Available online at www.sciencedirect.com

ScienceDirect

Information Processing and Management 44 (2008) 1611–1623

INFORMATION PROCESSING & MANAGEMENT

www.elsevier.com/locate/infoproman

A new variant of the Pathfinder algorithm to generate large visual science maps in cubic time

A. Quirin [a,*], O. Cordón [a], J. Santamaría [b], B. Vargas-Quesada [c], F. Moya-Anegón [c]

[a] *European Centre for Soft Computing, Edificio Científico Tecnológico, 33600 Mieres, Spain*
[b] *Department of Software Engineering, University of Cádiz, Cádiz, Spain*
[c] *SCImago Group, Library and Information Science Faculty, University of Granada, 18071 Granada, Spain*

Received 16 April 2007; received in revised form 3 September 2007; accepted 8 September 2007
Available online 24 October 2007

Abstract

In the last few years, there is an increasing interest to generate visual representations of very large scientific domains. A methodology based on the combined use of ISI–JCR category cocitation and social networks analysis through the use of the Pathfinder algorithm has demonstrated its ability to achieve high quality, schematic visualizations for these kinds of domains. Now, the next step would be to generate these scientograms in an on-line fashion. To do so, there is a need to significantly decrease the run time of the latter pruning technique when working with category cocitation matrices of a large dimension like the ones handled in these large domains (Pathfinder has a time complexity order of O(n^4), with n being the number of categories in the cocitation matrix, i.e., the number of nodes in the network).

Although a previous improvement called Binary Pathfinder has already been proposed to speed up the original algorithm, its significant time complexity reduction is not enough for that aim. In this paper, we make use of a different shortest path computation from classical approaches in computer science graph theory to propose a new variant of the Pathfinder algorithm which allows us to reduce its time complexity in one order of magnitude, O(n^3), and thus to significantly decrease the run time of the implementation when applied to large scientific domains *considering the parameter* $q = n - 1$. Besides, the new algorithm has a much simpler structure than the Binary Pathfinder as well as it saves a significant amount of memory with respect to the original Pathfinder by reducing the space complexity to the need of just storing two matrices. An experimental comparison will be developed using large networks from real-world domains to show the good performance of the new proposal.

Keywords: PFNETs; Pathfinder algorithms; Cocitation analysis; Information visualization; Large scientific domain visual maps; Graph shortest path algorithms

* Corresponding author. Tel.: +34 985456545; fax: +34 985456699.
E-mail addresses: arnaud.quirin@softcomputing.es (A. Quirin), oscar.cordon@softcomputing.es (O. Cordón), jsantam@uca.es (J. Santamaría), benjamin@ugr.es (B. Vargas-Quesada), felix@ugr.es (F. Moya-Anegón).

0306-4573/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2007.09.005

1. Introduction

The goal of generating schematic visualizations for scientific domain analysis has been pursued since several decades ago and different approaches have been used to put it into effect (Borner, Chen, & Boyack, 2003; Buzydlowski, 2002; Chen, 1999; Lin, White, & Buzydlowski, 2003; White, 2003). Their good performance have made the size of the tackled domain progressively increase, with the final aim of being able to represent the largest possible one, the World (Boyack, Klavans, & Borner, 2005; Leydesdorff, 2004b; Leydesdorff, 2004a; Samoylenko, Chao, Liu, & Chen, 2006).

In 1998, Chen (1998a, 1998b) was the first researcher to bring forth the use of Pathfinder Networks (PFNETs) in citation analysis. Since then, it has been used for the study and representation of minor domains or scientific community. In 2004, Moya-Anegón et al. (2004) proposed the combination of PFNET and ISI categories cocitation, making possible the depicting and analysis of large scientific domains in an easy way. The scientific community is understood in the terms put forth by Hjorland and Albrechtsen (1995), as the reflection of interactions between authors, and their role in science, through citation (i.e., classical author cocitation analysis). The new technique is based on the use of thematic classification since categories taken from the ISI–JCR are considered as entities of cocitation and units of measure (Moya-Anegón et al., 2005; Moya-Anegón et al., 2006; Vargas-Quesada & Moya-Anegón, 2007). The cocitation matrix is then treated as a graph which represents a social network of the existing relations and processed through social network analysis: the graph is pruned by means of the Pathfinder algorithm (Dearholt & Schvaneveldt, 1990) to get a PFNET, keeping just the most salient relations, and the resulting graph is graphically represented using a graph drawing algorithm, Kamada–Kawai (Kamada & Kawai, 1989).

So, once an appropriate methodology has been designed to graphically represent very large scientific domains, the next challenge is to build them in a very small amount of time, allowing us to generate the scientograms on line. If this goal is finally achieved, these kinds of visual science maps could be used to design an information retrieval system, composing an *Atlas of Science* as the one that is being implemented by Felix de Moya's Scimago research group for the IberoAmerican scientific production[1].

The key problem to generate scientograms of large scientific domains by means of the Pathfinder algorithm is the great time and space complexity it requires. As we will see later, the pruning it applies is based on eliminating those links which violate the triangle inequality (Schvaneveldt, 1990). To do so, there is a need to compute a progressive series of q matrices D^i of dimension n^2 which store the shortest paths between each pair of entities (graph nodes) considering paths comprised by as much q links. Moreover, their computation requires the use of an additional series of q auxiliary matrices W^i. This way, as a value of q equal to $n - 1$ is required in order to achieve an appropriate pruning in large scientific domains keeping only the most salient links, the resulting time and space complexity of the Pathfinder algorithm are $O(n^4)$ and $O(n^3)$ (in fact, $2 \cdot (n - 1)$ matrices of dimension n^2 are stored), respectively. Since the value of n is high in the large scientific domains handled, we come up to the undesired conclusion that the run time of the algorithm is prohibitive to generate the maps on-line.

We should note that a previous attempt was made in this aim by Guerrero-Bote et al., which recently proposed an improved variant of the original Pathfinder algorithm, called Binary Pathfinder (Guerrero-Bote, Zapico-Alonso, Espinosa-Calvo, Gómez-Crisóstomo, & Moya-Anegón, 2006), that reduced its time complexity for the current case to $O(\log n \cdot n^3)$. However, although the reduction is very significant, it is not enough to allow us to generate the maps "on the fly" since, for values of n around 250, as those handled in our very large domains, the run of the Binary Pathfinder takes several seconds, and this amount of time is then increased by that corresponding to Kamada–Kawai's layout algorithm.

In this contribution, we introduce *Fast Pathfinder*, a new Pathfinder variant taking as a base a classical algorithm in graph theory, Floyd–Warshall's (Cormen, Leiserson, Rivest, & Stein, 2001), to compute the shortest paths in the graph in a different way. Thanks to that and to the fact that we fix the value of q to $n - 1$, we are able to reduce the time complexity of the original algorithm in one order of magnitude, $O(n^3)$, which is a killer advantage when applied to the generation of scientograms for large scientific domains.

[1] http://www.atlasofscience.net/.

Moreover, the new algorithm has a much simpler structure than Binary Pathfinder, since it only requires three loops wrapping two simple operations, as well as it only requires two squared matrices to operate. An experimental comparison will be developed using large networks from real-world domains corresponding to the scientific production of different countries to show the good performance of the new proposal in comparison with both the original and the Binary Pathfinder.

To do so, the paper is structured as follows. Section 2 briefly reviews the original Pathfinder and the Binary Pathfinder algorithms. The new proposal is introduced in Section 3, together with a detailed analysis of its advantages in terms of speed, memory saving and simplicity. Section 4 collects the experiments developed to test Fast Pathfinder. Finally, some concluding remarks are pointed out in Section 5.

2. Preliminaries

This section is devoted to introduce the preliminaries needed to achieve a good understanding of our proposal. With this aim, the next two subsections respectively describe the original Pathfinder and the Binary Pathfinder algorithms.

2.1. The Pathfinder algorithm

Pathfinder was introduced by Dearholt and Schvaneveldt as a technique to choose the shortest links in a network in the field of social networks (Dearholt & Schvaneveldt, 1990). The result of the Pathfinder procedure is a pruned network called PFNET – which is either a directed or undirected graph depending on the fact that the original similarity matrix is symmetrical or not – that only keeps those links which do not violate the triangle inequality stating that the direct distance between two nodes must be lesser than or equal to the distance between them passing through any group of intermediate nodes. As said by its creators, PFNETs provide unique representations of the underlying structure for domains in which objective measures of distance are available (Schvaneveldt, 1990).

The Pathfinder algorithm is based on two main parameters:

1. $r \in [1, \infty]$, which defines the adaptive metric, the *Minkowski r-metric*, considered to measure the distance between two network nodes not directly connected:

$$D = \left\{ \sum_i d_i^r \right\}^{\frac{1}{r}} \tag{1}$$

   When r takes value 1, the Minkowski metric results in the sum of the link weights; when it takes value 2, it becomes the usual Euclidean metric; and when r tends to ∞, the path weight is the same as the maximum weight associated with any link along the path.
2. $q \in [2, n-1]$ (with n being the number of nodes in the network), which limits the number of links in the paths for which the triangle inequality is ensured in the final PFNET. Hence, every path connecting two nodes that violate the triangle inequality, having an associated Minkowski distance greater than any other path between the same two nodes composed of up to q links, will be removed.

Note that $r = \infty$ and $q = n - 1$ are the common parameter values when Pathfinder is used for large domains scientogram generation. These values are very advantageous for large network pruning (Chen, 2004).

To build a PFNET, two different kinds of auxiliary matrices are used:

– W^i_{jk}, which stores the minimum cost to go from node j to node k by following exactly i links. This matrix is computed recursively using matrix W^{i-1}_{jk}, with W^1 being the original weight matrix.
– D^i_{jk}, which stores the minimum cost to go from node j to node k by following any path in the network composed of *i or less* links. This matrix is computed recursively using matrices $W^1_{jk}, \ldots, W^i_{jk}$.

```
1. Compute $W^{i+1} = W \odot W^i$, as follows: $w_{jk}^{i+1} = \mathrm{MIN}((w_{jm})^r + (w_{mk}^i)^r)^{1/r}$, for
   $1 \le m \le n$.
2. Compute $D^i$, as follows: $d_{jk}^i = \mathrm{MIN}(w_{jk}^1, \ldots, w_{jk}^i)$, for $j \neq k$.
3. Iterate until $W^q$ and $D^q$ are computed.
4. Compare $W^1$ and $D^q$ : all the links having the same values in these two matrices
   will belong to the final PFNET.
```

Fig. 1. The original Pathfinder algorithm.

The original Pathfinder algorithm pseudocode is shown in Fig. 1.

Notice that the algorithm has a time complexity order $O(q \cdot n^3)$ as q steps have to be done to build the q matrices W^i and D^i. Each of the latter matrices stores n^2 weights, so a loop of this order is needed to compute them in each step. Finally, an additional loop of n steps is needed to compute each component of W^{i+1}, as seen in line 1 of the algorithm. As the maximum possible value for q is $n - 1$, Pathfinder has a time complexity of $O(n^4)$ in that case.

On the other hand, the resulting space is thus of complexity $O(q \cdot n^2)$ ($O(n^3)$ when $q = n - 1$), since there is a need to build q matrices W^i and other q matrices D^i, as seen above.

2.2. The Binary Pathfinder algorithm

Guerrero-Bote et al. (2006) recently proposed the *Binary Pathfinder* algorithm, an improved variant of the original Pathfinder aiming at reducing its time and space complexity. Binary Pathfinder takes the following two aspects as a base to put this improvement into effect:

1. The only matrix in the series of D^i that is actually needed for the algorithm to operate is the last one, D^q, to be compared with the initial weight matrix W^1. The remainder are not necessary.
2. The matrices D^i can be directly generated from two previous ones in the same way as done for the consecutive W^i matrices: $D^{i+j} = D^i \odot D^j$.

Hence, the authors demonstrated that the distance matrix D^{i+j} storing the minimum distances between each couple of nodes can be calculated from D^i and D^j as follows:

$$d_{kl}^{i+j} = \mathrm{MIN}\{d_{kl}^i, d_{kl}^j, ((d_{km}^i)^r + (d_{ml}^j)^r)^{1/r}\} \tag{2}$$

where $d_{kl}^1 = w_{kl}$, obtaining the same result as with the original Pathfinder algorithm described in the previous subsection.

Thanks to the latter, a new Pathfinder algorithm was designed which does not need to compute every D^i matrix, $i = 1, \ldots, q$, but can make larger steps. Taking the procedure to transform an integer number to binary as a base (that is the inspiration for the algorithm's name), Guerrero-Bote et al.'s Binary Pathfinder reduces the task to calculating just $\log(q)$ matrices, those corresponding to indices being powers of 2: D^1, D^2, D^4, $D^8, \ldots$

The Binary Pathfinder algorithm pseudocode is shown in Fig. 2. Notice that the principle loop reduces the number of steps of the original Pathfinder from q to $\log q$. Therefore, the time complexity of the new Binary Pathfinder variant becomes $O(\log q \cdot n^3)$ instead of $O(q \cdot n^3)$, which in the maximum case becomes $O(\log n \cdot n^3)$ instead of $O(n^4)$, a very significant time difference for large networks. Empirical tests showing these differences on real cases are shown in Guerrero-Bote et al. (2006) and in Section 4 of the current paper. On the other hand, the space complexity is even more significantly reduced, as only two squared matrices to compute D^i in each step, another matrix to store the final distance values D^q, and one last matrix W to store the original weights are required, instead of $2 \cdot q$ matrices W^i and D^i, as in the original algorithm.

```
 1. $i = 1$; $nq = 0$; Generate $D^1 = W$; $D^q \leftarrow \infty$.
 2. IF ($q \mod 2 = 1$) THEN Compute $D^q = D^q \odot D^1$.
 3. $nq = 1$.
 4. WHILE ($2 \cdot i \leq q$)
 5.         Compute $D^{2 \cdot i} = D^i \odot D^i$.
 6.         IF (($q - nq) \mod (4 \cdot i) > 0$) THEN
 7.                 Compute $D^q = D^q \odot D^{2 \cdot i}$.
 8.                 $nq = nq + 2 \cdot i$.
 9.         $i = 2 \cdot i$.
10. Compare $W^1$ and $D^q$ : all the links having the same values in these two matrices
    will belong to the final PFNET.
```

Fig. 2. The Binary Pathfinder algorithm.

3. Fast Pathfinder

As we have seen in the previous section, the Binary Pathfinder approach is able to achieve an important speed up of the Pathfinder algorithm. Unfortunately, this time complexity reduction, although significant, is not enough for the aim of generating scientograms of very large scientific domains in an on-line fashion since, for values of n around 250 and for $q = n - 1$, the run of the Binary Pathfinder still takes several seconds (see Section 4).

In this section, we introduce *Fast Pathfinder*, a new variant of the Pathfinder algorithm which tries to solve the latter problem. To do so, we first analyze the underlying idea of this approach, which is based on the use of classical algorithms in graph theory for shortest path computation. In fact, the new variant is based on the idea that a PFNET can be obtained with a Shortest Path algorithm when $q = n - 1$. Then, we introduce the Fast Pathfinder's pseudocode and analyze its main advantages and its only disadvantage.

3.1. Underlying idea: graph shortest path computation algorithms

As we need to fix the value of q to $n - 1$, the triangle inequality is verified for the best path between any couple of nodes in the graph, thus the problem becomes a shortest path problem. This is why we can replace steps 1–3 in the original Pathfinder algorithm (see Fig. 1) to achieve the same result in less computation time. When analyzing the operation mode of this algorithm from a computer science point of view, one can recognize that what it does is nothing but computing a distance matrix D^{n-1} storing the lengths of all the shortest paths (regarding the Minkowski r-metric) between any pair of network nodes comprised by up to $n - 1$ links, and then comparing the latter values to the original weights in matrix W^1 to determine which links will finally belong to the PFNET.

To do so, it applies the classical *dynamic programming* approach in algorithm theory (Cormen et al., 2001) in order to ensure the obtaining of the optimal solution for the graph shortest path problem. Dynamic programming (Dreyfus, 1965) constitutes the practical embodiment of the Bellman's principle of optimality (Bellman & Kalaba, 1965) through a clever ("moon walking" type) technique for computing optimal sequential-decisions by a forward-looking, backward-recursive search. Hence, the Pathfinder algorithm is a direct instance of the latter algorithmic methodology, that applies the usual bottom-up approach based on a progressively increasing building of the matrices ensuring to take the best decision at each step, taking into account all the partial decisions made in the previous ones. This results in the Pathfinder algorithm structure where, to build the matrices W^i and D^i of dimension n^2 in each of the $n - 1$ steps, an additional loop of size n is required to check all the possible choices of crossing a link for the shortest path computation between two nodes. All of the latter defines the O(n^4) time complexity.

Notice that Binary Pathfinder keeps the same algorithmic approach than the original Pathfinder version, and the improvement introduced is due to the fact that it smartly reduces the number of steps in the outer loop needed to compute the same distance matrix D^{n-1} while still satisfying the Bellman's principle of optimality.

Hence, as seen in Binary Pathfinder, the only two matrices that are finally needed to obtain the PFNET as a result of pruning the original network are D^{n-1} and W^1. As we know that D^{n-1} is a shortest path distance matrix when $q = n - 1$, we can borrow alternative (and quicker ways) to compute it from the classical algorithms in graph theory (Cormen et al., 2001). In fact, there are at least two classical graph shortest path algorithms, respectively called *Floyd–Warshall*'s and *Dijkstra*'s and also based on the dynamic programming approach, that are able to compute all the shortest paths of length up to $n - 1$ links (according to an Euclidean metric) in a cubic time complexity. The adaptation of Floyd–Warshall's algorithm to the computation of the D^{n-1} matrix for a PFNET using the Minkowski r-metric is thus the base of our new Fast Pathfinder proposal.

3.1.1. Floyd–Warshall's shortest path algorithm

Floyd–Warshall's algorithm (Floyd, 1962; Warshall, 1962) is a dynamic programming algorithm giving the shortest paths between any source node and any destination node in a directed graph in cubic time. The algorithm computes, for each pair of nodes i and j, the minimum weight among all paths between them, storing it into a distance matrix $D = d_{ij}$. The weight of a path between two nodes is the sum of costs of the links in that path. Additionally, a predecessor matrix P can be used to retrieve the links composing the shortest paths themselves, where p_{ij} corresponds to the index of the last node included in the optimal path from i to j.

The basic Floyd–Warshall's algorithm pseudocode is shown in Fig. 3. When the predecessor matrix is to be computed, it becomes the pseudocode shown in Fig. 4.

It is very simple to check that both variants of the algorithm have a time complexity of O(n^3).

3.1.2. Dijkstra's shortest path algorithm

Dijkstra's algorithm (Dijkstra, 1959) is both a greedy and a dynamic programming algorithm that solves the single-source shortest path problem for a directed graph with nonnegative link weights. As in Floyd–Warshall's technique, the cost of a path between two nodes is the sum of costs of the links in that path. The algorithm gives the costs of the shortest paths from a single, fixed node s to all the other nodes in the graph in

```
1. $D \leftarrow W$.
2. FOR $k$ from 1 to $n$ DO
3.         FOR $i$ from 1 to $n$ DO
4.                 FOR $j$ from 1 to $n$ DO
5.                         $d_{ij} = \mathrm{MIN}(d_{ij}, d_{ik} + d_{kj})$.
```

Fig. 3. Basic Floyd–Warshall's algorithm.

```
1. $D \leftarrow W$.
2. FOR $i$ from 1 to $n$ DO
3.         FOR $j$ from 1 to $n$ DO
4.                 IF ($d_{ij} \neq \infty$) THEN $p_{ij} = i$ ELSE $p_{ij} = \infty$.
5. FOR $k$ from 1 to $n$ DO
6.         FOR $i$ from 1 to $n$ DO
7.                 FOR $j$ from 1 to $n$ DO
8.                         sum $= d_{ik} + d_{kj}$.
9.                         IF ($d_{ij} >$ sum)
10.                        THEN $d_{ij} =$ sum; $p_{ij} = p_{kj}$.
```

Fig. 4. Floyd–Warshall's algorithm using the predecessor matrix.

```
1. $V[G] \leftarrow$ Set of graph nodes.
2. FOR each node $v \in V[G]$ DO
3.         $d_v = \infty$; $p_v = \emptyset$.
4. $d_s = 0$; $E \leftarrow \emptyset$; $F \leftarrow V[G]$.
5. WHILE ($F \neq \emptyset$)
6.         $u =$ EXTRACT-MIN($F$); $E \leftarrow E \cup \{u\}$.
7.         FOR each node $v \in$ NEIGHBORS($u$) DO
8.                 IF ($d_v > d_u + w_{uv}$) THEN $d_v = d_u + w_{uv}$; $p_v = u$.
```

Fig. 5. Dijkstra's algorithm.

quadratic time. To retrieve the paths, it uses the same kind of predecessor data structure (an array *p*, in this case) than Floyd–Wharshall's algorithm (see the previous subsection). In order to get the all-pairs shortest paths, there is a need to wrap Dijkstra's algorithm into a linear loop for all the graph nodes, thus resulting in a time complexity of O(n^3).

Dijkstra's algorithm pseudocode is shown in Fig. 5, where $V[G]$ is the set of graph nodes, F is a set of unvisited nodes by the algorithm, and $u = EXTRACT - MIN(F)$ returns the node u with the lowest distance value in F and removes that node from it. On the other hand, E is the set of already visited nodes.

In its simplest implementation, a normal array is used to store the links, and thus operation EXTRACT-MIN is simply a linear search through all nodes in F. In this case, the time complexity is O($m \cdot n$), with m being the number of the s node links. As the maximum number of links for any node in the graph is $n - 1$, the maximum time complexity becomes O(n^2).

We should also notice that there are more efficient implementations of Dijkstra's algorithm for the case of sparse graphs with a number of links significantly lower than n^2. They are based on storing the graph in the form of an adjacency list and using more advanced data structures than a simple list. With a heap in the EXTRACT-MIN function, the time complexity becomes O($(m + n) \cdot \log n$). When a Fibonacci heap is considered, it becomes O($m + n \cdot \log n$).

3.2. Structure of Fast Pathfinder

Taking into account what has been explained in the latter subsection, Floyd–Warshall's algorithm has been selected to substitute the costly computation of matrix D^{n-1} in the original Pathfinder algorithm. Dijkstra's algorithm is not as well suited to do so. The reasons are mainly related to the structure of the graphs and the greater simplicity of the Floyd–Warshall's algorithm implementation and will be detailed at the end of this section.

Since working with Floyd–Warshall's algorithm we are able to build this matrix in cubic time and we avoid the need to compute the temporary matrices W^i and D^i, the substitution is much more effective. To do so, there is a need to only perform one trivial change to the pseudocode shown in Section 3: the Minkowski r-metric has to be used to compute the path lengths. In this way, we can directly substitute the three first lines of the Pathfinder algorithm in Fig. 1 by the five lines of the basic Floyd–Warshall's pseudocode in Fig. 3, with the previous adaptation. The resulting Fast Pathfinder pseudocode is shown in Fig. 6.

Since the shortest path computation procedure has an O(n^3) time complexity and the W–D comparison takes time O(n^2), the algorithm will have a time complexity of O(n^3) + O(n^2) = max {O(n^3),O(n^2)} = O(n^3). Besides, notice that the computation of the predecessor matrix is not needed and the algorithm only requires to store two square matrices to operate (W and D).

On the other hand, there is another alternative for the PFNET link selection (lines 6–8). Actually, by using Floyd–Warshall's algorithm for the shortest path computation, there is not a need to compare the distance matrix D to the original weight matrix W to select the PFNET links, but those links can be directly extracted from the predecessor matrix P. In this second way, we used this matrix as an adjacency matrix to improve the speed of the computation. At the start of the algorithm, each link is considered belonging to a possible shortest

```
1. $D \leftarrow W$; $PFNET \leftarrow \emptyset$.
2. FOR $k$ from 1 to $n$ DO
3.         FOR $i$ from 1 to $n$ DO
4.                 FOR $j$ from 1 to $n$ DO
5.                         $d_{ij} = \mathrm{MIN}\left\{d_{ij}, ((d_{ik})^r + (d_{kj})^r)^{1/r}\right\}$.
6. FOR $i$ from 1 to $n$ DO
7.         FOR $j$ from 1 to $n$ DO
8.                 IF $(d_{ij} = w_{ij})$ THEN $PFNET \leftarrow PFNET \cup (i,j)$.
```

Fig. 6. The Fast Pathfinder algorithm.

```
1. $D \leftarrow W$; $PFNET \leftarrow \emptyset$.
2. FOR $i$ from 1 to $n$ DO
3.         FOR $j$ from 1 to $n$ DO
4.                 $p_{ij} = \mathit{true}$.
5. FOR $k$ from 1 to $n$ DO
6.         FOR $i$ from 1 to $n$ DO
7.                 FOR $j$ from 1 to $n$ DO
8.                         sum $= ((d_{ik})^r + (d_{kj})^r)^{1/r}$.
9.                         IF $(d_{ij} >$ sum$)$
10.                        THEN $d_{ij} =$ sum; $p_{ij} = \mathit{false}$.
11. FOR $i$ from 1 to $n$ DO
12.        FOR $j$ from 1 to $n$ DO
13.                IF $(p_{ij} = \mathit{true})$ THEN $PFNET \leftarrow PFNET \cup (i,j)$.
```

Fig. 7. The Fast Pathfinder algorithm variant, using a predecessor matrix.

path, thus to the PFNET, so the corresponding component in P is set to *true*. Then, each time the distance d_{ij} of a link (i,j) is greater than the distance of the path (i,k,j), the link (i,j) is discarded from the PFNET and the corresponding component in the matrix is set to *false*. The final value of P indicates exactly those links that must be preserved. Hence, we don't need to take matrix W into account to select the links for the PFNET. The pseudocode of this second Fast Pathfinder algorithm variant is shown in Fig. 7.

This second way of selecting the PFNET links has also an O(n^2) time complexity, so the time complexity of the second Fast Pathfinder variant is still O(n^3). However, its actual running time would be slightly larger than the former version because of the computation time needed to generate the predecessor matrix. We will experimentally check this assumption in the next section.

Concerning its space complexity, it is the same than the former, since it also requires to store two square matrices to operate: D, the same that the other variant, and the predecessor matrix P (in the place of the weight matrix W).

Finally, we should also note that two new variants could also be designed in case Dijkstra's algorithm would have been considered instead of Floyd–Warshall for the shortest path matrix computation. Besides, at first sight, it could seem that a lower time complexity Fast Pathfinder could be obtained proceeding in that way by means of the advanced implementation of the former algorithm based on the use of the Fibonacci heap.

However, we have decided not to consider Dijkstra's variants due to two main reasons. On the one hand, it is well known in algorithm theory that Dijkstra's algorithm is quicker than Floyd–Warshall's for the case of sparse graphs, i.e., when the number of links in the graph $|A|$ tends to the number of nodes n, while the opposite holds for dense graphs, i.e., when $|A| \rightarrow n^2$. Since the graphs resulting from cocitation matrices associated to large scientific domains are actually “small worlds” (Watts & Strogatz, 1998; Watts, 2004), they are very dense and thus Floyd–Warshall’s proposal is the best choice. On the other hand, the use of the latter variant is also beneficial since it results in a simpler implementation of the Fast Pathfinder algorithm.

3.3. Main advantages and disadvantage of Fast Pathfinder

In summary, the Fast Pathfinder proposal introduced in the current contribution based on Floyd–Warshall’s shortest path algorithm has the two following advantages associated:

1. *Speed Increase:* Thanks to the change in the shortest path distance matrix computation, we are able to reduce the time complexity of the original algorithm in one order of magnitude when q is fixed to $n-1$, from $O(n^4)$ to $O(n^3)$, which is a great advantage when applied to large networks and, specifically, for the generation of scientograms of large scientific domains. In this way, it is also significantly lower than the quickest Pathfinder variant, Binary Pathfinder ($O(\log n \cdot n^3)$ when $q = n-1$).
2. *Simplicity:* Moreover, the new algorithm has a much simpler structure than the previous approach reducing the original Pathfinder run time, Binary Pathfinder, since it only requires three loops wrapping two simple operations. On the other hand, Fast Pathfinder significantly reduces the space complexity since it only requires two square matrices to operate instead of the $2 \cdot n - 1$ ones needed by the original algorithm and the four ones by Binary Pathfinder.

On the other hand, its only disadvantage with respect to Binary Pathfinder is that while in our case the value of the parameter q is always fixed to $n-1$, the latter algorithm allows any possible value for q. Of course, this restricts the generic applicability of Fast Pathfinder, but we should remind that it has been specifically proposed for the on-line generation of large scientific domain visual maps. Note also that any valid value for the second main parameter r can be considered.

4. Experimental results

In the current section, some experiments will be developed to test the actual run time improvement obtained by our proposal, and to empirically prove that it always achieves the same result as the original algorithm. To test the run time improvement, we have compared our two Fast Pathfinder proposals, the one considering the same link selection procedure than Pathfinder, and the other making use of the predecessor matrix for this task (see Section 3.2), with respect to the current state-of-the-art Pathfinder variants, the original algorithm and Binary Pathfinder. To do so, since the aim to propose this algorithm was to use it for the design of scientograms of large scientific domains, we have applied the four algorithms to 20 real networks of this kind, obtained from the JCR category cocitation information available at the Scimago research group’s *Atlas of Science* (http://www.atlasofscience.net/). Their sizes range from 212 to 263 nodes, and from 8485 to 23430 links. Notice that, the link weights in this graph correspond to similarities instead of distance measurements[2].

In order to do a fair comparison, the original and Binary Pathfinder implementations considered are the same ones used by the Binary Pathfinder’s authors. Our two Fast Pathfinder variants have also been implemented in C. The four algorithms have been compiled with the GNU `GCC` compiler with the `-O3` option, under the Linux operating system, and run in an Intel dual-core Pentium 3.4 GHz computer with 2 GB of memory. Pathfinder parameters have been set to $q = n-1$ (when considered) and $r = \infty$, the typical values

[2] According to Moya’s methodology (Moya-Anegón et al., 2004; Vargas-Quesada & Moya-Anegón, 2007), the normalized cocitation coefficients are used and correspond to similarities. The interested reader can refer to that paper for more details. Actually, using similarities or distances has no influence at all in our proposal. In case of using similarities, we would only need to replace MIN by MAX, ’>’ by ’<’, and use $r = -\infty$ to mimic the MIN function instead of the MAX function in the Fast Pathfinder algorithm (see Figs. 6 and 7).

Table 1
Comparison of all the algorithms (computation times are expressed in milliseconds on an Intel dual-core 3.4 GHz CPU with 2 GB of memory), sorted by the number of nodes then by the number of links

#	Domain (year)	# Nodes	# Links	Original PF	Binary PF	Fast PF (predecessor)	Fast PF
1	China (2002)	212	8541	37644.78	2544.5	103.8	92.0
2	Japan (2002)	213	9028	27041.28	2288.76	93.6	85.4
3	France (2002)	216	10087	30105.24	2909.78	97.2	93.8
4	Peru (2002)	218	8485	41196.14	2866.12	126.4	102.6
5	Germany (2002)	218	11745	33631.5	2099.06	118.8	105.4
6	UK (2002)	218	13567	50484.46	2147.4	116.4	94.0
7	Europe (2002)	218	17242	53357.56	2169.38	122.2	94.6
8	USA (2002)	218	18132	54046.88	2195.02	110.6	95.2
9	World (2002)	218	20154	37976.02	2178.64	122.4	97.4
10	Cuba (2004)	219	10644	45319.38	2065.04	123.0	97.0
11	Spain (1994)	219	13478	49800.4	3022.02	122.8	101.0
12	Cuba (2006)	221	11286	33784.98	2813.92	107.6	106.6
13	Spain (1998)	223	16226	44860.4	2854.34	130.0	122.2
14	Venezuela (2005)	239	15415	50741.26	4248.92	148.0	135.0
15	Spain (2002)	240	19183	77421.34	4723.66	154.0	145.8
16	Spain (2004)	240	23430	56890.9	4716.78	192.64	142.2
17	Chile (2004)	242	17914	56025.26	2928.4	150.0	132.8
18	Mexico (2005)	250	21264	100131.76	5569.52	181.4	155.0
19	Portugal (2005)	254	22179	79733.1	4767.96	203.2	163.6
20	Argentina (2005)	263	19562	110309.5	5447.66	194.8	166.4

in large domain scientogram design. Fifty independent runs have been performed for each algorithm and each network, and the global run time has been divided by 50 for each, in order to obtain more precise statistics (notice that, although the algorithms are deterministic, the measurement of the run time values can have small fluctuations in some cases, so this is a most robust procedure).

The results obtained are shown in Table 1, where the run times are expressed in milliseconds. As expected, both Fast Pathfinder variants cleary outperform the original Pathfinder algorithm in terms of run time, being around 450 times quicker, and what is more important, they are significantly quicker than the Binary Pathfinder, reducing its run time in the order of around 23 times for the predecessor-based variant and around 27 times for the other. In this way, it can be seen how we were right in the assumption that the Fast Pathfinder variant not making use of the predecessor matrix is slightly faster (approximately a 10%) than the other version not requiring the computation of such data structure.

The most important conclusion we can draw from this experimental study is that, using our new Fast Pathfinder proposal, we are able to generate real scientograms of very large scientific domains in around 166 ms in the worst case (Argentina (2005) domain), while the current state-of-the-art approach, Binary Pathfinder, required more than 2 s in the best case (Cuba (2004)) and more than 5.5 s in the worst one (Mexico (2005)). This constitutes a great step ahead since this time reduction allows us to properly combine this pruning algorithm with the Kamada–Kawai layout technique, thus being able to generate these kinds of scientograms in real time.

Our second experiment concerns the comparison of the visual science maps obtained by the fastest variant of Fast Pathfinder (without using the predecessor matrix) with those obtained by the Binary Pathfinder algorithm. This is to empirically prove that the two algorithms give exactly the same results. To do so, we have written a bash script able to generate 1’000’000 random matrices, from size 3 to 500, containing integral or real numbers (two options selected randomly), and used as the cocitation matrices of virtual social networks. The goal was to compare edge by edge the results provided by the two considered algorithms. Only symmetric matrices were considered in this experiment and the parameters were set to $q = n - 1$ and $r = \infty$. In conclusion, during this long empirical experiment, no networks were found where a single edge differs from the Binary and the Fast Pathfinder algorithm’s results. For instance, the comparison of the results obtained by the two algorithms on a 500-nodes map is shown in Fig. 8.

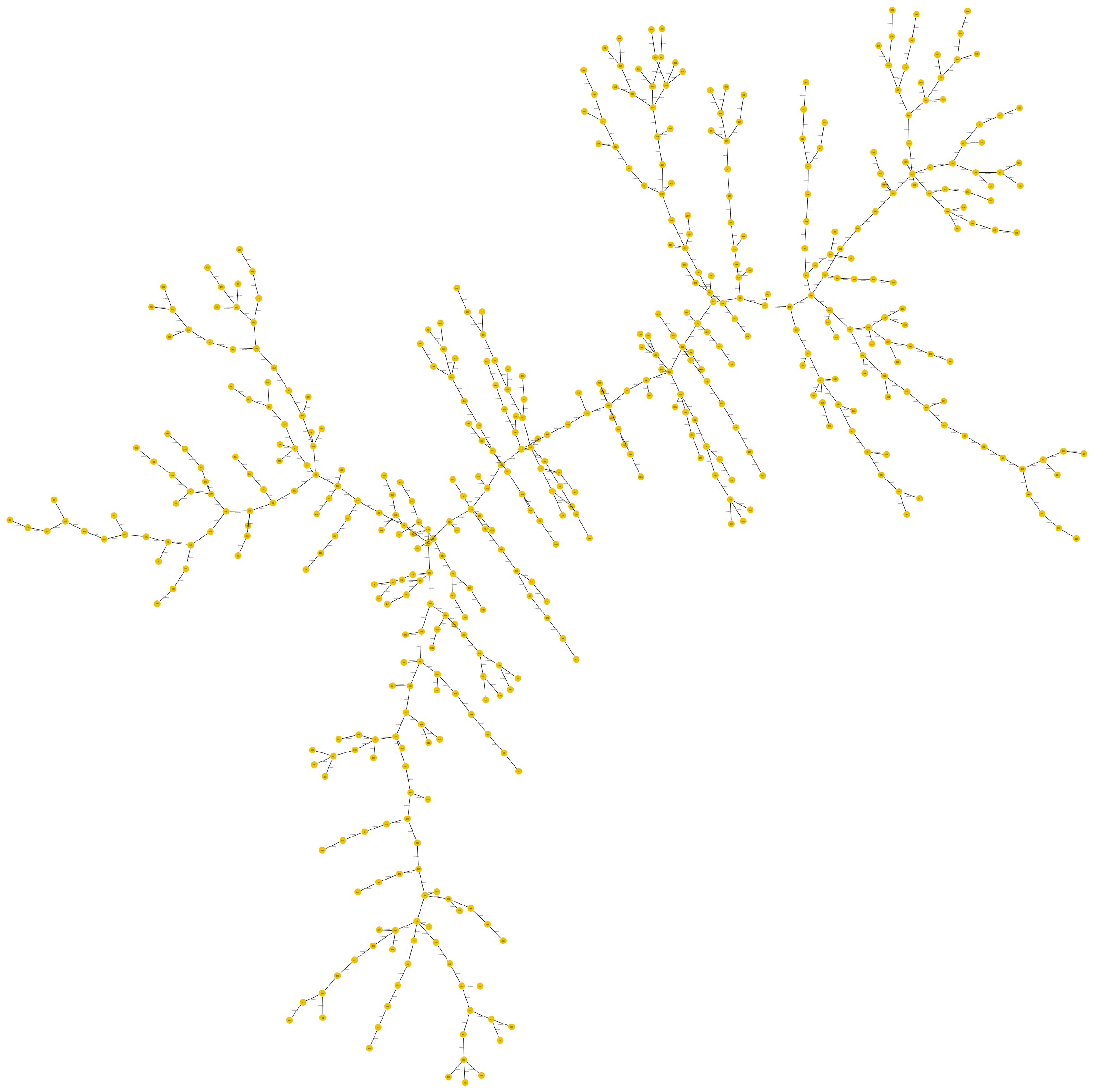

Fig. 8. Comparison of the results obtained by the Binary Pathfinder algorithm (left) and the Fast Pathfinder algorithm (right) on a 500-nodes map generated randomly.

5. Concluding remarks

In this paper, we have presented a new variant of the Pathfinder algorithm, to be used as a network pruning algorithm for the generation of visual representations of very large scientific domains, aiming to decrease its actual run time. Taking the classical Floyd–Warshall's graph shortest path algorithm as a base, we have been able to reduce the original Pathfinder time complexity in one order of magnitude, from $O(n^4)$ to $O(n^3)$, thus also clearly outperforming the state-of-the-art variant in terms of run time (Binary Pathfinder, $O(\log n \cdot n^3)$). The new algorithm has also a much simpler structure than the Binary Pathfinder, while maintaining the original Pathfinder's r parameterization (q must be fixed to $n - 1$ in the current application) and saving a great amount of memory.

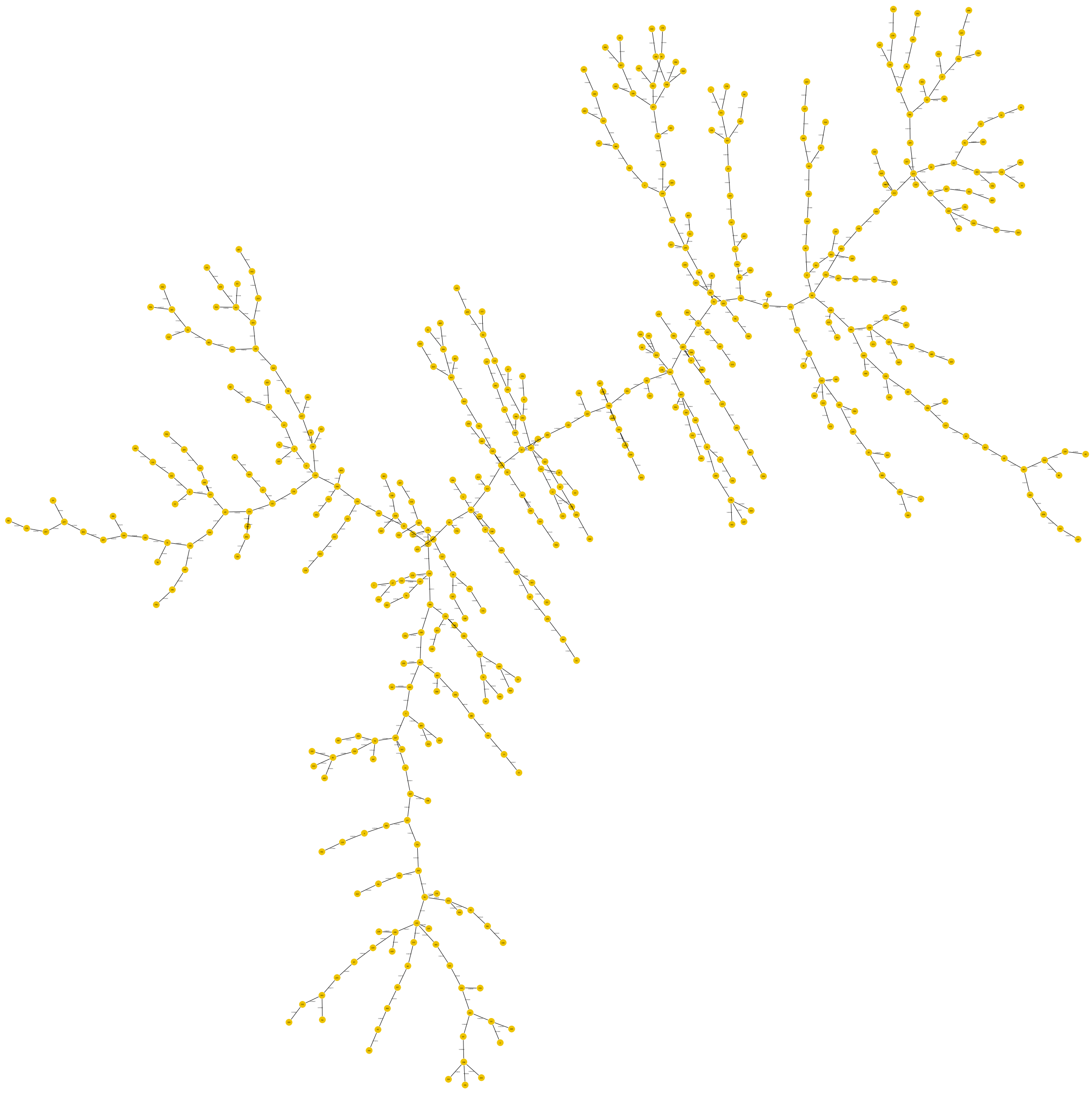

Fig. 8 (*continued*)

The experimental comparison developed using 20 large networks from real-world domains has demonstrated the capability of the new proposal to generate scientograms of very large scientific domains in real time.

Acknowledgements

We would like to thank Dr. Vicente Guerrero-Bote, from the University of Extremadura, Spain, for providing us with his Binary Pathfinder's code. We also would like to thank the anonymous reviewers for their interesting comments and suggestions, which has allowed us to improve the quality of the contribution.

References

Bellman, R., & Kalaba, R. (1965). *Dynamic programming and modern control theory*. New York, USA: Academic Press.

Borner, K., Chen, C., & Boyack, K. (2003). Visualizing knowledge domains. *Annual Review of Information Science and Technology, 37*, 179–255.

Boyack, K., Klavans, R., & Borner, K. (2005). Mapping the backbone of science. *Scientometrics, 64*, 351–374.

Buzydlowski, J. (2002). A comparison of self-organizing maps and pathfinder networks for the mapping of co-cited authors. PhD thesis, Drexel University.

Chen, C. (1998a). Bridging the gap: The use of pathfinder networks in visual navigation. *Journal of Visual Languages and Computing, 9*, 267–286.

Chen, C. (1998b). Generalised similarity analysis and pathfinder network scaling. *Interacting with Computers, 10*, 107–128.

Chen, C. (1999). Information visualization and virtual environments. Berlin, Germany: Springer.

Chen, C. (2004). Information visualization: Beyond the horizon. Berlin, Germany: Springer.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). Introduction to algorithms (2nd ed.). The MIT Press.

Dearholt, D., & Schvaneveldt, R. (1990). Properties of pathfinder networks. In R. Schvaneveldt (Ed.), *Pathfinder associative networks: Studies in knowledge organization* (pp. 1–30). Ablex Publishing Corporation.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*, 269–271.

Dreyfus, S. (1965). Dynamic programming and the calculus of variations. New York, USA: Academic Press.

Floyd, R. (1962). Algorithm 97: Shortest path. *Communications of the ACM, 5*(6), 345.

Guerrero-Bote, V., Zapico-Alonso, F., Espinosa-Calvo, M., Gómez-Crisóstomo, R., & Moya-Anegón, F. (2006). Binary pathfinder: An improvement to the pathfinder algorithm. *Information Processing and Management, 42*, 1484–1490.

Hjorland, B., & Albrechtsen, H. (1995). Toward a new horizon in information science: Domain analysis. *Journal of the American Society for Information Science, 46*(6), 400–425.

Kamada, T., & Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information Processing Letters, 31*(1), 7–15.

Leydesdorff, L. (2004a). Clusters and maps of science journals based on bi-connected graphs in the journal citation reports. *Journal of Documentation, 60*, 371–427.

Leydesdorff, L. (2004b). Top-down decomposition of the journal citation report of the social science citation index: Graph and factor-analytical approaches. *Scientometrics, 60*, 159–180.

Lin, X., White, H. D., & Buzydlowski, J. (2003). Real-time author co-citation mapping for online searching. *Information Processing and Management, 39*(5), 689–706.

Moya-Anegón, F., Vargas-Quesada, B., Herrero-Solana, V., Chinchilla-Rodríguez, Z., Corera-Álvarez, E., & Muñoz-Fernández, F. (2004). A new technique for building maps of large scientific domains based on the cocitation of classes and categories. *Scientometrics, 61*(1), 129–145.

Moya-Anegón, F., Vargas-Quesada, B., Chinchilla-Rodríguez, Z., Corera-Álvarez, E., Herrero-Solana, V., & Muñoz-Fernández, F. (2005). Domain analysis and information retrieval through the construction of heliocentric maps based on ISI–JCR category cocitation. *Information Processing and Management, 41*, 1520–1533.

Moya-Anegón, F., Vargas-Quesada, B., Chinchilla-Rodríguez, Z., Corera-Álvarez, E., González-Molina, A., Muñoz-Fernández, F., et al. (2006). Visualización y análisis de la estructura científica española: ISI web of science 1990-2005 (in Spanish). *El Profesional de la Información, 15*(4), 258–269.

Samoylenko, I., Chao, T.-C., Liu, W.-C., & Chen, C.-M. (2006). Visualizing the scientific world and its evolution. *Journal of the American Society for Information Science and Technology, 57*, 1461–1469.

Schvaneveldt, R. (1990). Pathfinder associative networks. Ablex Publishing Corporation.

Vargas-Quesada, B., & Moya-Anegón, F. (2007). Visualizing the structure of science. Springer.

Warshall, S. (1962). A theorem on boolean matrices. *Journal of the ACM, 9*(1), 11–12.

Watts, D. (2004). *Small Worlds. The Dynamics of Networks Between Order and Randomness*. Princeton University Press.

Watts, D., & Strogatz, S. (1998). Collective dynamics of 'small-world' networks. *Nature, 393*, 440–442.

White, H. (2003). Pathfinder networks and author cocitation analysis: A remapping of paradigmatic information scientists. *Journal of the American Society for Information Science and Technology, 54*(5), 423–434.