

Distributed Shortest Paths on Power Law Networks in the Generalized Linear Preference Model: An Experimental Study

Mattia D'Emidio and Daniele Frigioni(✉)

Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Via Vetoio, L'Aquila 67100, Italy
{mattia.demidio,daniele.frigioni}@univaq.it

Abstract. The problem of computing, in a distributed fashion, the shortest paths of a dynamic graph is a core functionality of modern communication networks. Distance vector algorithms are widely adopted solutions for this problem when scalability and reliability are key issues or when nodes have limited hardware resources, as they result very competitive in terms of memory and computational requirements. In this paper, we first discuss some recent distance vector solutions, and then present the results of an ongoing experimental study, conducted on a prominent category of networks, namely *generalized linear preference* power-law networks, to rank the performance of such solutions.

1 Introduction

The efficient distributed computation of the shortest paths of a graph whose topology dynamically changes over time is a core functionality of many modern and emergent digital infrastructures, as, for example, communication networks. For this reason, the problem has been widely investigated in past decades and many solutions have been proposed, which can be classified into two categories, namely *distance vector* and *link state* algorithms [12,13,15].

On one hand, distance vector algorithms, as for example the *Distributed Bellman-Ford* (DBF) method [17] are characterized by a *local* nature, in the sense that a generic node interacts only with its neighbors and stores minimal information about the global status of the network. In more details, each node typically maintains only a *routing table* containing, for each other node of the network, the *distance*, i.e. the weight of a shortest path, and the *next hop*, i.e. the next node on the same shortest path. The computation of the routing table is performed by solving very simple equations (see [4] and references therein), thus making distance vector algorithms very competitive solutions from the computational complexity point of view. However, in dynamic scenarios they can suffer from the well-known *looping* and *count-to-infinity* phenomena, that can heavily

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11620, pp. 72–81, 2019.
https://doi.org/10.1007/978-3-030-24296-1_8

affect their performance in terms of communication resources (a.k.a. *message complexity*), tough quite efficient countermeasures are known (see, e.g. [12]).

On the other hand, link state algorithms, as for example the *Open Shortest Path First* (OSPF) protocol [15], are characterized by a *global* nature, as they require each node to store the entire network topology. Shortest paths are usually computed by running a centralized algorithm, as for example the classic Dijkstra's algorithm. This results in a usage of memory which is quadratic in the number of nodes of the network. On the positive side, link state approaches do not incur in looping and count-to-infinity phenomena, thus being more competitive than distance vector algorithms in terms of message complexity. However, this is counterbalanced in dynamic scenarios, where they perform quite poorly in this sense, since each node needs to receive and store up-to-date information on the entire network topology after any change [15,17].

In the last years, there has been a renewed interest in devising efficient and light-weight distance vector shortest path algorithms for large-scale networks (see, e.g., [4–11,18]), when scalability and reliability are key issues or when the computational resources of the nodes of the network are limited. The most important distance vector algorithm in the literature is considered DUAL (*Diffuse Update ALgorithm*) [12], which is part of CISCO's *Enhanced Interior Gateway Routing Protocol* (EIGRP). DUAL is more complex than DBF since it uses, besides the routing table, several auxiliary data structures to guarantee freedom from looping and count-to-infinity. Another loop-free distance vector algorithm that is worth to be mentioned is *Loop Free Routing* (LFR), which has been more recently proposed in [6]. LFR has the same message complexity of DUAL but it uses an amount of data structures per node which is always smaller than that of DUAL. From the experimental point of view, LFR has been shown in [6] to be very effective in practice. In [10], a general technique, named *Distributed Computation Pruning* (DCP) has been proposed in order to overcome some of the main limitations of distance vector algorithms, e.g. high number of messages sent and poor convergence. DCP can be applied to any distance vector algorithm to boost its performance, and it has been designed to be effective on networks following a power-law node degree distribution, also referred as *power-law networks*. Such class of networks is of particular practical relevance, since it includes some of the most important modern network applications, as the Internet and social networks. In [10] the effectiveness of DCP has been shown via an experimental study conducted on: (i) IPv4 topology datasets of the *Cooperative Association for Internet Data Analysis* (CAIDA) [14]; (ii) synthetic power-law topologies generated by the *Barabási-Albert* algorithm [1].

In this paper, we show the results of an ongoing experimental study whose purpose is to assess the performance of DUAL, LFR and their combinations with DCP on a practically relevant class of power-law networks, namely the artificial instances obtained by the Generalized Linear Preference (GLP) model [3]. The GLP framework has been shown to model very well the Internet, and parts of it. In particular, GLP predicts the structure of real-world communication networks better than the *Barabási-Albert* model. This is due to the fact that GLP adds more flexibility in specifying how nodes connect to other nodes. Our

experimental study gives strong evidences that combining DUAL and LFR with DCP, and running them on GLP topologies, allows: (i) a huge reduction in the utilization of communication resources (measured in terms of number of messages sent) with respect to the case without DCP; (ii) a significant improvement in terms of memory requirements with respect to the case without DCP.

2 Background

We consider the classic distributed scenario where we have a network made of processors that are connected through communication channels and exchange data using a message passing model, in which: each processor can send messages only to processors it is connected with; messages are delivered to their destination within a finite delay; there is no shared memory among the processors; the system is *asynchronous*, that is a sender of a message does not wait for the receiver to be ready to receive the message. The message is delivered within a finite time.

Graph Notation. We represent a network by an undirected weighted connected graph $G = (V, E, w)$, where V is a finite set of n nodes, one for each processor, E is a finite set of m edges, one for each communication channel, and w is a weight function $w : E \rightarrow \mathbb{R}^+$. Given a graph $G = (V, E, w)$, we denote by: (v, u) an edge of E that connects nodes $v, u \in V$, and by $w(v, u)$ its weight, respectively; $N(v) = \{u \in V : (v, u) \in E\}$ the set of neighbors of a node $v \in V$; $deg(v) = |N(v)|$ the degree of v, for each $v \in V$; $\{u, \ldots, v\}$ a generic path in G between nodes u and v and, given a path P, $w(P)$ is its *weight*, i.e. the sum of the weights associated to its edges. A path $P = \{u, ..., v\}$ is called a *shortest path* if and only if P is a path having minimum weight among all possible paths between u and v in G. Given two nodes $u, v \in V$, we denote by $d(u, v)$ the topological *distance* between u and v, i.e. the weight of a *shortest path* between u and v. Finally, we call $via(u, v)$ the *via* from u to v, i.e. the set of neighbors of u that belong to a shortest path from u to v. More formally, $via(u, v) \equiv \{z \in N(u) \mid d(u, v) = w(u, z) + d(z, v)\}$. We concentrate on the realistic case of *dynamic networks* that vary over time due to change operations occurring on the edges.

Distance Vector Algorithms. Most of distance vector algorithms share a set of common features which can be summarized as follows. Let be given a weighted graph $G = (V, E, w)$, and a generic node v of G executing a distance vector algorithm. Node v knows the identity of any other node of G, as well as the identity of its neighbors and the weights of its adjacent edges. It maintains a routing table that has n entries, one for each $s \in V$, which consists of at least two fields: (1) the *estimated distance* $\mathrm{D}_v[v, s]$ towards s, i.e. an estimation of $d(v, s)$; (2) the *estimated via* $\textsc{Via}_v[s]$ towards s, i.e. an estimation of $via(v, s)$. Node v requests data to neighbors, regarding estimated distances, and receives the corresponding replies from them, through a dedicated exchange of messages. v propagates a variation, occurring on the distance or on the via, to the rest of the network by sending out to its neighbors a dedicated notification message; a

node that receives this kind of message executes a corresponding routine. It is well known that a distance vector algorithm can be designed to be free of looping or count-to-infinity by incorporating suitable sufficient conditions in the routing table update procedures [12].

Power-Law Networks. A *power-law network*, in the most general meaning, is a network where the distribution of the nodes' degree follows a power-law trend, thus having many nodes with low degree and few (core) nodes with very high degree (see e.g. [1]). Such class of networks is very important in practice, since it includes many of the currently implemented communication infrastructures, like the Internet, some social networks, and so on. For this reason, many methods to generate artificial topologies exhibiting a power-law behaviour have been proposed in the literature. Among them, the most prominent are the Barabási-Albert (BA) model [1] and the Generalized Linear Preference (GLP) model [3].

3 Distributed Computation Pruning

The DCP technique of [10] is not an algorithm by itself, instead it is a general technique that can be applied on top of a distance vector algorithm with the aim of improving its performance. Given a generic distance vector algorithm A, the combination of DCP with A induces a new algorithm, denoted by A-DCP.

In the following, we report some definitions given in [10] that are useful to capture the typical structure of a power-law network. Given a graph $G = (V, E, w)$, nodes, edges and paths in G can be classified as follows. A node $v \in V$ is *central* if $deg(v) \geq 3$ and *non-central* otherwise. A non-central node v can be either *semi-peripheral*, if $deg(v) = 2$, or *peripheral*, if $deg(v) = 1$. Given a path $P = \{v_0, v_1, \ldots, v_j\}$ of G, P is *central* if v_i is central, for each $0 \leq i \leq j$. Any edge belonging to a central path is called *central edge*. P is *peripheral* if v_0 is central, v_j is peripheral, and all v_i, for each $1 \leq i \leq j-1$, are semi-peripheral. Any edge belonging to a peripheral path, accordingly, is called *peripheral edge*. P is *semi-peripheral* if v_0 and v_j are two distinct central nodes, and all v_i are semi-peripheral nodes, for each $1 \leq i \leq j-1$. If $v_0 \equiv v_j$ then P is called a *semi-peripheral cycle* Each edge belonging to such a path is called *cyclic edge* and each node $u \neq v_0$ in P is called *cyclic node*.

Description. DCP has been designed to reduce the communication overhead of distance vector algorithms in power-law networks. This can be done since nodes with small degree often do not provide any useful information for the distributed computation of shortest paths. in fact, there are many topological situations in which these nodes neither perform, nor be involved in any kind of distributed computation, since their shortest paths depend on those of higher degree nodes.

In particular, DCP forces the distributed computation to be carried out by the central nodes only (which are few in power-law networks). Non-central nodes, which are instead the great majority, play a passive role and receive updates about routing information from the central nodes, without taking part to any kind of distributed computation and by performing few trivial operation

to update their routing data. Hence, it is clear that the larger is the set of non-central nodes of the network, the bigger is the improvement in the pruning of the distributed computation of shortest paths and, consequently, the improvement in the global number of messages sent by A-DCP with respect to A.

The main difference in the behaviour of A, when combined with DCP, resides in the fact that in a A every node performs the same code thus having the same behaviour, while in A-DCP, central and non-central nodes are forced to have different behaviours. In particular, central nodes detect (and handle) changes concerning all kinds of edges, while peripheral, semi-peripheral, and cyclic nodes detect changes concerning only peripheral, semi-peripheral, and cyclic edges.

Practical Effectiveness of DCP. In [10] DCP has been combined with both DUAL and LFR by obtaining algorithms denoted as DUAL-DCP and LFR-DCP, and an extensive experimental study has been performed to assess the effectiveness of DCP. In these experiments the power-law networks of the *CAIDA IPv4 topology dataset* [14], and the random power-law networks generated by the *Barabási-Albert* (BA) algorithm [1] were used, subject to randomly generated sequences of updates. The results of [10] have shown that the combinations of DUAL and LFR with DCP provide a huge improvement in the number of messages sent. In particular, in CAIDA instances the ratio between the number of messages sent by DUAL-DCP and those sent by DUAL is within 0.03 and 0.16 which means that DUAL-DCP sends a number of messages which is between 3% and 16% that of DUAL. The ratio between the number of messages sent by LFR-DCP and those sent by LFR is within 0.10 and 0.26. Similar results are obtained on *Barabási-Albert* instances.

4 Generalized Linear Preference Model

The Generalized Linear Preference (GLP) model has been introduced in [2], and it is a variant of the Barabási–Albert (BA) model [1]. A BA topology is generated by iteratively adding one node at a time, starting from a connected graph with at least two nodes. A newly added node is connected to any other existing nodes with a probability that is proportional to the degree of the existing nodes. In detail, let $P(i)$ denote the probability that a new node will be connected to node i, the *linear preference* connectivity rule of the BA model is defined as follows:

$$P(i) = \frac{deg(i)}{\sum_j deg(j)} \tag{1}$$

As a result of the above connectivity mechanism, the power-law graphs generated by the BA algorithm have average node degree approximately equal to 3 and a number of nodes with degree smaller than 3 approximately equal to $7/10n$, where n is the number of nodes of the graph. In [3] it has been shown that in the real Internet, new Autonomous Systems (AS) have a much stronger preference to connect to high degree ASs than predicted by the linear preference model. This behaviour is better captured by the GLP model which adds more flexibility

than the BA one in specifying how nodes connect to other nodes. In detail, the GLP model reflects the fact that the evolution of the AS graph is mostly due to two operations, the addition of new nodes and the addition of new links between existing nodes. It starts with n_0 nodes connected through $n_0 - 1$ links. At each time-step, one of the following two operations is performed: (1) with probability $p \in [0, 1]$, $k < n_0$ new links are added between existing nodes; (2) with probability $1 - p$, a new node is added and connected to k existing nodes. If $P(i)$ denotes the probability that a new node is connected to node i, the *generalized linear preference* connectivity rule is defined as follows:

$$P(i) = \frac{deg(i) - \beta}{\sum_j (deg(j) - \beta)} \tag{2}$$

where $\beta \in (-\infty, 1)$ is a tunable parameter that can be adjusted such that nodes have a stronger preference of being connected to high degree nodes than predicted by Eq. (1). In particular, it indicates the preference for a new node (edge) connecting to high degree nodes. The smaller the value of β is, the less preference is given to high degree nodes. As a result of Eq. (2), Power-law graphs generated by the GLP model have average node degree of 4.8 and a number of nodes with degree smaller than 3 that is on average around $8/10n$.

5 Evaluation of DCP on GLP Networks

In this section we present the results of our new experimental study of algorithms DUAL, LFR, DUAL-DCP, and LFR-DCP on GLP networks. Our experiments have been performed on a workstation equipped with a Quad-core 3.60 GHz Intel Xeon X5687 processor, with 12 MB of internal cache and 24 GB of main memory, and consist of simulations within the OMNeT++ 4.0p1 environment [16]. All software has been written in C++ and compiled with GNU g++ compiler v.4.8 under Linux (Kernel 2.6.32).

Executed Tests. We implemented the GLP algorithm and first generated a large connected graph, having 16 031 nodes, by fixing β to 0.6447, since this choice has been shown to allow the generation of networks that are similar to real-world ones [3]. Since DUAL has high memory requirements, we were unable to perform tests on such large graph. Therefore, we randomly extracted different subgraphs of smaller sizes, by performing pruned Breadth First Searches rooted at different randomly chosen nodes. Finally, we generated a set of different experimental settings similarly to what was done in [10], where each setting consists of a n-node subgraph of the above graph, denoted as G_{GLP-n}, and a set of k edge updates, with k assuming values in $\{5, 10, \ldots, 30\}$. Edge weights are non-negative real numbers randomly chosen in $[1, 1\,000\,000]$ and an edge update consists of multiplying the weight of a random selected edge by a percentage value randomly chosen in $[50\%, 150\%]$. For each test configuration (a graph and a fixed value of k) we performed 5 different experiments (for a total amount of 30 runs), measured performance metrics of interest, and computed average values for each

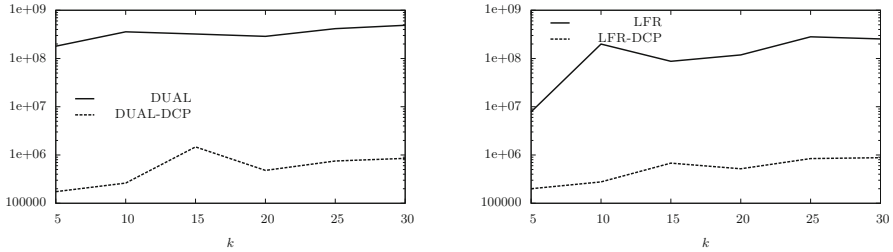

Fig. 1. (a) Number of messages sent by DUAL and DUAL-DCP on $G_{GLP-8000}$. (b) Number of messages sent by LFR and LFR-DCP on $G_{GLP-8000}$.

metric. We ran simulations on GLP instances with $n \in \{1200, 5000, 8000\}$. The results of our experiments on the different instances are similar, hence we report only those on $G_{GLP-8000}$, having 8000 nodes, 19158 edges, average node degree equal to 4.8, around 58% of peripheral nodes, and around 22% of semi-peripheral nodes.

Analysis. A summary of our results is shown in Fig. 1, where we report in (a) the average number of messages sent by DUAL and DUAL-DCP, and in (b) that of LFR and LFR-DCP, respectively, on $G_{GLP-8000}$, per test.

Our data suggest that the use of DCP provides a huge improvement in the global number of messages sent. In more details, in the tests of Fig. 1(a) the ratio between the number of messages sent by DUAL-DCP and those sent by DUAL is within $73 \cdot 10^{-5}$ and $457 \cdot 10^{-5}$ while in the tests of Fig. 1(b) the ratio between the number of messages sent by LFR-DCP and those sent by LFR is within $138 \cdot 10^{-5}$ and $2555 \cdot 10^{-5}$. The improvement provided by DCP in GLP graphs is by far more significant than that provided in case of BA graphs in [10]. In fact, the ratio between the number of messages sent by DUAL-DCP (LFR-DCP, respectively) and those sent by DUAL (LFR, respectively) is always larger than 10^{-2}. Since the GLP model is known to better predict real-world network infrastructures with respect to the BA one, this provides even more evidences of the effectiveness of DCP in real-world network instances.

Notice that the very good performance of DCP in this case is probably due to the topological structure of GLP graphs, which have more central nodes than BA ones. This results in a much more effective pruning of both DUAL and LFR with respect to what happens in BA instances. Notice also that this behaviour is more evident for LFR-DCP as LFR includes two sub-routines, the Local-Computation and the Global-Computation, which worst case message complexity depends on the number of central nodes (see [6]), while DUAL uses a single sub-routine, the Diffuse-Computation, which worst case message complexity depends on the same parameter (see [12]).

By the two diagrams of Fig. 1, it is clear that LFR outperforms DUAL on $G_{GLP-8000}$. This is due to the fact that GLP graphs have been shown to model very accurately real world topologies where LFR is very effective in terms of

Table 1. Space occupancy per node of the algorithms on $G_{GLP-8000}$.

Algorithm	MAX		AVG	
	Bytes	Ratio	Bytes	Ratio
DUAL	38 888 000	1	359 580	1
DUAL-DCP	8 163 826	0.21	327 366	0.91
LFR	241 115	1	192 069	1
LFR-DCP	354 212	1.47	348 636	1.82

number of messages sent. This consideration also explains why the reduction in the number of messages sent induced by DCP is less emphasized for LFR.

To conclude our analysis, we consider the space occupancy. Note that, when DCP is combined with DUAL, a subset of nodes of the network can avoid to maintain some data structures of DUAL, as either the information stored in them can be inferred, or it is not needed due to the pruning mechanism of DCP. For instance, each node of the network executing DUAL-DCP does not need to store the data structure of DUAL that implements the finite state machine with respect to non-central nodes, as no distributed computation can be initiated for this kind of nodes. Similar observations can be done with respect to LFR, since also here some of the nodes of the network can avoid to maintain some of the data structures used by the algorithm, which can be either inferred or it is not needed at all due to the pruning mechanism of DCP.

To support the above considerations we provide measurements on the space occupancy per node in Table 1, where we report the maximum and the average space occupancy per node (in Bytes) of each algorithm on $G_{GLP-8000}$. We also report the ratio between the space occupancy per node of the algorithms integrating DCP and that of the original algorithms, for each test instance. Since the space occupancy per node of LFR and LFR-DCP depends on the number of weight change operations, we report median values for each of the algorithms.

Our experiments show that the use of DCP induces a clear improvement also in the space requirements per node. In particular, DUAL-DCP requires a maximum space occupancy per node which is 0.29 times that of DUAL. This behaviour is confirmed also in the average case, where DUAL-DCP requires 0.91 times the average space occupancy per node of DUAL. On the contrary, our data show that the average space occupancy per node of LFR-DCP is slightly larger than that of LFR and that the use of DCP induces an overhead in the average space occupancy per node which is equal to 82%. This is due to the fact that the space occupancy of LFR is quite low by itself and that, in this case, the space occupancy overhead due to the data structures of DCP (see [10]) is larger than the space occupancy reduction induced by the use of DCP.

6 Conclusions

In conclusion, our experiments show that in the case of GLP networks, the DCP technique is very effective when combined with some prominent distance vector algorithms in terms of improving the usage of communication resources.

The study of new techniques, which take advantage of the structural properties of real-world networks, deserves further investigation. In particular, it would be interesting to evaluate how DCP scales to bigger a/o more dynamic networks, and also to develop techniques for shortest paths distributed algorithms to be efficient in other practically interesting scenarios.

References

1. Albert, R., Barabási, A.-L.: Emergence of scaling in random networks. Science **286**, 509–512 (1999)
2. Albert, R., Jeong, H., Barabási, A.-L.: Error and attack tolerance of complex networks. Nature **406**, 378–381 (2000)
3. Bu, T., Towsley, D.: On distinguishing between internet power law topology generators. In: Proceedings IEEE INFOCOM, pp. 638–647 (2002)
4. Cicerone, S., D'Angelo, G., Di Stefano, G., Frigioni, D., Maurizio, V.: Engineering a new algorithm for distributed shortest paths on dynamic networks. Algorithmica **66**(1), 51–86 (2013)
5. D'Angelo, G., D'Emidio, M., Frigioni, D.: Fully dynamic update of arc-flags. Networks **63**(3), 243–259 (2014)
6. D'Angelo, G., D'Emidio, M., Frigioni, D.: A loop-free shortest-path routing algorithm for dynamic networks. Theor. Comput. Sci. **516**, 1–19 (2014)
7. D'Angelo, G., D'Emidio, M., Frigioni, D.: Distance queries in large-scale fully dynamic complex networks. In: Mäkinen, V., Puglisi, S.J., Salmela, L. (eds.) IWOCA 2016. LNCS, vol. 9843, pp. 109–121. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44543-4_9
8. D'Angelo, G., D'Emidio, M., Frigioni, D.: Fully dynamic 2-hop cover labeling. J. Exp. Algorithmics **24**(1), 1.6:1–1.6:36 (2019)
9. D'Angelo, G., D'Emidio, M., Frigioni, D., Maurizio, V.: A speed-up technique for distributed shortest paths computation. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2011. LNCS, vol. 6783, pp. 578–593. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21887-3_44
10. D'Angelo, G., D'Emidio, M., Frigioni, D., Romano, D.: Enhancing the computation of distributed shortest paths on power-law networks in dynamic scenarios. Theory Comput.Syst. **57**(2), 444–477 (2015)
11. Elkin, M.: Distributed exact shortest paths in sublinear time. In: Proceedings of 49th Annual Symposium on Theory of Computing (STOC 2017), pp. 757–770. ACM (2017)
12. Garcia-Lunes-Aceves, J.J.: Loop-free routing using diffusing computations. IEEE/ACM Trans. Netw. **1**(1), 130–141 (1993)
13. Humblet, P.A.: Another adaptive distributed shortest path algorithm. IEEE Trans. Commun. **39**(6), 995–1002 (1991)
14. Hyun, Y., et al.: The CAIDA IPv4 routed/24 topology dataset. http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml

15. Moy, J.T.: OSPF: Anatomy of an Internet Routing Protocol. Addison-Wesley, Boston (1998)
16. OMNeT++. Discrete event simulation environment. http://www.omnetpp.org
17. Rosen, E.C.: The updating protocol of ARPANET'S new routing algorithm. Comput. Netw. **4**, 11–19 (1980)
18. Ray, S., Guérin, R., Kwong, K., Sofia, R.: Always acyclic distributed path computation. IEEE/ACM Trans. Netw. **18**(1), 307–319 (2010)