Improving Resource Location with Locally Precomputed Partial Random Walks*

Víctor M. López Millán[1], Vicent Cholvi[2],
Luis López[3], and Antonio Fernández Anta[4]

[1] Universidad CEU San Pablo, Spain
vmlopez.eps@ceu.es
[2] Universitat Jaume I, Spain
vcholvi@uji.es
[3] Universidad Rey Juan Carlos, Spain
llopez@gsyc.es
[4] Institute IMDEA Networks, Spain
antonio.fernandez@imdea.org

Abstract. Random walks can be used to search complex networks for a desired resource. To reduce search lengths, we propose a mechanism based on building random walks connecting together partial walks (PW) previously computed at each network node. Resources found in each PW are registered. Searches can then jump over PWs where the resource is not located. However, we assume that perfect recording of resources may be costly, and hence, probabilistic structures like Bloom filters are used. Then, unnecessary hops may come from false positives at the Bloom filters. Two variations of this mechanism have been considered, depending on whether we first choose a PW in the current node and then check it for the resource, or we first check all PWs and then choose one. In addition, PWs can be either simple random walks or self-avoiding random walks. Analytical models are provided to predict expected search lengths and other magnitudes of the resulting four mechanisms. Simulation experiments validate these predictions and allow us to compare these techniques with simple random walk searches, finding very large reductions of expected search lengths.

Keywords: Random walks, self-avoiding random walks, network search, resource location, search length.

1 Introduction

A *random walk* in a network is a routing mechanism that chooses the next node to visit at random among the neighbors of the current node. Random walks have been extensively studied in mathematics, and have been used in a wide range of

* This research was supported in part by Comunidad de Madrid grant S2009TIC-1692, Spanish MICINN grant TEC2011-29688-C02-01, Spanish MEC grant TIN2011-28347-C02-01 and Bancaixa grant P11B2010-28.

V. Gramoli and R. Guerraoui (Eds.): NETYS 2013, LNCS 7853, pp. 144–158, 2013.

applications such as statistic physics, population dynamics, bioinformatics, etc. When applied to communication networks, random walks have had a profound impact on algorithms and complexity theory. Some of the advantages of random walks are their simplicity, their small processing power consumption at the nodes, and the fact that they need only local information, avoiding the communication overhead necessary in other routing mechanisms. An important application of random walks has been the search for resources held in the nodes of a network, also known as the *resource location problem*. Roughly speaking, the problem consists of finding a node that holds the resource, starting at some *source node*. Random walks can be used to perform such a search as follows. It is checked first if the source node holds the resource. If it does not, the search hops to a random neighbor, that repeats the process. The search proceeds through the network in this way until a node that holds the resource is found. Due to the random nature of the walk, some nodes may be visited more than once (unnecessarily from the search standpoint), while other nodes may remain unvisited for a long time. The number of hops taken to find the resource is called the *search length* of that walk. The performance of this direct application of random walks to network search has been studied in [1,2,3,4,5].

The use of random walks for resource location has several clear applications, like unstructured peer-to-peer (P2P) file sharing systems or content-centric networks (CCN) [6]. The latter are networks in which the key elements are named content chunks, which are requested by users using the content name. Content chunks have to be efficiently located and transferred to be consumed by the user. The techniques described in this paper could be used in the context of CCN to locate content chunks.

Contributions. This paper proposes an application to resource location of the technique of concatenating partial walks (PW) available at each node to build random walks. A PW is a precomputed random walk of fixed length. Two variations are considered, depending on whether the search mechanism first randomly chooses one of the PWs in the current node and then checks its associated information for the desired resource, or it first checks all PWs in the node and then randomly chooses among those with a positive result. Both of these variations may use PWs that are simple random walks (RW) or self-avoiding random-walks (SAW), resulting in four mechanisms referred to as *choose-first* PW-RW or PW-SAW, and *check-first* PW-RW or PW-SAW, respectively. Our mechanisms assume the use of Bloom filters [7] to efficiently store the set of resources (not their owners) held by the nodes in each partial walk. The compactness of Bloom filters comes at the price of possible *false positives* when checking if a given resource is in the partial walk. False positives occur with a probability p, which is taken into account in our analyses. These assumptions provide generality to our model, since a probability of $p = 0$ models the case in which the full list of resources found are stored (instead of using a Bloom filter).

We provide an analytical model for the choose-first PW-RW technique, with expressions for the *expected search length*, the *optimal length of the partial walks*, and for the *optimal expected search length*. We found that, when the probability

of false positives in Bloom filters is small, the optimal expected search length is proportional to the square root of the expected search length achieved by simple random walks, in agreement with the results in [8]. Another interesting finding is that the optimal length of the partial walks does not depend on the probability of false positives of the Bloom filters. We also provide analytical models for the choose-first PW-SAW mechanism as well as for the check-first variations, which predict their expected search length. Then, the predictions of the models are validated by simulation experiments in three types of randomly built networks: regular, Erdős-Rényi, and scale-free. These experiments are also used to compare the performance of the four mechanisms, and to investigate the influence of parameters as the false positive probability and the number of partial walks per node. Finally, we have compared the performance of the four search mechanisms with respect to simple random walk searches. For choose-first PW-RW we have found a reduction in the average search length ranging from around 98% to 88%. For choose-first PW-SAW such a reduction is even bigger, ranging from 12% to 5% with respect to PW-RW. Check-first PW-RW and PW-SAW can achieve still larger reductions increasing the number of PWs available at each node.

Related Work. Das Sarma et al. [8] proposed a distributed algorithm to obtain a random walk of a specified length ℓ in a number of rounds[1] proportional to $\sqrt{\ell}$. In the first phase, every node in the network prepares a number of *short (random) walks* departing from itself. The second phase takes place when a random walk of a given length starting from a given source node is requested. One of the short walks of the source node is randomly chosen to be the first part of the requested random walk. Then, the last node of that short walk is processed. One of its short walks is randomly chosen, and it is *connected* to the previous short walk. The process continues until the desired length is reached.

Hieungmany and Shioda [9] proposed a *random-walk-based* file search for P2P networks. A search is conducted along the concatenation of hop-limited shortest path trees. To find a file, a node first checks its *file list* (i.e., an index of files owned by neighbor nodes). If the requested file is found in the list, the node sends the file request message to the file owner. Otherwise, it randomly selects a leaf node of the hop-limited shortest path tree, and the search follows that path, checking the *file list* of each node in it.

The use of partial random walks in resource location has been proposed in [10] for networks with dynamic resoures. Our work in this paper incorporates efficient storage by means of Bloom filters, in the context of static resources. The use of SAWs as PWs is also proposed and compared with simple RWs.

Structure. The next section presents a model for the four search mechanisms proposed. Then, the choose-first PW-RW is evaluated in Section 3. For the

[1] A *round* is a unit of discrete time in which every node is allowed to send a message to one of its neighbors. According to this definition, a simple random walk of length ℓ would then take ℓ rounds to be computed.

sake of clarity, the choose-first PW-SAW mechanism is covered separately in Section 4, which includes the corresponding analysis together with performance results. Similarly, the check-first PW-RW/PW-SAW mechanisms are presented in Section 5.

2 Model

Let us consider a randomly built network of N nodes and arbitrary topology, whose nodes hold resources randomly placed in them. Resources are unique, i.e., there is a single instance of each resource in the network. The resource location problem is defined as visiting the node that holds the resource, starting from a certain node (the *source* node). For each search, the source node is chosen uniformly at random among all nodes in the network.

The search mechanisms proposed in this paper exploit the idea of efficiently building *total random walks* from *partial random walks* available at each node of the network. This process comprises two stages:

(1) Partial Walks Construction. Every node i in the network precomputes a set W_i of w random walks in an initial stage before the searches take place. Each of these partial walks has length s, starting at i and finishing at a node reached after s hops. In the PW-RW mechanism, the partial walks computed in this stage are simple random walks. During the computation of each partial walk in W_i, node i registers the resources held by the s first nodes in the partial walk (from i to the one before the last node). As mentioned, for generality, we assume that the resources found are stored in a Bloom filter. This information will be used in Stage 2. Bloom filters are space-efficient randomized data structures to store sets, supporting membership queries. Thus, the Bloom filter of a partial walk can be queried for a given resource. If the result is negative, the resource is not in any of the nodes of the partial walk. If the result is positive, the resource is in one of the nodes of the partial walk, unless the result was a *false positive*, which occurs with a certain probability p.[2] The size of the Bloom filters can be designed for a target (small) p considered appropriate. A variation of the partial walk construction mechanism consists of using PWs that are *self-avoiding* walks (SAW). The resulting mechanism, called PW-SAW, is analyzed in Section 4.

(2) The Searches. After the PWs are constructed, searches are performed in the following fashion when the choose-first PW-RW/PW-SAW mechanisms are used. When a search starts at a node A, a PW in W_A is chosen uniformly at random. Its Bloom filter is then queried for the desired resource. If the result is negative, the search *jumps* to node B, the last node of that partial walk. The process is then repeated at B, so that the search keeps jumping in this way while the results of the queries are negative. When at a node C, the query to the Bloom filter (of the PW randomly chosen from W_C) gives a positive result,

[2] More concretely, p is the probability of obtaining a positive result conditioned on the desired resource not being in the filter.

the search *traverses* that partial walk looking for the resource until the resource is found or the partial walk is finished. If the resource is found, the search stops. If the search reaches the last node D of the partial walk without having found the resource in the previous nodes, it means that the result of the Bloom filter query was a false positive. The search then randomly chooses a partial walk in W_D and decides whether to jump over it or to traverse it depending on the result of the query to its Bloom filter, as described above. A variation of this behavior consists of first checking all PWs of the node for the desired resource, and then randomly choosing among the ones with a positive result. The resulting mechanisms, called check-first PW-RW/PW-SAW are analyzed in Section 5.

In this work, we are interested in the number of *hops* to find a resource (when PWs of length s are used), which is defined as the *search length* and denoted L_s. Some of these hops are *jumps* (over PWs) and other are *steps* (traversing PWs). In turn, we distinguish between *trailing steps*, if they are the ones taken when the resource is found, and *unnecessary steps*, if they are taken when the resource is not found. The search length is a random variable that takes different values when independent searches are performed. The *search length distribution* is defined as the probability distribution of the search length random variable. We are interested in finding the *expected search length*, denoted $\overline{L}_s$. Figure 1 summarizes the behavior of the search mechanisms.

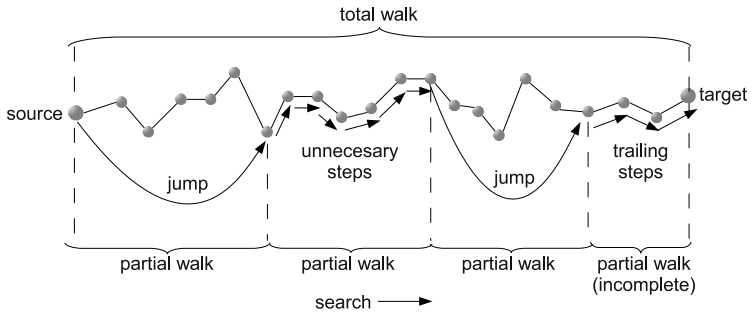


Fig. 1. An example of search, using PWs of length $s = 6$

At this point, we emphasize the difference between the *search* just defined and the *total walk* that supports it, consisting of the concatenation of *partial walks* as defined above. Searches are shorter in length than their corresponding total walks because of the number of steps saved in jumps over partial walks in which we know that the resource is not located (although these saving may be reduced by the unnecessary steps due to Bloom filter false positives).

3 Choose-First PW-RW

3.1 Analysis of Choose-First PW-RW

We make an additional assumption in order to simplify this analysis. Once a PW has been used in the total walk of a search, it is never reused again in that

total walk or in any other searches. Thus we guarantee that the total walks are true random walks. This implies that in practice each node needs to have a large number of precomputed partial walks (w), assumption that would compromise the benefits of the proposed mechanism in practice. Simulations in Section 3.3 show that real cases with small w behave very similarly to the base case provided by this analysis.

Let L_s be the random variable representing the number of hops in the search (i.e., its length) when PWs of length s are used. The expected search length is denoted by $\overline{L}_s$. Let L be the random variable representing the number of hops of the corresponding total walk. Its expected search length is denoted $\overline{L}$. Making use of the assumption that partial walks are never reused, L can be viewed as the length of a search based on a simple random walk in the considered network, and $\overline{L}$ as the expected search length of random walks in that network. Then, we can state the following theorem:

Theorem 1. *If the expected number of trailing steps is assumed to be uniformly distributed in* $[0, s-1]$[3]*, then the expected search length is:*

$$\overline{L}_s = \left(\frac{s}{2} + \frac{2\overline{L}+1}{2s} - 1\right) \cdot (1-p) + \overline{L} \cdot p. \tag{1}$$

Proof. Let P, J, U and T be random variables representing the number of partial walks, jumps, unnecessary steps and trailing steps in a search, respectively. Their expectations are denoted as $\overline{P}$, $\overline{J}$, $\overline{U}$ and $\overline{T}$. Since hops in a search can be jumps, unnecessary steps or trailing steps, it follows that, $L_s = J + U + T$. Then, the expected search length for partial walks of size s is[4] $\overline{L}_s = \overline{J} + \overline{U} + \overline{T}$.

The expected number of jumps can be obtained from the expected number of partial walks in the search ($\overline{P}$) and from the probability of false positive (p) as $\overline{J} = \overline{P} \cdot (1-p)$, since J follows a binomial distribution $B(P, 1-p)$, where the number of experiments is the random variable representing the number of partial walks in a search (P) and the success probability is the probability of obtaining a negative result in a Bloom filter query $(1-p)$.[5]

For the expected number of unnecessary steps, $\overline{U} = \overline{P} \cdot p \cdot s$, since $\overline{P} \cdot p$ is the expected number of false positives in the search and each of them contributes with s unnecesary steps. The number of partial walks in a search can be obtained dividing the length of the total walk by the size of a partial walk: $P = \left\lfloor \frac{L}{s} \right\rfloor = \frac{L-T}{s}$. Then, the expected number of partial walks in a search is $\overline{P} = \frac{\overline{L}-\overline{T}}{s}$.

[3] This is, in fact, a pessimistic assumption. The distribution of trailing steps is approximately uniform, but shorter walks have a slightly higher probability than longer ones. This can be shown analytically and has been confirmed in our experiments (see Appendix A in [11]). Therefore, the expected value in our analysis, derived from a perfectly uniform distribution, is slightly higher than the real average value.

[4] In the following, we make implicit use of the linearity properties of expectations of random variables.

[5] If Y is a random variable with a binomial distribution with success probability p, in which the number of experiments is in turn the random variable X, it can be easily shown that $\overline{Y} = \overline{X} \cdot p$ (see Appendix B in [11]).

Since we assume that the expected number of trailing steps is uniformly distributed between 0 and $(s-1)$, its expectation is $\overline{T} = \frac{s-1}{2}$.

Using the previous equations we have:

$$\overline{L}_s = \left(\frac{s}{2} + \frac{2\overline{L}+1}{2s} - 1\right) + p \cdot \left(\overline{L} - \left(\frac{s}{2} + \frac{2\overline{L}+1}{2s} - 1\right)\right), \tag{2}$$

where the first term is the expectation of the search length for a "perfect" Bloom filter (one that never returns a false positive) and the second term is the expectation of the additional search length due to false positives.

Another interpretation of this expression is obtained if we reorganize it to make explicit the contributions of a perfect filter and of a "broken" filter (one that always returns a false positive result when the resource is not in the filter, i.e., $p = 1$) as

$$\overline{L}_s = \left(\frac{s}{2} + \frac{2\overline{L}+1}{2s} - 1\right) \cdot (1-p) + \overline{L} \cdot p. \tag{3}$$

From this theorem and using calculus, we have the following corollary.

Corollary 1. *The optimal length of the partial walks, i.e., the length of the partial walks that minimizes the expected search length, is:*

$$s_{opt} = \sqrt{2\overline{L}+1}. \tag{4}$$

The obtained value needs to be rounded to an integer, which is omitted in the notation. Observe that *the optimal length of the partial walks is independent from the probability of false positives in the Bloom filters*, while the expected search length ($\overline{L}_s$) does of course depend on it.

Corollary 2. *The optimal expected search length, i.e., the expected search length when partial walks of optimal length are used, is:*

$$\overline{L}_{opt} = \left(\sqrt{2\overline{L}+1} - 1\right)(1-p) + \overline{L}\,p = (s_{opt}-1)(1-p) + \overline{L}\,p. \tag{5}$$

This result is an interesting relation between the optimal length of the search and the optimal length of the PWs. If we consider perfect Bloom filters ($p = 0$), we have $\overline{L}_{opt} = s_{opt} - 1$, which for large $\overline{L}$ (e.g. for large networks) becomes $\overline{L}_{opt} \approx s_{opt}$. Therefore, we have found that, for large N and $p = 0$, *the optimal expected search length approximately equals the optimal length of the partial walks.* For arbitrary values of p, Equation 5 shows that $\overline{L}_{opt}$ is linear in p.

This completes the analysis of choose-first PW-RW. Appendix D in [11] provides an alternative analysis using a different approach. Instead of assuming that the total walk is a random walk, it considers that it is built using the w PWs available at each node, which avoids the need of $\overline{L}$. On the other hand, the alternative model does not provide expressions for $\overline{L}_{opt}$ or s_{opt}.

3.2 Cost of Precomputing PWs

Since searches use the partial walks precomputed by each of the nodes of the network, the cost of this computation must be taken into account. We measure this cost as the number of messages C_p that need to be sent to compute all the PWs in the network. This quantity has been chosen to be consistent with our measure of the performance of the searches. Indeed, each *hop* taken by a search can be alternatively considered as a *message* sent. In addition, C_p is independent from other factors like the processing power of nodes, the bandwidth of links and the load of the network. The cost of precomputing a set of PWs can be simply obtained as $C_p = Nw(s+1)$, since each of the N nodes in the network computes w partial walks, sending s messages to build each of them plus one extra message to get back to its source node.

Let's suppose that each node starts on the average b searches that are processed by the network with the set of PWs precomputed initially. We define C_s to be the total number of messages needed to complete those searches. If the expected number of messages of a search is $\overline{L}_s + 1$ (counting the message to get back to the source node), we have that $C_s = Nb(\overline{L}_s + 1)$. Now, defining C_t as the *average total cost per search*, we can write:

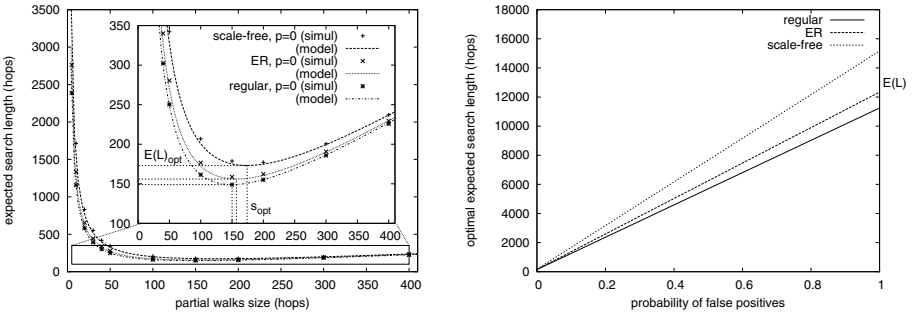
$$C_t = \frac{C_s + C_p}{Nb} = (\overline{L}_s + 1) + \frac{w}{b}(s + 1). \tag{6}$$

The second term in Equation 6 is the contribution to the cost of the precomputation of the PWs. This contribution will remain small provided that the number of searches per node in the interval is large enough.

3.3 Performance Evaluation

The goal of this section is to apply the model for choose-first PW-RW presented in the previous section to real networks, and to validate its predictions with data obtained from simulations. Three types of networks have been chosen for the experiments: regular networks (constant node degree), Erdős-Rényi (ER) networks and scale-free networks (with power law on the node degree). A network of each type and size $N = 10^4$ has been randomly built with the method proposed by Newman et al. [12] for networks with arbitrary degree distribution, setting their average node degree to $\overline{k} = 10$. Each network is constructed in three steps: (1) a preliminary network is constructed according to its type; (2) its degree distribution is extracted, and (3) the final (random) network is obtained feeding the Newman method with that degree distribution. For each experiment, 10^6 searches have been performed, with the source node chosen uniformly at random among the N nodes. Likewise, the resource has been placed in a node chosen uniformly at random for each experiment.

Optimal PW Size and Expected Search Length in Choose-First PW-RW. We start by applying Theorem 1 to the networks described above



(a) **(b)**

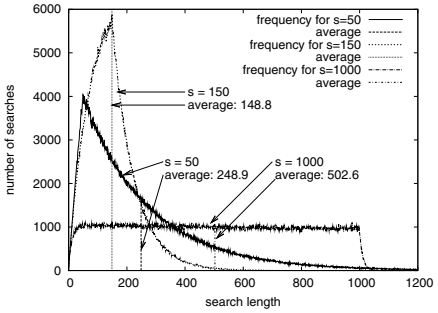
Fig. 2. (a) Expected search length ($\overline{L}_s$) as a function of s when $p = 0$ in a regular network, an ER network and a scale-free network. The optimal points $(s_{opt}, \overline{L}_{opt})$ for each network are $(150, 149)$, $(157, 156)$, and $(174, 173)$. (b) Optimal expected search length ($\overline{L}_{opt}$) as a function of p.

to obtain the expected search length as a function of the size of the PWs.[6] Figure 2(a) provides plots of the expected search lengths ($\overline{L}_s$) given by Equation 1 as a function of the size of the PWs (s), when the probability of a false positive in the Bloom filter is set to $p = 0$, for the three types of networks considered. Results from the analytical model are shown as curves while simulation data are shown as points. The curves for the three networks show a minimum point $(s_{opt}, \overline{L}_{opt})$. This behavior is due to the fact that, when s is small, the number of jumps needed to reach a PW containing the chosen resource grows, therefore increasing the value of $\overline{L}$. In turn, for larger values of s, the number of trailing steps within the last PW grows, also increasing the value of $\overline{L}$.

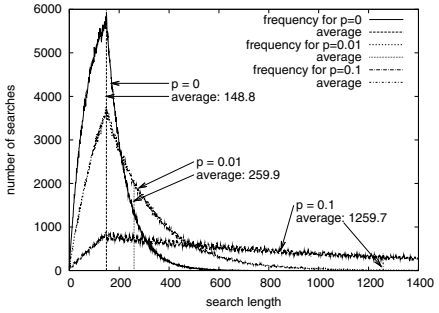
Figure 2(b) shows the linear relation between $\overline{L}_{opt}$ and p (Equation 5). The regular network exhibits the smallest slope, followed by the ER network and then by the scale-free network. For $p = 0$, Equation 5 degenerates to $\overline{L}_{opt} = \overline{L}$, since the search performs all the hops of the total walk (i.e., it is a RW). In fact, Equation 1 also degenerates to $\overline{L}_s = \overline{L}$ in this case, meaning that the expected search length is that of random walk searches regardless the size of the PWs (s).

Distributions of Search Lengths in Choose-First PW-RW. The aim of this section is to experimentally explore how the use of PWs affects the statistical distribution of search lengths.

[6] For each network, the expected length of a random walk search ($\overline{L}$) is needed. We estimate these simulating 10^6 simple random walk searches and averaging their lengths for each network. Average search lengths are denoted in lowercase ($\overline{l}$) to distinguish them from the actual expected value ($\overline{L}$) in the model. The values obtained are: $\overline{l}_{reg} = 11246$, $\overline{l}_{ER} = 12338$, and $\overline{l}_{sf} = 15166$). These results agree with the approximate analytical method in [13] (a modification of the one provided in [5]), which produces the following results: $\overline{l}_{reg} = 11095$, $\overline{l}_{ER} = 12191$, and $\overline{l}_{sf} = 14920$.



(a) Search lengths for $p = 0$ and for $s = s_{opt} = 150$, $s = 50$ and $s = 1000$.



(b) Search lengths for s_{opt} and for $p = 0, 0.01, 0.1$.

Fig. 3. Distributions of search lengths (histograms) with PWs that are not reused in the regular network.

Length Distributions. We first obtain the lengths distributions of searches using PWs that are never reused. Later in this section we will discuss the effect of having a limited number of partial random walks that are reused. We consider each random walk to be the total walk of a search based on PWs. For each original random walk, we break it in pieces of size s, which are taken as the PWs that make up the total walk. Then we consider a search that uses those PWs and count the number of hops (jumps plus trailing steps plus unnecessary steps). This gives the length of the search if it had been constructed using those (precomputed) PWs. Note that the PWs are not reused because they are obtained from independent (real) random walks.

The search length distributions in the regular network for $p = 0$ and for several values of s are shown in Figure 3(a). The average search lengths of each distribution are also shown as vertical bars. These values are very close to the expected values calculated with Equation 1 ($\overline{L}_{50} = 248.9$, $\overline{L}_{150} = 149.0$ and $\overline{L}_{1000} = 510.2$). Therefore, our model accurately predicts average lengths of searches based on PWs of size s in the three types of networks considered.

The shape of the distributions is such that for low s ($s = 50$ in Figure 3(a)) search lengths are dominated by the number of jumps, which is proportional to the length of the total walk. For high s ($s = 1000$ in Figure 3(a)) the distribution adopts a rather uniform shape since search lengths are dominated by the number of trailing steps, assumed to have an approximately uniform distribution between 0 and $s - 1$. The optimal length for the PWs, s_{opt} ($s = 150$ in Figure 3(a)), represents a transition point between these two effects. The shape is such that the values around the average search length (which approximately equals s_{opt}, according to Equation 5) are also the most frequent.

Once it has been found the optimal length for the PWs s_{opt} (known to be independent of p), we investigate the effect of the probability of false positive of Bloom filters in these distributions. Figure 3(b) shows the distributions of search

lengths (histograms) for the regular network when $s = s_{opt}$ and for several values of p. It can be seen that the distributions get wider and lower as p grows, pushing average search lengths to higher values, in accordance with Figure 2(b). However, we observe that the most frequent lengths remain the same regardless of the value of p. For $p = 0$, the most frequent value for each network approximately equals the average search length which, in turn, approximately equals the optimal length of the PWs ($s_{opt} = 150$ for the regular network). For greater values of p, the average search length grows while the most frequent value stays the same. Distributions for the ER and the scale-free networks have similar shapes and are omitted here. However, they have been used in Table 1(a) (explained below).

Effect of Reusing PWs. At this point, we note that we have been assuming that PWs are never reused. However, in practical scenarios it seems quite reasonable to consider a limited number of partial random walks that are reused. In Appendix F of [11] we have explored the distributions of search lengths when the total walks are built reusing a limited number w of PWs precomputed in each node. As it can be readily seen there, we conclude that, for the types of networks in our experiment, just two precomputed PWs per node are enough to obtain searches whose lengths are statistically similar to those that would be obtained with PWs that are not reused. So, we can say that our results using not reused PWs are also valid when using a limited number of PWs that are reused.

Comparison of Performance with Respect to Random Searches. Finally, in Table 1(a) we compare the performance of the proposed mechanism and that of random walk searches. The reduction in the average search length that PW-RW achieves with respect to simple random walks is lower for higher p, ranging from around 98% in the case when $p = 0$ to 88% when $p = 0.1$. Furthermore, we also see that the achieved reductions are independent of the network type.

Table 1. Reductions of average search lengths

(a) PW-RW with respect to random walk searches

	Reduction of $\overline{l}$ (%)		
Network type	$p = 0$	$p = 0.01$	$p = 0.1$
Regular	98.67	97.68	88.73
ER	98.71	97.68	88.42
Scale-free	98.83	97.79	88.43

(b) PW-SAW with respect to PW-RW

	Reduction of $\overline{l}$ (%)		
Network type	$p = 0$	$p = 0.01$	$p = 0.1$
Regular	5.67	8.22	11.24
ER	6.25	9.10	11.88
Scale-free	6.53	9.75	12.65

4 Choose-First PW-SAW

As it was pointed in Section 2 when describing the PW construction mechanism, a possible variation consists of using self-avoiding walks (SAW) instead of RWs.

The resulting mechanism is called PW-SAW. The aim is to revisit less nodes, increasing the chances of locating the resource. In short, a SAW chooses the next node to visit uniformly at random among the neighbors that have not been visited so far by the walk. If all neighbors have already been visited, it chooses uniformly at random among all neighbors, like a simple random walk.

Analysis of Choose-First PW-SAW. When PWs are self-avoiding walks, their concatenation is not a random walk, and hence Theorem 1 is no longer valid. We state a new theorem here for the choose-first PW-SAW mechanism and prove it in Appendix C of [11] using a different approach.

Theorem 2. *If the expected number of trailing steps is assumed to be uniformly distributed in* $[0, s-1]$*, then the expected search length of PW-SAW is*

$$\overline{L}_s = \frac{1}{N} \sum_k n_k \left(\frac{1}{p_{tp}(k)} \cdot (p_n(k) + s \cdot p_{fp}(k)) + \frac{s-1}{2} \right). \tag{7}$$

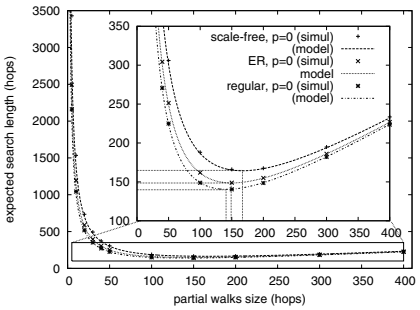
In the above theorem, p_n, p_{tp}, and p_{fp} are the probabilities that the query of the Bloom filter of the chosen PW in the current node returns a (true) negative, a true positive, and a false positive result, respectively, as a funcion of k, the degree of the node holding the resource. The proof in Appendix C of [11] gives expressions for these probabilities.

Expected Search Length in PW-SAW. In this section, we compare the analytic results from the model with experimental data from simulations. Figure 4(a) shows the expected search length ($\overline{L}_s$) as a function of the size of PWs (s) in a regular network, an ER network and a scale-free network, for $p = 0$. The curves in this graph are plotted using the equations in Appendix C of [11]).

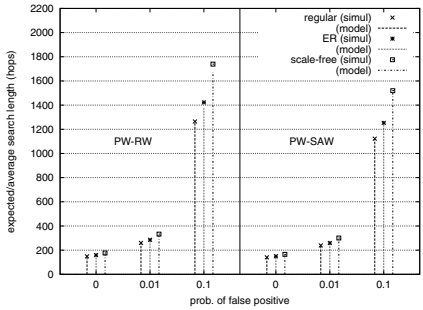
According to the results computed using the PW-SAW model, the minimum search lengths occur for values around $s = 141$, $s = 149$ and $s = 167$ for the regular, ER and scale-free networks, respectively. These values are slightly lower than the ones predicted by the PW-RW model (Figure 2(a)), which were $s_{opt} = 150, 157$ and 174, respectively.

Both the model curves and the simulation experiments have been computed for $w = 5$, chosen as a reference value. However, it has been observed that very similar results are obtained if we change the value of w. Furthermore, plots of the model equations for different values of w are coincident. This behavior was also observed for PW-RW (Section 3.3), where we found that the average search length remained almost constant as we increased w. The reason for this is that the probability of the resource being in the chosen PW does not depend on the number of PWs in the node.

We now compare the results of the PW-RW and PW-SAW mechanisms. Figure 4(b) shows results for PW-RW (left part) and for PW-SAW (right part), in the three networks considered in our study, and for values of $p = 0, 0.01$ and 0.1. Expected search lengths from the analytical models are shown as vertical bars, while average search lengths from the simulations experiments are shown



(a) Vs. s for $p = 0$. Optimal points $(s_{opt}, \overline{L}_{opt})$ are $(141, 139.92)$, $(149, 148.55)$, and $(167, 164.75)$.



(b) Comparison with PW-RW for $p = 0, 0.01, 0.1$.

Fig. 4. Expected search length of PW-SAW in a regular network, an ER network and a scale-free network

as points. The size of the PWs has been set to $s = 150, 157$ and 174 for the regular, ER and scale-free networks, respectively, which are the optimal values predicted by the PW-RW model. For all the networks, we have found a very good correspondence between model predictions and simulation results.

Comparison of Performance with Choose-First PW-RW. The reduction in the average search length that PW-SAW achieves with respect to PW-RW for a given p is largest for the scale-free network, followed by the ER network and then by the regular network. For each network type, the reduction is larger for higher p. Actual values can be found in Table 1(b).

5 Check-First PW-RW and PW-SAW

We now present the check-first versions of the PW-RW and PW-SAW mechanisms, introduced in Section 2. Suppose the search is currently in a node and it needs to pick one of the PWs in that node to decide whether to traverse it or to jump over it. In the check-first mechanisms, it first *checks* the associated resource information of *all* the PWs, and then randomly *chooses* among those with a positive result, if any (otherwise, it chooses among all PWs, as the choose-first version). Performance is improved since the probability of choosing a PW with the resource increases. This comes at the expense of slightly incrementing the processing power used since several PWs need to be checked, but without incurring extra storage space costs.

A minor additional difference between the algorithms is that in the check-first version, the resource information is registered from the *first* node (the node next to the current node) to the *last* node in the PW. This change slightly improves the performance of the new version, since the probability of choosing a PW with the resource increases also in the cases where the resource is held by the last

node of the PW. We have adapted the analysis presented in Section 4 to reflect the new behavior of the check-first mechanisms (see Appendix E of [11]).

Expected Search Length in Check-First PW-RW/PW-SAW. Figure 5 shows the expected search length ($\overline{L}_s$) vs. the size of PWs (s) in a regular network for the four mechanisms presented, for $p = 0.01$ and $w = 5$. The check-first mechanisms achieve a lower minimum expected search length than the original choose-first mechanisms, as expected. In fact, the expected search length can be lowered further by increasing w, the number of PWs per node, clearly at the expense of increasing the cost of the PWs construction stage. In addition, the minimum expected search length occurs for significantly lower s (s_{opt} falls from 150 to about 50), meaning shorter PWs in the nodes, which in turn decreases the cost of the PWs construction stage. As for the PW-SAW mechanisms, we note that both versions achieve a slight decrease in the expected search length with respect to their PW-RW counterparts (which was already observed in Table 1). Results for the ER and scale-free networks are similar and are omitted here.

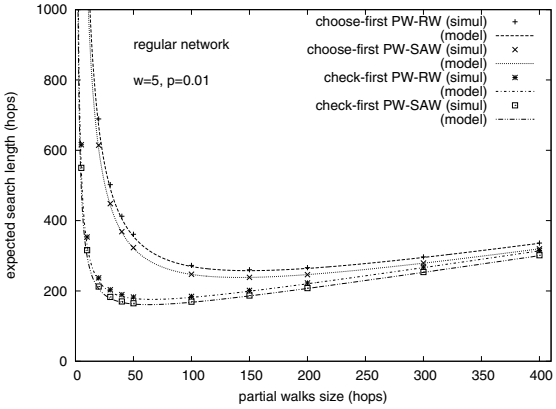


Fig. 5. Expected search length of choose-first and check-first versions of PW-RW and PW-SAW as a function of s in a regular network for $p = 0.01$ and $w = 5$

6 Future Work

The proposed mechanisms could be improved with new strategies to choose from the PWs at the nodes. Smarter variants of RWs could be used as PWs. It would be interesting to compare their application to unstructured P2P networks with algorithms for structured overlays like DHT or quorum systems.

References

1. Lada, A., Adamic, R.M., Lukose, A.R.: Puniyani, and Bernardo A. Huberman. Search in power-law networks. Physical Review E 64(046135) (2001)
2. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 16th International Conference on Supercomputing, ICS 2002, pp. 84–95. ACM, New York (2002)
3. Yang, S.-J.: Exploring complex networks by walking on them. Physical Review E 71(016107) (2005)
4. Gkantsidis, C., Mihail, M., Saberi, A.: Random-walks in peer-to-peer networks: algorithms and evaluation. Performance Evaluation 63, 241–263 (2006)
5. Rodero-Merino, L., Fernández Anta, A., López, L., Cholvi, V.: Performance of random walks in one-hop replication networks. Computer Networks 54(5), 781–796 (2010)
6. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N., Braynard, R.: Networking named content. Commun. ACM 55(1), 117–124 (2012)
7. Broder, A., Mitzenmacher, M.: Network applications of bloom filters: A survey. Internet Mathematics 1(4), 485–509 (2004)
8. Sarma, A.D., Nanongkai, D., Pandurangan, G., Tetali, P.: Distributed random walks. J. ACM 2, 2:1–2:31 (2013)
9. Hieungmany, P., Shioda, S.: Characteristics of random walk search on embedded tree structure for unstructured p2ps. In: International Conference on Parallel and Distributed Systems, pp. 782–787 (2010)
10. Millán, V.M.L., Cholvi, V., López, L., Fernández Anta, A.: Resource location based on partial random walks in networks with resource dynamics. In: Proceedings of the 4th International Workshop on Theoretical Aspects of Dynamic Distributed Systems, TADDS 2012, pp. 26–31. ACM, New York (2012)
11. Millán, V.M.L., Cholvi, V., López, L., Fernández Anta, A.: Improving resource location with locally precomputed partial random walks. arXiv:1304.5100 (2013)
12. Newman, M.E.J., Strogatz, S.H., Watts, D.J.: Random graphs with arbitrary degree distributions and their applications. Physical Review E 64(026118) (2001)
13. Millán, V.M.L., Cholvi, V., López, L., Fernández Anta, A.: A model of self-avoiding random walks for searching complex networks. Networks 60(2), 71–85 (2012)