation between people. Looking at this graph of relations from a's standpoint, d and f being both two hops away in the graph should be considered equally friends to a if there were no other information in the graph. Yet, by looking at the other edges in the graph it is easy to observe that it is very likely that d is *closer* to a since they have two friends in common, namely c and b. This could result in d having twice more chances of being invited to a's birthday than f.

Such social information is typically what SoCS aims at capturing in the resulting social embedding: this is achieved since the attractive forces applied between c and d bring a and d closer in the social space. In addition, due to d's repulsive forces, c and b are closer to a than e. As a result, the list of friends of a in the resulting space is then, ordered from the closest to the farthest: b and c, e, d, f.

These results can be interpreted as follows: since (a, b, d, c) is a rectangle, it has more *cohesion* than a single line, and members of that structure get close positions. The generalization of this intuition is that nodes within the same

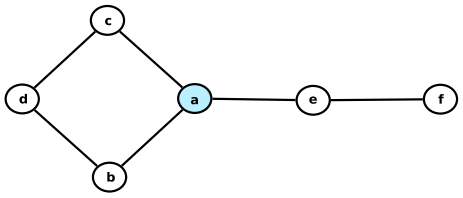


Figure 2: Example of a friendship social network

community get closer positions than nodes from different communities. This is exactly the approach advocated by Noack, that recently bridged community detection and force-based graph embedding [21]. Indeed, the notion of community can be extracted from the graph. A community is a part of a graph that contains a greater link density. Thus, link prediction algorithms are empowered by the knowledge of communities, as shown by Clauset *et al.* in [4]. SoCS achieves such a social embedding, capturing, as the evaluation will confirm, the notion of clusters and communities by placing nodes of a community close to each other in the host space. Thus the Euclidean distance can be directly leveraged to predict links between nodes.

3.2 System model

SoCS is a peer-to-peer (P2P) algorithm that embeds a social graph and operates on a P2P overlay network. We consider a system of n nodes and assume a one-to-one mapping between the nodes of the social graph and the machines connected to the P2P network, hence we refer as node both to the machine connected to the network and to the logical entity in the social graph. The social graph is an input in SoCS, so each node is aware of the other nodes it is connected to in the graph. We call this set of nodes the *Graph view*, or graph neighbors, of the node. In a dynamic social graph, the graph view can vary during the execution: the graph view only contains nodes that are connected to the network. To join an existing SoCS P2P network, a node contacts its graph neighbors to establish connexions. Traditionally, in FBEs, each node is assigned a random initial position. Since SoCS considers the case of persistent networks (as opposed to a one-shot embedding), a node joining the network can leverage the position of the nodes in its graph view to infer a starting position close to them, at the barycenter of their coordinates. In SoCS and more generally in a graph embedding system, a node, in order to compute its position, needs to compute the sum of attractions and repulsions that applies to it. As explained in Section 2, attractions take into account the nodes in the graph view. Since a SoCS node is directly connected to all the nodes in its graph view, computing the attractions is straightforward. We will detail, in the following, how the repulsions are approximated and computed.

3.3 Local repulsions

Computing the repulsion based on all nodes is both extremely costly in a large system and hard to achieve in a decentralized way. Instead, we only consider in SoCS the repulsion forces of the *close* nodes in the social space. This optimization was first introduced by Fruchterman and Reingold [10] to optimize the time complexity of the algorithm.

Considering only the local nodes in the computation of the repulsion is relevant for several reasons. Firstly, in SoCS, we are targeting content recommendation. Therefore considering for a node, only a local portion of the graph is enough both for attractions and repulsions. In addition, in the HC model, the repulsion force between two nodes is inversely proportional to the square of the distance between the nodes. Therefore the repulsion becomes negligible if two

nodes are distant. Let $B(i,k)$ to be the set of i's k closest neighbors in P:

$$j \in B(i,k) \Leftrightarrow |\{v \in V, ||\overrightarrow{P_iP_v}|| \leq ||\overrightarrow{P_iP_j}||\}| \leq k.$$

The repulsion thus becomes

$$\overrightarrow{R_i} = -\sum_{j \in B(i,k)} ||\overrightarrow{P_iP_j}||^{fr} . \frac{\overrightarrow{P_iP_j}}{||\overrightarrow{P_iP_j}||}.$$

Therefore, in SoCS, each node needs to compute its repulsions based on its local neighborhood. The benefit of this optimization is that each node needs to know its closest neighbors (but not only the direct ones) in the social space. The size of this neighborhood, *i.e.* the number of repulsion is represented by the parameter r. This is precisely what the SoCS algorithm achieves: it provides, using a gossip protocol, each node with its r closest neighbors in a fully decentralized way.

3.4 The SoCS decentralized algorithm

We are considering a fully decentralized system in which each node knows only a portion of the network. The nodes that each node knows about represent its neighbors. Maintaining such networks in a fully decentralized way can be achieved using gossip protocols. In such protocols, each node periodically exchange some information about its neighbors with one of its neighbors, called the gossip target. The nodes then update their list of neighbors, taking into account their current view and the new received information. The way nodes update their neighbors depends on the targeted structure of the network. Gossip protocols can be used to build and maintained connected fully unstructured networks as well structured ones.

In this paper we consider two gossip protocols simultaneously running: The *Random Peer Sampling* (RPS) [13] and the *Neighbor Peer Sampling* (NPS) [26]. To this end, each node maintains two data structures: a *Random view* composed of a random sample of c nodes, provided by the RPS and a *Neighbor view*, composed of the r nodes which are the closest in the social space, maintained by the NPS.

The *Random Peer Sampling* service provides each node with a continuously changing random sample of the nodes if the network. The resulting overlay has the properties of a random graph and ensures that the network remains connected. The *Neighbor Peer Sampling* service builds a view of the closest peers according to a given metric. In SoCS, the metric is the Euclidean distance in the social space. The NPS protocol fills up the Neighbor view of each node by gossiping with other nodes, but also by leveraging the information from the RPS. The later is crucial when the node joins the network since it speeds up the convergence a lot. In SoCS, each node runs two gossip protocols: (i) a RPS service and (ii) a NPS protocol.

Algorithm 1 is the decentralised SoCS algorithm. The initialisation is done lines $1-3$. In a static setting (G does not change over time), we assume that nodes are provided with a function `Converged()` that stops the execution of the protocol on a given node. For instance, a node can stop when it did not moved over the last couple of rounds. In a dynamic setting, `Converged()` is always `false`.

Algorithm 1 Graph embedding algorithm, code for Node i

1: $P_i \leftarrow$ `GetBarycenter(GetGraph())`
2: $v_{prev} = 1$
3: $\overrightarrow{d}_{prev} \leftarrow$ new `RandomDirection()`
4: **while not** `Converged()` **do** ▷ Execute a cycle
5: $\quad\overrightarrow{d} \leftarrow \overrightarrow{0}$
6: $\quad$**for** $j \in$ `GetGraph()` **do** ▷ Compute attractions
7: $\quad\quad\overrightarrow{d} \leftarrow \overrightarrow{d} + ||\overrightarrow{P_iP_j}||^{fa}.\frac{\overrightarrow{P_iP_j}}{||\overrightarrow{P_iP_j}||}$
8: $\quad$**end for**
9: $\quad$**for** $j \in$ `GetNeighbors()` **do** ▷ Compute repulsions
10: $\quad\quad\overrightarrow{d} \leftarrow \overrightarrow{d} - ||\overrightarrow{P_iP_j}||^{fr}.\frac{\overrightarrow{P_iP_j}}{||\overrightarrow{P_iP_j}||}$
11: $\quad$**end for**
▷ Compute speed
12: $\quad\overrightarrow{d} \leftarrow \frac{\overrightarrow{d}}{||\overrightarrow{d}||}$
13: $\quad v = (\frac{\cos(\overrightarrow{d}, \overrightarrow{d}_{prev})}{2} + 1)v_{prev}$
▷ Move
14: $\quad\overrightarrow{d}_{prev} = \overrightarrow{d}$
15: $\quad v_{prev} = v$
16: $\quad P_i = P_i + v.\overrightarrow{d}$
17: **end while**

Each node retrieves its neighbors view by calling `GetNeighbors()` and its graph view by calling `GetGraph()`. Note that a node can be in both sets, and that the neighbors set may only contain node positions instead of node identities. Since these two sets are the only knowledge of each node, we believe a that a privacy-preserving protocol could be derived from SoCS.

Instead of computing directly a new node position, the presented algorithm computes a couple (speed, direction) toward the ideal position (lines $12 - 13$). This improvement, also used by [10], allows a faster convergence by speeding up nodes in the earlier steps while avoiding oscillation during the late ones.

Two key parameters allow to tune the algorithm. The first key parameter is the number of neighbors considered for computing the repulsions, r. The second key parameter is d, the dimension of the host space. The impact of both parameters is detailed in Section 4.

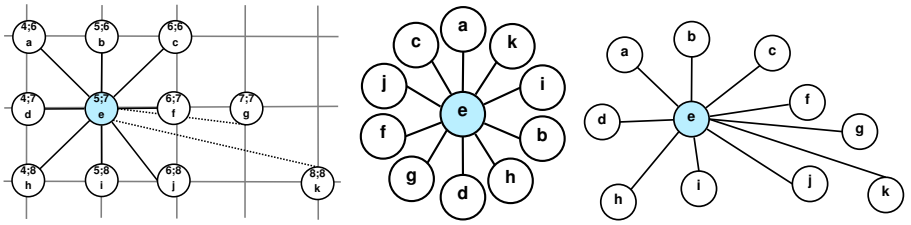
4 Experimental evaluation

We evaluate the quality of the embedding created by SoCS, against two graphs under both the HC and the LinLog energy models. The first graph we consider is a synthetic small-world graph ($1,024$ nodes) generated from a grid topology. We consider the original grid from which we generate the graph as the target semantic space. By comparing the coordinates assigned by SoCS to the original coordinates of nodes on the grid, we thus can precisely analyse the quality of the SoCS coordinate assignment. More precisely, we use this synthetic topology

to compare LinLog and HC force models, and to study the resilience of SoCS to dynamics. Note that the algorithm does not make any assumption of the nature of the graph. In the second set of experiments, we consider a real topology: a DBLP co-authors graph. Using SoCS to determine social coordinates of the authors, we then use the social distance as a link predictor.

4.1 Generated data: Small-world network

4.1.1 Setup



(a) Grid coordinate space (generated) (b) Social graph (SoCS input) (c) SoCS coordinate space (SoCS output)

Figure 3: Generating the small-world grid: node e (5;7) and its graph neighbors

In order to evaluate SoCS against objective criteria, we first experiment it in a controlled environment. As previously mentioned, most of the graphs representing social and semantic relations in Web 2.0 exhibit a small-world topology. Small-world graphs are characterized by a high clustering coefficient and a small diameter and have been widely studied from a graph perspective. Therefore we generated such a small-world graph in this section. We followed the methodology described below:

1. Firstly, we generate a 2D grid topology according to the Kleinberg model in the Euclidean plane [15]. Each node is assigned to *grid* coordinates (Figure 3a);

2. Secondly, we extract the graph from this initial coordinate space (Figure 3b);

3. Finally, we run SoCS with this graph as an input and compare the coordinates assigned by SoCS to the initial grid coordinates (Figure 3c).

Kleinberg small-world We consider a fully populated square grid of 1024 nodes in a 2-dimensional Euclidean space as shown on Figure 3a. Each node n (a point in the grid) has two types of links: (i) it is connected to the closest nodes on the grid (short-range links) as shown Figure 3a. In most cases, nodes have thus 8 short-range graph neighbors; (ii) Following the model defined by Kleinberg [15], each node n also draws one long-range link. The second end of this link is chosen with a probability proportional to $1/d(n,v)^2$, where $d(n,v)$ is the distance[1] between a potential candidate v and n. Intuitively the closer v to n, the higher its probability to be picked as a long-range graph neighbor.

[1]For the sake of uniformity across experiments, we used an Euclidean distance, whereas [15] used the Manhattan distabce. Yet, this does not impact the results.

This distribution of long-range links is justified by the properties of the greedy routing associated, but this is not a property we are using in this paper.

For the sake of our experiment, we assume this grid to be a *social space* such that the Euclidean distance in the Grid reflects the social distance between the nodes. The presence of long-range links may be interpreted as connections between communities. Additionally, it is important to notice that this small-world model does not contains communities in its traditional sense: because of the continuity of the grid space, a community detection algorithm would behave poorly there. However, we hereafter show that the local aspect of short-range links is adequately captured by SoCS.

Extracting the graph from this Euclidean space, we obtain a small-world topology from the set of links generated on this grid (Figure 3b). Since the graph is not oriented, each node has at least one long link and on average two. We then simulate the distributed system on this graph: each of the $1,024$ nodes is provided only with the list of its graph neighbors and starts an instance of the SoCS algorithm. Thus, SoCS runs on a binary graph and is not provided with any further information (such as the nature of the links, the number of short-range links, or the small-world nature of the graph).

4.1.2 Quality of SoCS embedding in a static setting

In this set of experiments, the graph is static. We evaluate the quality of the embedding generated by SoCS after it has converged. We use three different metrics: *(i)* a global Person coefficient; *(ii)* a local Pearson coefficient; and *(iii)* a link classification metric. We will explain each in detail below.

SoCS distance vs Grid distance: Global measure Our first set of experiment aims at evaluating the coordinates assigned by SoCS against the initial ones in the Grid from a global point of view. A direct comparison of both coordinates is not possible, since the coordinates produced by SoCS and the coordinates in the original grid may differ with respect to scale and orientation. However, if SoCS is accurate, then the SoCS distance between any pair of nodes is correlated to their original distance on the grid. We measure this correlation using the Pearson coefficient.

Figure 4 plots this metric as a function of d, the number of dimensions of the host space, for different sets of parameters. More specifically, we run SoCS using both the HC and the LinLog models for different values of r (1023 (all), 100, 20). All pairs of nodes are taken into account in the measure.

We observe that the LinLog force model produces coordinates highly correlated with the initial grid distances. When r is set to $1,023$, *i.e* all the repulsion are taken into account, the Pearson coefficient is equal to 0.95 regardless of the number of dimensions. This shows that the SoCS distributed algorithm associated to the LinLog model is able to capture the nature of the graph, by closely matching the initial coordinates. The HC model generates worse results, but the correlation always remains above 0.85 when all the repulsion are taken into account ($r = 1,023$).

Yet, in a large-scale distributed system, considering all the other nodes by setting r to the number of nodes is too time and bandwidth consuming. Reducing the value of r consists in considering only the nodes whose SoCS coordinates are the closest. In this case, we observe a drop in precision in both the HC and

the LinLog models. More generally, we observe from the figure that as soon as r is set to a sufficiently high value, the number of dimensions does not have a large impact. When the value of r decreases, a high number of dimensions slightly decreases the quality of the results.

SoCS distance vs Grid distance: Local measure While the previous paragraph examines the distances between all node pairs, we now focus on nodes that are connected (graph neighbors) in the graph. This assesses the ability of SoCS to provide a **given node** with its relative distance to any of its graph neighbors.

To this end, we consider each node independently and examine the length of its edges (*i.e.* the distances to its graph neighbors). As explained in Section 4.1.1, the graph neighbors lie at different social distances. Indeed, by construction, each node has short-range and long-range links. Yet, the input graph does not provide directly this information that differentiates them. Nodes connected through short-range links are at Grid distance 1 or $\sqrt{2}$ (see Figure 3a), and nodes connected through long-range links are more distant.

Figure 5 shows the results using a local version of the Pearson correlation between the SoCS and the Grid distances. More specifically, for each node in the graph, we compute the Pearson correlation between the SoCS distance and the Grid distances to all its graph neighbors. Then, this Pearson correlation value is averaged on all the nodes. We call this measure a local Pearson correlation, for it represents the local vision of each node in the network. This shows how well a given node is able to compare its social distance to its graph neighbors, without having any other knowledge. Interestingly enough, the results are fairly

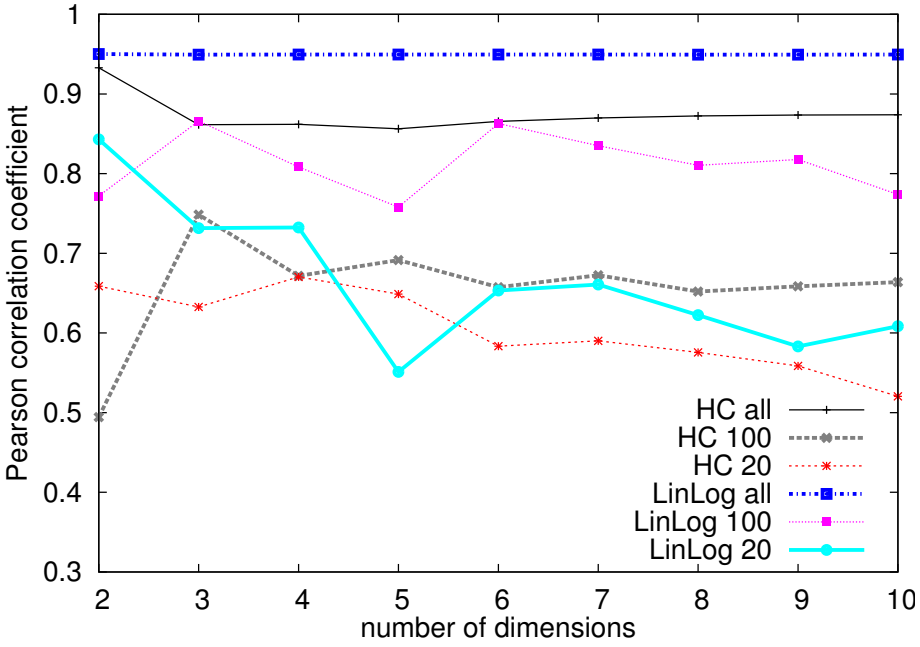


Figure 4: Global Pearson correlation between SoCS distance and grid distance

different from the previous ones. The figure shows the results using the same parameters values as in Figure 4.

Using SoCS with the LinLog force model, the best correlation is obtained with $r = 20$. Again, configurations with less repulsions are more sensitive to the number of dimensions, the performance slightly decreases when the number of dimensions increases. On the contrary, the correlation increases with the number of dimensions in the HC force model. In that case, the best results are obtained with $r = 100$. Clearly, SoCS, with both the LinLog and the HC force models produces a high local correlation between the SoCS length and the social length of the graph edges. Nevertheless, the configurations that maximize this performance are different from the ones that produce the best correlations between any two pairs of nodes.

To corroborate the previous results, we conducted an another set of experiments and consider the measure of *edges classification*, a problem introduced by Kleinberg in [16] (problem 9): Given a small world graph, is it possible to differentiate the short-range links from the long-range ones? This also measures the ability of SoCS to provide a node with a way to compare its distance to its neighbors in the graph. Looking at this under this angle matches some theoretical approaches that have been designed to solve this problem. For instance, in [9], the number of triangles formed by edges between nodes is used to decide whether those nodes are short or long-range neighbors.

Typically, a graph embedding system able to detect communities in a social graph could be used to distinguish such links. We believe that SoCS brings a new experimental approach to this problem. To this end, each node ranks its graph neighbors (Figure 3c), from the closest to the most distant using the

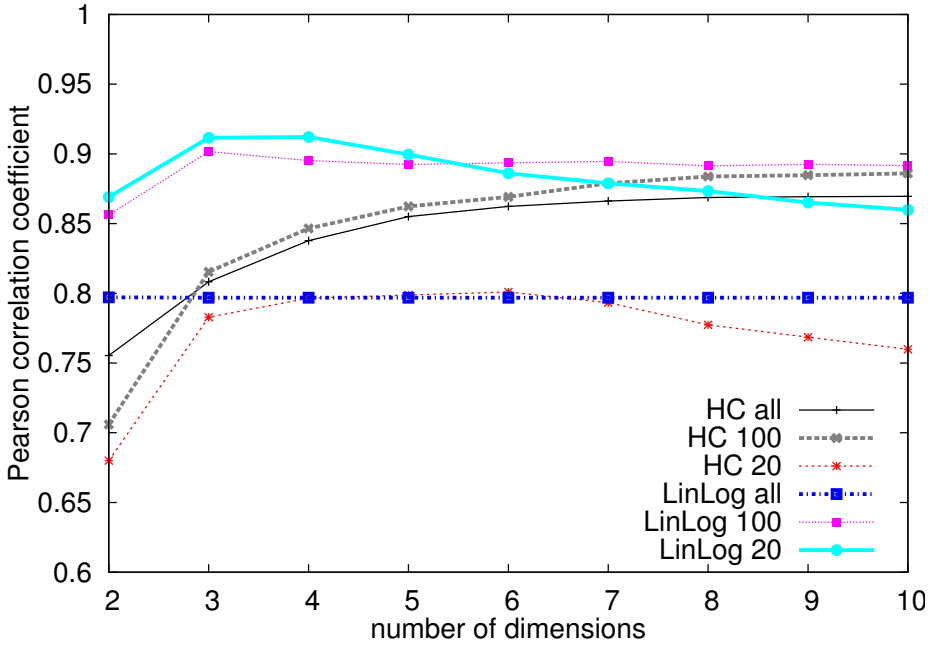


Figure 5: Local Pearson correlation between SoCS distance and social distance (graph neighbors)

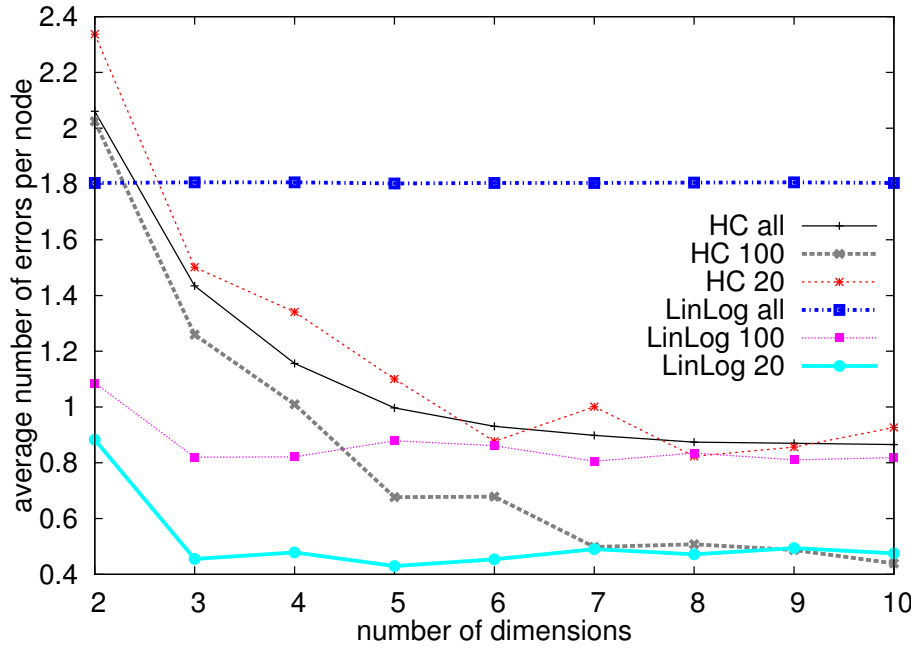


Figure 6: Number of errors in short/long links classification

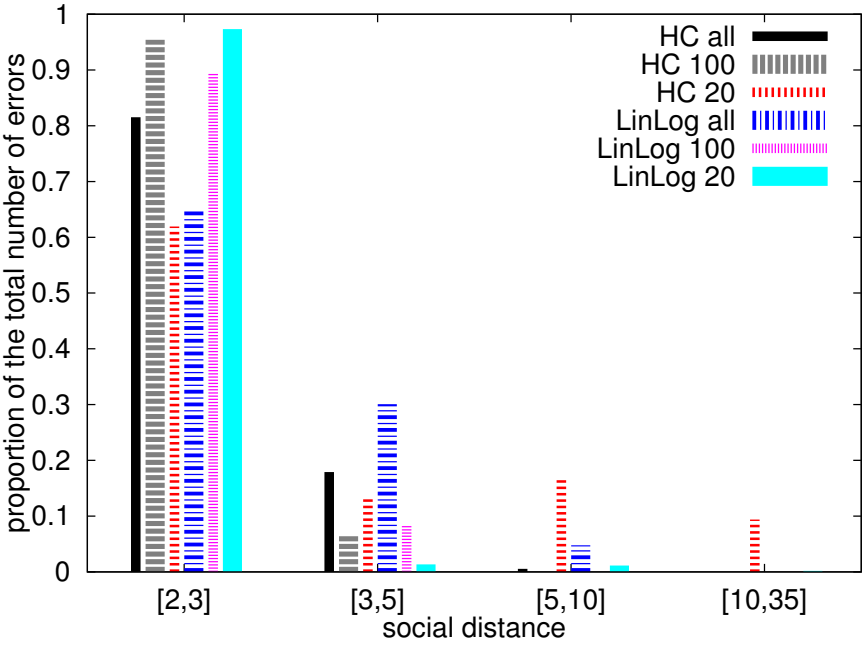


Figure 7: Distribution of the real grid length of misclassified long-range links

SoCS coordinates. We compare the outcome of the ranking with the initial Grid (Figure 3a). The number of errors in the ranking reflects the quality of the SoCS embedding. If all short-range links of a node are shorter using SoCS distance than the long-range ones, there is no error. To measure the number of errors

on a per-node basis, we sum for each long-link the number of short-links that are longer according to the SoCS coordinates. This metric can be seen as the number of bubble-sort operations that would be necessary to obtain a correct ranking.

Figure 6 represents the average number of ranking errors per node. These results confirm the previous ones: a small value of r improves the classification accuracy. The best results are obtained with $r = 20$ with the LinLog repulsions model and $r = 100$ with the HC model. This shows that this problematic fits very well a distributed setting in which only local knowledge is available. Increasing the number of dimensions also improves the quality of the ranking. Furthermore, Figure 6 plots the original length (in the grid) of misclassified links. It shows that most of them consist in inverting a "short" long-range link (social length ≈ 2) with a "long" short-range link (social length $= \sqrt{2}$). For instance, using HC with $r = 100$, more than 95% of misclassified links have a length smaller than 3. In other words, more than 95% of the errors are due to ranking long-range neighboring nodes that are short-range graph neighbors of at least one short-range neighbor of the originating node.

The results here confirm that SoCS is able to distinguish short-range links from long-range ones. In addition, when errors occur, they concern links whose social length is, in fact, very similar. Therefore the SoCS errors do not lead to misinterpretations socially-wise.

4.1.3 SoCS under dynamics

So far, we have considered a static system. In this section we evaluate SoCs in a dynamic environement where users may join and leave the system (aka churn), creating perturbations. SoCS relies on gossip protocols to maintain the network knowledge of each node. These algorithms have been extensively studied within different churn conditions and have been shown to be very resilient, even in the case of massive failures. In SoCS, the arrival or departure of a node modifies the underlying social graph. Therefore, the SoCS coordinates should be updated in order to reflect these modifications. In this section, we study the quality of the SoCS coordinates during different churn conditions. We rely on the link classification errors measure to illustrate this.

Figure 8 presents the average number of errors per node during a "cold start". The $1,024$ nodes of the small-world grid simultaneously join the network, without running any intermediate SoCS coordinate adjustment. We study the convergence speed of SoCS. The HC force model converges faster than the LinLog model for a fixed value of r. Indeed, when all the repulsions are taken into account, the number of errors in the HC model has converged after 8 cycles only, while it takes 25 cycles to reach the same state with the LinLog model. We also observe that for both force models, the convergence time decreases with r.

Figure 9 presents the same measure for another churn scenario. In this case, we divided the grid in 2 parts, selecting alternate nodes on each line, so that only the diagonal short links are present in each part. We run SoCS on the 512 nodes of the first part until the system has converged, and then add all the remaining nodes at once. Since the additionnal nodes join an existing network, they do not start at random coordinates, but leverage the position of their graph neighbors to obtain an initial placement, as explained in Section 3. First, we observe that SoCS coordinates are hardly affected by such perturbations. This

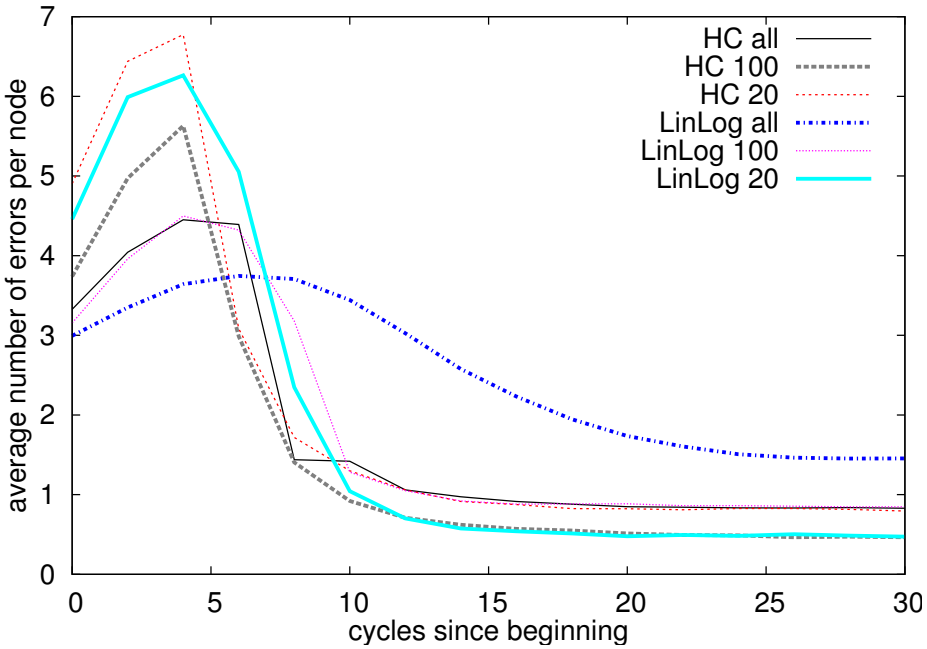


Figure 8: SW grid: classification errors from random start

is due to the fact that nodes present in the system from the beginning have already been assigned correct positions. Thus, as joining nodes initialise their SoCS coordinates from their graph neighbors, the number of errors per node remains low. Here also HC outperforms Linlog. The convergence is much faster with the first force model. As expected, a low value of r comes with an increased convergence speed.

Finally, we consider the case of regular random churn: nodes join and leave the network at each cycle. Due to space constraints, we do not display the results in this paper, but the conclusions drawn from this experiment are similar to the previous ones. HC stabilizes faster than LinLog, hence sustaining more churn without performance loss. When r is low, each node is only impacted by the churn that takes place in its close social neighborhood, so the resilience is higher.

These experiments show that the HC force model has the advantage of a faster convergence speed. This is easily explained by the impact of the repulsions: their weight is much lower in the HC model ($fr = -2$). Taking into account only the closest nodes for the repulsions, *i.e.* selecting a low value of r, also increases the convergence speed as it removes perturbations caused by the repulsions from distant nodes. These experiments show that SoCS performs very well as a distributed system since the configurations that converge the fastest and have a reasonable network cost are also the ones that obtain very good results at representing close social neighbors.

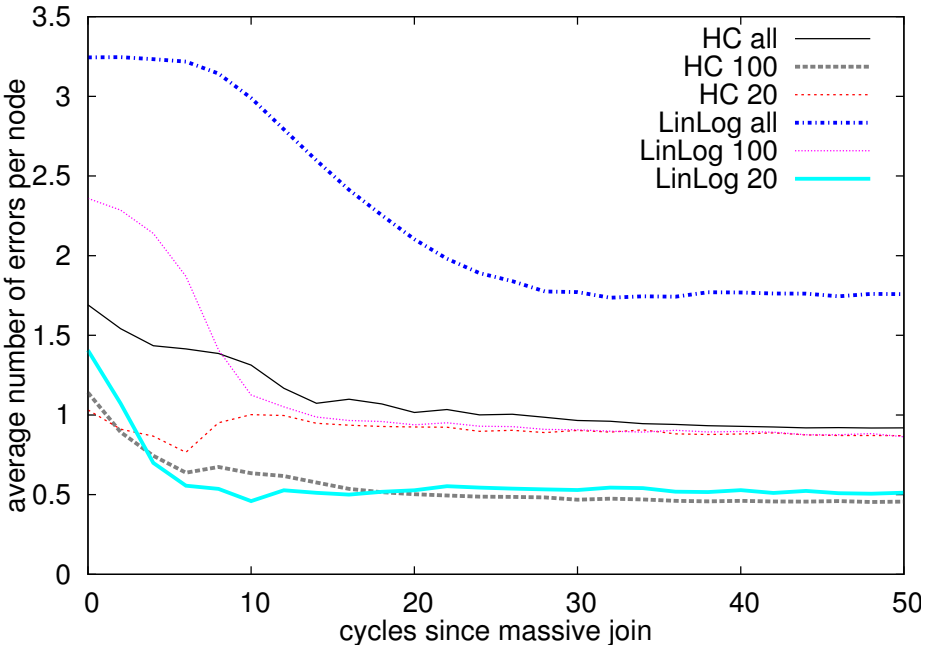


Figure 9: SW grid: classification errors with massive churn

4.2 Real data: DBLP dataset

4.2.1 Setup

In this section, we present the results on experiments conducted on a real data set, namely the DBLP dataset[2] from which we generate a co-author graph. We consider the graph in which all the authors having at least one publication in common are connected through an edge. In order to limit the size of the graph, we only consider the 27,905 authors who have at least 20 publications. We use the biggest connected component: 27,680 nodes connected by 211,078 edges. Note that the edges are not weighted and the information of the number of papers co-authored between two authors is ignored in this experiment.

4.2.2 SoCS embedding quality: Link prediction

As opposed to our previous dataset, we do not have baseline information to compare with. The DBLP graph has small-world characteristics, but we do not have any knowledge of the social distances between nodes. Therefore in order to assess the quality of the SoCS embedding, we evaluate its ability to correctly predict links (edges) in the graph. The link prediction problem [18] consists in finding the links that are likely to appear in the future of a social network. To this end, we select a subset of the edges in the DBLP graph (1% , 2,110 edges), which we call H, at random. We evaluate the performance of SoCS in the link prediction problem by trying to predict the edges of H from the DBLP graph where all the edges in H have been removed. Note that H has been selected

[2]http://dblp.uni-trier.de/xml/dblp.xml.gz

so that the graph remain connected without this set of edges. We use ROC curves [23] to display the results.

We apply the following strategy to predict links between authors in the graph: For each node, we predict a link to the closest neighbors in the SoCS coordinates space, then to the second closest, and so on. The order of the prediction takes into account the rank of the prediction only: we predict the closest link of each node, then the second closest link of each node, and so on.

Figure 10 displays the results of this prediction when SoCS uses a $d = 20$ dimensions. The results confirm the previous observations on the small-world grid. Indeed, to accurately predict links between nodes, a good representation of the close social neighborhood is required. Thus, the configurations that perform well here are similar to the ones of the link-classification problem (Section 4.1.2). A small number of repulsions helps preserving a good representation of the neighborhood by removing noise from distant nodes. Indeed, the quality of the link-prediction increases when moving from a all-repulsions configuration to $r = 100$ for both HC and LinLog. The LinLog energy model does not perform as well as the HC layout in this experiment, yet the results are very satisfactory. LinLog shows its limits in complex graphs, while HC is able to generate a social space with a high number of dimensions and gets better results.

For the purpose of comparison, we run two other sets of experiments and predict links using two other approaches. In the first one, the links are predicted between nodes based on their graph shortest path distance. As shown on Figure 10, this technique leads to poor results, and shows that SoCS is able to go beyond this simple metric and captures more precise connectivity properties in the graph. The second plot uses first the shortest path between nodes, and then, for nodes at distance 2, ranks them depending on the number of graph

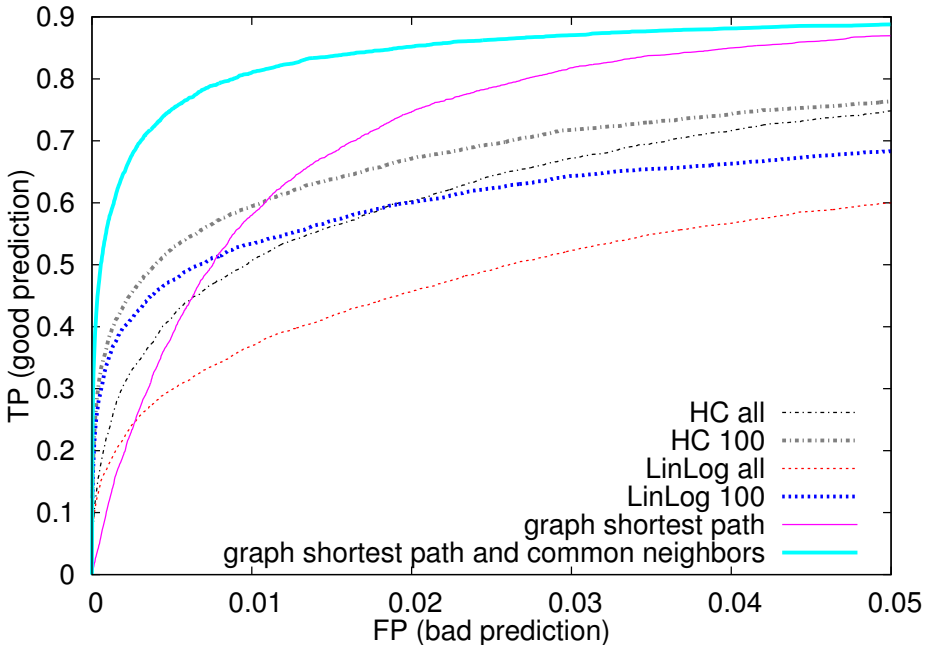


Figure 10: DBLP link prediction

neighbors in common. This method gives very good results and outperforms SoCS. However, it uses much more information than SoCS, since nodes only communicate with their graph neighbors in SoCS and never compute shortest paths. Given the expansion of modern social graph, computing and maintaining the shortest path distance to other nodes in a dynamic graph is very costly and does not scale to large graphs. The common neighbors measure has been shown to perform very well in many social graphs [18], but we believe that the setup of this experiment largely favors the common neighbors metric. Indeed, scientific publications often have more than 2 authors, which translates into a highly redundant common neighbors set.

5 Related Work

Graph embedding is a mathematical problem that has been receiving a lot of attention for it is a generic problem as explained by Linial, *et al.* [19]:

> Many problems concern either directly or implicitly the distance, or *metric* on the vertices of a graph. It is, therefore, natural to look for connections between such questions and classical geometric structures.

Most of the works in this area extend the results of Bourgain [2] that proved the existence of an embedding of any n-point metric in an $O(\log n)$-dimensional Euclidean space with a logarithmic distortion. They then explore how to exploit these low-distortion mappings to find efficient approximations for various problems such as k-commodity flow [1], or clustering (see Duda and Hart [6], chap. 6). The reader may find in [19] an extensive study of the important results and applications of graph embeddings.

Another active and related branch of graph embedding is the graph drawing problem. In this context, the host dimension is always 2 or 3 [25, 17, 11]. The objective is here to produce results that are visually meaningful or to match certain applications requirements such as minimising the number of crossing when integrating electrical components on a board.

In the sequel, we survey related works in distributed graph embedding, not specifically targeting content recommendation, and works related to content recommendation typically non-linear dimensionality reduction.

5.1 Distributing Graph embedding algorithms

To the best of our knowledge, this paper presents the first distributed graph embedding application. However, it is important to consider that some older works closely relate to embedding graphs in a distributed fashion.

As opposed to FBE, another approach to graph embedding relies on the graph spectral properties. The basic idea is to use the Eigenvectors associated to the k largest Eigenvalues of a graph's Laplacian matrix to embed the graph in a k-dimensional host space. This technique was introduced by Hall [12] and is extensively described in [17]. Recently, Kempe and McSherry [14] provided an algorithm to compute the k first Eigenvectors of the adjency matrix of a graph, based on decentralized orthogonal iteration. The number of steps the algorithm has to run to achieve an error ϵ is $O(\log^2(\frac{n}{\epsilon}).\tau_{mix}(G))$, where $\tau_{mix}(G)$ denotes

the graph G mixing time. The algorithm thus presents attractive costs for fast mixing graphs. However, the application of these works to graph embedding is not straightforward, and the stability of this algorithm when facing churn or dynamicity of the graph has yet to be studied.

Assigning Internet coordinates to nodes in a fully distributed way is a problem that received a lot of attention. One of the most well known system is Vivaldi [5], a distributed protocol that aims to predict Round Trip Times (RTT) between nodes on the Internet. The system is modelled as a valued graph where vertices are nodes and edges are measured RTT values. In a nutshell, a distributed spring embedding algorithm is used to embed the graph in a low (2-3) dimensional space. The idea is that even though each Internet node maintains an up-to-date RTT to some Internet nodes, the node's distance in the host space efficiently approximates the RTT between them. The main difference with our paper is that the embedding aims to be as isometric as possible.

5.2 Non linear dimensionality reduction

Another class of works that relate to SoCS is dimensionality reduction (DR). DR consists in discovering close items from a set items described by highly dimensional data. These methods are heavily used in machine learning and recommendation systems. Finding relationships between the describing variables that are not linearly correlated led people to consider variables correlated in a manifold (such as the distances between cities of the Earth globe). The goal of the non-linear DR algorithm is to flatten this data into a Euclidean space (a world map). The approach followed by these methods (*e.g.* KPCA, LMDS [3], LLE [24]) is to achieve DR while preserving the "local structure" of the data.

Recently, Chen and Buja [3] pointed out the close link between graph drawing and non linear DR. Their conclusion is that graph drawing, under certain circumstances, is achieving DR (or proximity analysis). This allows to see SoCS as a non linear DR algorithm that "flattens" the social manifold. Moreover, the importance of locality in non-linear DR fits with the distributed implementation of SOCS.

6 Conclusion

In this paper, we presented SoCS, a fully distributed algorithm for social graph embedding. SoCS distributes traditional force-based embedding algorithms to embed graphs while preserving their community structure [21]. SoCS exploits the benefits of the community structure knowledge for link prediction [4] to achieve meaningful recommendations. SoCS relies on gossip protocols to approximate the forces applied to the nodes and achieve scalability in large and dynamic graphs.

We extensively analyzed the performance of SoCS on both synthetic and real data. Our experiments show that SoCS is able to accurately assign social coordinates to nodes. The configurations that obtain the best results are the ones that favor a precise representation of the local social neighborhood to global accuracy. Not only does this setting improve the quality of the results, it also offers better scalability and more resilience to large-scale network perturbations such as churn.

Future work includes evaluating the impact of other force models in order to find the best trade-off between accuracy and applicability in P2P systems, as well as achieving a better understanding of the algorithm parameter space. The privacy-protecting properties of SoCS have also to be studied.

References

[1] Y. Aumann and Y. Rabani. An o(log k) approximate min-cut max-flow theorem and approximation algorithm, 1997.

[2] J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52:46–52, 1985.

[3] L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009.

[4] A. Clauset, C. Moore, and M. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[5] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *SIGCOMM*, pages 15–26, 2004.

[6] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

[7] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[8] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222, 2003.

[9] P. Fraigniaud, E. Lebhar, and Z. Lotker. Recovering the Long-Range Links in Augmented Graphs. In *SIROCCO*, page 118, 2008.

[10] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement, 1991.

[11] O. Goussevskaia, M. Kuhn, M. Lorenzi, and R. Wattenhofer. From web to map: Exploring the world of music. In *WI-IAT*, pages 242–248, 2008.

[12] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 1970.

[13] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *TOCS.*, 25(3):8, 2007.

[14] D. Kempe and F. McSherry. A decentralized algorithm for spectral analysis. In *STOC*, pages 561–568, 2004.

[15] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *STOC*, pages 163–170, 2000.

[16] J. Kleinberg. Complex networks and decentralized search algorithms. In *ICM*, 2006.

[17] Y. Koren. Drawing graphs by eigenvectors: theory and practice. *Computers & Mathematics with Applications*, 49(11-12):1867–1888, 2005.

[18] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *ASIS&T*, 58(7), 2007.

[19] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:577–591, 1994.

[20] A. Noack. An energy model for visual graph clustering. In *Graph Drawing*, pages 425–436, 2003.

[21] A. Noack. Modularity clustering is force-directed layout. *Physical Review E*, 2009.

[22] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[23] F. J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *ICML*, 1998.

[24] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323, 2000.

[25] F. van Ham and J. J. van Wijk. Interactive visualization of small world graphs. In *INFOVIS*, pages 199–206, 2004.

[26] S. Voulgaris and M. V. Steen. Epidemic-style management of semantic overlays for content-based searching. In *EuroPar*, pages 1143–1152, 2005.

[27] P. M. Winkler. Proof of the squashed cube conjecture. *Combinatorica*, 3(1):135–139, 1983.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399