A Quick MST-Based Algorithm to Obtain Pathfinder Networks (∞, $n-1$)

Arnaud Quirin and Oscar Cordón
European Centre for Soft Computing, Edf. Científico Tecnológico, Mieres, Spain.
E-mail: {arnaud.quirin, oscar.cordon}@softcomputing.es

Vicente P. Guerrero-Bote
Department of Information and Communication, University of Extremadura, Badajoz, Spain.
E-mail: guerrero@unex.es

Benjamín Vargas-Quesada and Felix Moya-Anegón
SCImago Group, Communication and Information Science Faculty, University of Granada, Granada, Spain.
E-mail: {benjamin, felix}@ugr.es

Network scaling algorithms such as the *Pathfinder* algorithm are used to prune many different kinds of networks, including citation networks, random networks, and social networks. However, this algorithm suffers from run time problems for large networks and online processing due to its $O(n^4)$ time complexity. In this article, we introduce a new alternative, the *MST-Pathfinder* algorithm, which will allow us to prune the original network to get its PFNET(∞, $n-1$) in just $O(n^2 \cdot \log n)$ time. The underlying idea comes from the fact that the union (superposition) of all the Minimum Spanning Trees extracted from a given network is equivalent to the PFNET resulting from the Pathfinder algorithm parameterized by a specific set of values ($r = \infty$ and $q = n-1$), those usually considered in many different applications. Although this property is well-known in the literature, it seems that no algorithm based on it has been proposed, up to now, to decrease the high computational cost of the original Pathfinder algorithm. We also present a mathematical proof of the correctness of this new alternative and test its good efficiency in two different case studies: one dedicated to the post-processing of large random graphs, and the other one to a real world case in which medium networks obtained by a cocitation analysis of the scientific domains in different countries are pruned.

Introduction

Network models are used in many areas of cognitive and computer science. Among them, social network models depict the complex tissue of relationships between individuals. The most important key of social network analysis is that it does not focus on individual or other local properties, but on the relationship between individuals, groups, or other kind of social actors and at various scales, from personal to international. For that, the field of social network analysis provides various metrics about individuals (or groups, societies, organizations, Web sites, etc. denoted by the term *node* in the following) and relationships (denoted by the term *link* in the following) in order to process, map, and visualize these entities. These metrics, such as the betweenness and the closeness for instance, are exploited in many scientific domains such as sociology, sociolinguistics, social psychology, communication studies, economics, and information science (Breiger, 2004; Martino & Spoto, 2006).

Social networks have some interesting and unusual topological properties that are often valuable to be printed graphically. However, the raw networks cannot be often visualized easily, especially when the sizes of the networks grow proportionally with the number of data to be dealt with, and thus specific algorithms for simplifying such large graphs have been developed. Network scaling algorithms, whose goal is to take proximity data and to obtain structures revealing the underlying organization of those data, use similarities, correlations, or distances to prune a graph based on the proximity between a pair of nodes. One of the most known, the Pathfinder algorithm (Dearholt & Schvaneveldt, 1990), is used frequently due to its various mathematical properties, including the conservation of the triangle inequalities among a path of any number of links, the capability of modeling asymmetrical relationships, the representation of the most *salient* relationships present in the data, and the fact that hierarchical constraints in most cluster analysis techniques do not apply to Pathfinder Networks (PFNETs; Dearholt & Schvaneveldt, 1990).

Received September 12, 2007; revised April 29, 2008; accepted April 29, 2008

 DOI: 10.1002/asi.20904

Pathfinder uses two parameters: r, which defines the Minkowski metric used to measure distances in a path, and q, which is a limit on the number of links allowed to violate the triangle inequality. The PFNET structure becomes sparser (has fewer links) as either r or q increases (Dearholt & Schvaneveldt, 1990), whereas its interpretability increases. This explains why PFNETs $(\infty, n-1)$, where both parameters were set to $q = n-1$ and $r = \infty$, are used in a large variety of applications, including author cocitation analysis (Buzydlowski, 2002), latent knowledge visualization (Chen, Kuljis, & Paul, 2001), scientific domain visualization (Chen, 1998a, 1998b, 2004; Moya-Anegón et al., 2007; Vargas-Quesada & Moya-Anegón, 2007), communication networks (Shope, DeJoode, Cooke, & Pedersen, 2004), animated visualization models of toxins (Chen & Morris, 2003), and mental models (Kudikyala & Vaughn, 2005).

Although this method is very powerful and has been successfully applied to many different tasks, it has a main drawback: its large run time consumption, which makes it difficult to apply it for the processing of large networks when time is a practical concern, as a consequence of its high order polynomial time complexity ($O(n^4)$). This slowness disallows the online pruning of networks and cannot provide a real time response when a strong interaction with a user is needed.

A couple of proposals have been made to solve this problem. Guerrero-Bote et al. (2006) proposed the *Binary Pathfinder* algorithm, which reduced both the time and the space complexity of the original Pathfinder to $O(n^3 \cdot \log n)$ and *4* squared matrices, respectively (Pathfinder requires the use of $2 \cdot (n-1)$ squared matrices). In Quirin, Cordón, Santamaría, Vargas-Quesada, & Moya-Anegón (2007), we considered the similarity of the latter with shortest path algorithms to design an even quicker variant by fixing the value of parameter q to $n-1$, reducing its run time to $O(n^3)$ and its space complexity to 2 squared matrices. In spite of its speed, this *Fast Pathfinder* variant is still too slow for the real time pruning of large networks, as the run time was only accelerated by a factor of n.

In this article, we introduce a new alternative, *MST-Pathfinder*, that drastically decreases the run time by a factor of $n^2 \cdot \log^{-1}(n)$ (i.e., from $O(n^4)$ to $O(n^2 \cdot \log n)$) and saves an amount of memory corresponding to $2 \cdot n - 5$ squared matrices, compared to the original Pathfinder. MST-Pathfinder space complexity is just a little bit larger than Fast Pathfinder and smaller than Binary Pathfinder. To obtain this result, we had to fix r and q, the two main parameters of Pathfinder, to some specific values, respectively ∞ and $n-1$. So, even if this algorithm cannot be used in the general case, this specific parameter setting is the most extended one for PFNET applications.

Two case studies are considered to test the efficiency of the new proposal. In the first one, we study large and fully connected random matrices and compare the run time of all the algorithms applied on them. In a second case study, we have selected the generation of visual science maps (or scientograms) for the representation of vast scientific domains, based on cocitation information. The efficiency of our new algorithm is critical for this real case in which the saving in time and memory is important for online visualization.

The structure of the current contribution is as follows. In the second section, we review the existing improvements of Pathfinder, designed for reducing its time and space complexity. In the third section, we present our new proposal, MST-Pathfinder. Last, in the fourth section, we experimentally check the validity of our new MST-Pathfinder proposal in comparison with the previous Pathfinder variants with respect to run time in the two case studies. Finally, some concluding remarks are pointed out in the last section.

Previous Approaches to Speed up the Pathfinder Run Time

In this section, we first review the basis of the original Pathfinder algorithm. Then, we describe the current state-of-the-art of the previous existing proposals to design new, quicker versions of it following the aim of speeding up its processing.

Pathfinder

Pathfinder was introduced by Dearholt & Schvaneveldt (1990) as a technique to choose the shortest links in a network in the field of social networks analysis. The result of the Pathfinder procedure is a pruned network called PFNET, which keeps only those links that do not violate the triangle inequality, stating that the direct distance between two nodes must be less 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 the following two main parameters.

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

$$D = \left(\sum_i d_i^r \right)^{1/r}$$

   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.

DOI: 10.1002/asi

1. Compute $W^{i+1} = W \Theta W^i$, as follows: $w_{jk}^{i+1} = \text{MIN}((w_{jm})^r + (w_{mk}^i)^r)^{1/r}$, for $1 \leq m \leq n$.
2. Compute D^i, as follows: $d_{jk}^i = \text{MIN}(w_{jk}^1, \ldots, w_{jk}^i)$, 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 Pathfinder algorithm.

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

- W_{jk}^i, 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_{jk}^{i-1}, with W^1 being the original weight matrix.
- D_{jk}^i, 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 $W_{jk}^1, \ldots, W_{jk}^i$.

The original Pathfinder algorithm pseudo-code is shown in Figure 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 $2 \cdot q \cdot n^2$ ($2 \cdot n^3 - 2 \cdot n^2$ when $q = n-1$), since there is a need to build q matrices W^i and q other matrices D^i, as seen above.

Binary Pathfinder

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 two following 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 was done for the consecutive W^i matrices: $D^{i+j} = D^i \Theta D^j$.

Hence, we 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:

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

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 that 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 indexes being powers of 2: D^1, D^2, D^4, D^8, ……

The Binary Pathfinder algorithm pseudo-code is shown in Figure 2. Notice that Binary Pathfinder keeps the same algorithmic approach than the original Pathfinder version, the classical *dynamic programming* approach (Dreyfus, 1965), 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 (Bellman & Kallaba, 1965).

The principal 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 $O(n^3 \cdot \log q)$ instead of $O(n^3 \cdot q)$, which in the maximum case becomes $O(n^3 \cdot \log n)$ instead of $O(n^4)$, a very significant time difference for medium and large networks. 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. Moreover, as Pathfinder, this algorithm can be applied on directed networks and can consider any possible value for q and r.

Fast Pathfinder

When analyzing the operation mode of the original Pathfinder algorithm with $q = n-1$ 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. In this case, the triangle inequality is

1. $i = 1$; $nq = 0$; Generate $D^1 = W$; $D^q \leftarrow \infty$.
2. IF (q mod $2 = 1$) THEN compute $D^q = D^q \Theta D^1$.
3. $nq = 1$.
4. WHILE ($2 \cdot i \leq q$)
5. Compute $D^{2 \cdot i} = D^i \Theta D^i$.
6. IF (($q - nq$) mod $(4 \cdot i) > 0$) THEN
7. Compute $D^q = D^q \Theta \mathrm{D}^{2 \cdot i}$.
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. Binary Pathfinder algorithm.

DOI: 10.1002/asi

```
1. D ← W; PFNET ← ∅.
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 = MIN{d_ij, ((d_ik)^r + (d_kj)^r)^1/r}.
6. FOR i from 1 to n DO
7.    FOR j from 1 to n DO
8.       IF (d_ij = w_ij) THEN PFNET ← PFNET ∪ (i, j).
```

FIG. 3. The Fast Pathfinder algorithm.

verified for the best path between any couple of nodes in the network, thus the problem becomes a shortest path one.

In Quirin et al. (2007), the authors took the latter idea as a base to compute the distance matrix in a more direct way, thus reducing the number of steps required and speeding up the algorithm. To do so, we applied again the dynamic programming approach in order to ensure the obtaining of the optimal solution for the graph shortest path problem.

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. Since D^{n-1} is a shortest path distance matrix, we borrowed an alternative and quicker way to compute it from a classical algorithm in graph theory (Cormen, Leiserson, Rivest, & Stein, 2001): Floyd-Warshall's algorithm (Floyd, 1962; Warshall, 1962), also based on the dynamic programming approach, which is able to compute all the shortest paths of length up to $n-1$ links (according to an Euclidean metric) in a cubic time complexity.

Hence, the Floyd-Warshall's algorithm was adapted to the computation of the D^{n-1} matrix for a PFNET using the Minkowski r-metric and became the base of the *Fast Pathfinder* proposal. Working in this way, we are able to build this matrix in cubic time and avoid the need to compute the temporary matrices W^i and D^i, the substitution is much more effective. The Fast Pathfinder pseudo-code is shown in Figure 3.

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 $O(n^3) + O(n^2) = \text{MAX}\{O(n^3), O(n^2)\} = O(n^3)$. Besides, notice that the algorithm is required to store only two square matrices to operate (W and D).

MST-Pathfinder

The goal of this section is to introduce the MST-Pathfinder algorithm using a well-known relation between *Minimum Spanning Trees* (MST) and PFNETs parameterized with ($r=\infty$ and $q=n-1$). We first review the Kruskal's MST algorithm that has served as a base for the definition of the new algorithm, which is presented thereafter. Then, we present the mathematical proof of the correctness of MST-Pathfinder, and we detail its time and its space complexity.

Underlying Idea: Relation Between Minimum Spanning Trees and Pathfinder Networks ($r=\infty$, $q=n-1$)

It is well-known that there is a relationship between the results obtained with a MST[1] algorithm and the Pathfinder algorithm. Dearholt and Schvaneveldt (1990) explicitly stated that, for a given symmetric cost matrix W, r and q, the union of all the MSTs extracted from a PFNET(r, q) is PFNET(∞, $n-1$). As we have seen in the previous two sections, for each couple of nodes (i, j), PFNET(∞, $n-1$) is the set of links with the minimum cost among all the paths between the nodes i and j. On the other hand, Chen and Morris (2003) explored the relationships between both from a network visualization point of view, after introducing that a PFNET is the union set of all the possible MSTs derived from a network. The authors compared the Kamada-Kawai visualization of an MST network with the one obtained from a PFNET, using the parameters $q=n-1$ and $r=\infty$. They concluded by saying that the visual-spatial features of the PFNET are better than those of the MST applied on cocitation networks (for instance, MST gives more clustered networks), but the MST algorithm is more efficient.

So, the very challenging development we are tackling in this contribution is to provide the user with a new algorithm giving the same result than Pathfinder (i.e., the same PFNET(∞, $n-1$)) but with the same efficiency than the usual MST algorithms. In fact, our roadmap is to propose new implementations of PFNET-based algorithms, by imposing some constraints on the parameter values compared to the original Pathfinder, in such a way that the new variants can be faster for the generation of PFNETs(∞, $n-1$) online. As we have seen in the previous section, Binary Pathfinder is able to speed up the original Pathfinder algorithm, while Fast Pathfinder algorithm speeds up Binary Pathfinder. The latter one does so by imposing a single constraint, i.e., considering $n-1$ as the only possible value for the parameter q. We can notice that all the latter algorithms apply the *dynamic programming* approach, well-known in algorithm theory (Dreyfus, 1965).

Now, we have explored how imposing one additional constraint we can succeed in generating PFNETs using a *greedy* approach. Greedy algorithms are known to be faster than their dynamic programming counterparts when they are able to reach the globally optimal solution (Cormen et al., 2001). Therefore, our idea was to explore the potential of MST algorithms for this task, first because a connection with Pathfinder has already been proved, and second because they use the greedy approach so they should be faster than the current state-of-the-art Pathfinder variants.

To do so, we need to look for a stronger relation between the PFNET(∞, $n-1$) of a network G, and the different MSTs of G. Although it is not stated explicitly in Dearholt and

[1] Let $G=(V, E)$ be a non-directed weighted and connected network where V is the set of the nodes and E is the set of the links valued by their costs. A Minimum Spanning Tree of G is a sub-network $T=(V, E')$ of G, $E' \subset$ E, including all the nodes of G, where T is a tree and where the sum of the costs of each link is minimal.

DOI: 10.1002/asi

Schvaneveldt (1990), the union of all the MSTs extracted from a given network G is also its PFNET(∞, $n-1$).

This result is very important because even if it concerns only a specific setting of the Pathfinder algorithm ($q = n-1$ and $r = \infty$), this setting is the most used for many applications. In fact, when applied to a given network, Pathfinder(∞, $n-1$) prunes the network in such a way that two nodes only remain connected if the weight of their link is equal or better than the minimal link weight (the maximal similarity) on all the other paths. The consequences on the global network are that (1) the nodes are connected only if they are the most closest, in terms of the distance measure, and (2) the shortest path between two nodes in a PFNET is intuitively the best one to describe the relationship of these two nodes.

Hence, as the generation of PFNET(∞, $n-1$) is the simple union of all the MSTs of a given network, we can design a new algorithm, alternative to Pathfinder, to generate these PFNETs by using any existing MST algorithm. In fact, there are at least two of such algorithms that are well-known in graph theory and computer science literature, respectively called *Kruskal's* (Kruskal, 1956) and *Prim's* (Prim, 1957). Because of its simplicity, the adaptation of *Kruskal's* algorithm was finally used in the design of our new pruning technique; it was called MST-Pathfinder, in reference to the new approach to quickly prune networks based on MSTs, even if it does not follow the usual Pathfinder algorithm operation mode as the different variants introduced in the previous section.

Kruskal's MST Algorithm

Kruskal's algorithm (1956) is a greedy algorithm that gives an MST for a connected weighted network. In it, all the links are sorted in ascending order. The links are then added one by one to the final tree only if they are not already in the same cluster (i.e., the tree remains a tree and does not become a network). In this case, at each iteration, two nodes belonging to the current tree are only connected by the links having the minimal cost, so the total cost of the tree is also minimal.

The algorithm uses several sub-functions. The function CREATE-CLUSTER(v) applied to a node v creates a single cluster of size 1, just including v as member. The function CLUSTER(v) returns the cluster associated to node v. As the only purpose of this function is the comparison of two clusters to know if they are the same or not, this is usually done by returning an element (or an index) able to identify in an unique way the cluster containing v. The function MERGE-CLUSTER(u, v) performs the union of the cluster containing node u and that containing node v. *Kruskal's* algorithm pseudo-code is shown in Figure 4.

As stated before, this algorithm is quite simple. When a disjoint-set data structure is used to perform the operations on the clusters, its time complexity is $O(|E| \cdot \log(n))$, where $|E|$ is the number of links and n the number of nodes of G (Cormen et al., 2001). As $|E|$ is bounded by n^2, the time complexity is equivalent to $O(n^2 \cdot \log(n))$. As we need to store two link sets (T and F), the memory complexity is $2 \cdot n^2$.

```
1. Define a tree T = ∅.
2. Define V[G], the set of the nodes of the network G.
3. For each node v ∈ V[G]
4.    CREATE-CLUSTER(v).
5. Create F, a set of all the links of G sorted by their weights.
6. FOR each link e(u, v) ∈ F
7.    IF CLUSTER(u) ≠ CLUSTER(v), THEN
8.       T = T ∪ {e(u, v)}.
9.       MERGE-CLUSTER(u, v).
10. Return T.
```

FIG. 4. Kruskal's algorithm.

Structure of MST-Pathfinder

In view of the latter, to generate the PFNET(∞, $n-1$), we have to compute all the possible MSTs of a network and return the union of the corresponding link sets. As said, the two corresponding well-known algorithms to compute an MST are *Kruskal's* and *Prim's*. We could justify the choice of *Kruskal's* algorithm instead of *Prim's* one by noticing that the latter grows an initial tree by looking all the neighbours of a given node and selecting the one that minimizes the current cost of the tree. This neighbour-based behavior seems to be more efficient when the networks are represented as a real graph (by means of pointers, or some other data structures). Another more technical and important reason for this choice will be explained later.

Our proposal concerns a way to compute the union of all the possible MSTs of a given network with the same time complexity required to compute just one of these MSTs. First, we should notice that the differences between the various MSTs of a given network are only related to the links having the same values, but not being present in the same cluster during a given step of the algorithm. During this step, the algorithm has to choose between different links, and all these choices correspond to the same amount of different possible MSTs. In particular, if all the weights of the original network links are different, the MST is unique (Kravitz, 2007). This is related to the non-deterministic behavior of the original *Kruskal's* algorithm, which lets this algorithm produce, arbitrarily, only one of all the possible MSTs. In the following, we will refer to the links that are not shared by *all* the possible MSTs generated from a given network as *special-links*.

So, to achieve our final goal of merging directly the different MSTs during the run of the algorithm, we have to detect these special-links. The first property of these links is that they have the same weights. This fact can be efficiently checked once the link set is sorted (during the initialization of the algorithm). If the original algorithm is scrutinized, we can notice that once one of these links is added to the tree T, the two clusters corresponding to this link are immediately merged. In this case, it is impossible to detect if the two links are or not special-links in a further step.

The only solution is to store them in a temporary set H instead of adding them directly to the current tree, and process this set only when we are sure that this will not affect the

DOI: 10.1002/asi

1. Define a tree $T = \emptyset$.
2. Define $V[G]$, the set of the nodes of the network G.
3. Define W, the matrix of the costs for each link of G.
4. For each node $v \in V[G]$
5. CREATE-CLUSTER(v).
6. Create F, a set of all the links of G sorted by their weights.
7. FOR each link $e(u, v)$ remaining in F
8. $H = \emptyset$.
9. FOR each link $e(u', v')$ remaining in F where $w(u, v) = w(u', v')$
10. $F = F - \{e(u', v')\}$.
11. IF CLUSTER(u') $\neq$ CLUSTER(v'), THEN
12. $T = T \cup \{e(u', v')\}$.
13. $H = H \cup \{e(u', v')\}$.
14. FOR each link $e(u', v') \in H$
15. MERGE-CLUSTER(u', v').
16. Return T.

FIG. 5. The MST-Pathfinder algorithm.

detection of other special-links. The processing of the set (i.e., the union of all the links of the temporary set with the current tree) can be done once a new link with a different weight has been found. These remarks let us define directly the MST-Pathfinder algorithm, shown in Figure 5.

The main goal of this algorithm is to merge directly all the links corresponding to the different possible MSTs of a network G, so the final result is equivalent to the PFNET $(\infty, n-1)$ of that network. Notice that, on the contrary to the Pathfinder variants described in the previous section, there is not a need of any weight comparison to select the final links belonging to the PFNET but those are just the same ones in the MST. Thus, the algorithm is faster working in this way.

It is also worth noticing that this improvement can be done only with *Kruskal's* algorithm and not with *Prim's*. Indeed, in *Kruskal's*, the links-sort acts in a global way, allowing us to detect (and join) the different links that would be present in the different MST trees. In *Prim's* algorithm, the growing of the tree is done in an incremental (so a local) way, by adding a non-explored link to the current tree. Under that condition, the detection of two links with the same values that should be merged could not be achieved directly, at least using an efficient computation (a sorting operation would be required to do so because they could be located far away from each other).

Proof of the Correctness of MST-Pathfinder

Let G be a connected, weighted graph and let T be the subgraph of G produced by the MST-pathfinder algorithm (i.e., the union of all the MSTs of the graph G), and PF be the subgraph PFNET$(\infty, n-1)$.

If $T = PF$, then T is the PFNET$(\infty, n-1)$. Otherwise, there should be links in T that are not in PF or the contrary.

Let $e(u, v)$ be a link with weight w that is in T, but is not in PF. This means that there is a path M in PF connecting the vertex u and v with a lower cost than w. But this could not be possible because all the links composing M should have been first considered by the MST-pathfinder algorithm during its run. So, when $e(u, v)$ was considered, the vertexes u and v would have been in the same cluster. As a consequence, $e(u, v)$ should not be in T.

On the other hand, let $e(u, v)$ be an edge with weight w that is in PF, but is not in T. Because of the definition of PFNET, this means that all the other paths in T connecting u and v contain at least one link $e(i, j)$ with a weight larger than w. In that case, the MST-pathfinder algorithm could not have avoided the inclusion of $e(u, v)$ in T, because it would have been analyzed before or at the same time than that link $e(i, j)$, and u and v would have not been at this time in the same cluster. As a consequence, $e(i, j)$ should also be in T.

By contradiction, we have $T = PF$.

Time and Space Complexity of MST-Pathfinder

The algorithm needs $O(|E| \cdot \log(|E|))$ operations to sort the list of the links by their weights, where $|E|$ is the number of links. To know which cluster belongs to each node, we can use a disjoint-set data structure with union by rank and path compression. According to Cormen et al. (2001), the cost of the related operations depends on two parameters: M, the total number of CREATE-CLUSTER(v), CLUSTER(v) and MERGE-CLUSTER(u, v) calls and N, the number of CREATE-CLUSTER(v) calls. In this case, the best total cost of the calls is proved to be in $O(M \log(N))$. In MST-Pathfinder, CREATE-CLUSTER(v) is called n times. Now, although there are two nested FOR loops, at most $|E|$ CLUSTER(v) and $|E|$ MERGE-CLUSTER(u, v) operations are performed in the worst case, because the size of F decreases by 1 at each step, and both loops are based on F. In our case, $M = 3|E| + n$ and $N = n$, so the cost of the algorithm is expressed by:

$$\begin{aligned} &O(|E| \cdot \log(|E|)) + O((3|E| + n) \cdot \log(n)) \\ &\quad = \text{MAX}\{O(|E| \cdot \log(|E|)), O(|E| \cdot \log(n)), O(n \cdot \log(n))\} \\ &\quad = O(|E| \cdot \log(|E|)) \end{aligned}$$

When having a dense network, $|E|$ is close to n^2 and $\log(n^2)$ is $O(\log(n))$, so the theoretical time complexity of the full algorithm can be simplified to $O(n^2 \cdot \log(n))$, having the same time complexity that *Kruskal's* algorithm. In conclusion, this algorithm is much faster than the original Pathfinder ($O(n^4)$, when applied with $q = n - 1$), in spite of the recent improvements we have described in the previous section: the Binary Pathfinder has a time complexity of $O(n^3 \cdot \log(n))$ and Fast Pathfinder has a time complexity of $O(n^3)$.

It is important to mention that the time complexity developed previously has only a theoretical importance. In practice, due to the high number of calls to the CLUSTER(v) functions, compared to the number of calls to MERGE-CLUSTER(u, v), it is more efficient to consider another, much simpler data structure than the disjoint-set to implement these functions. In our case, each node v is affected by a single index $c(v)$ encoding the corresponding cluster in an

DOI: 10.1002/asi

TABLE 1. Comparison of the run time (expressed in seconds) of all the algorithms for the random matrices case studies.

#	#Nodes	#Links	Original PF	Binary PF	Fast-PF	MST-PF (low-complexity)	MST-PF (practical)
1	100	9.90E+03	1.55	0.183	0.00925	0.0021	0.00208
2	200	3.98E+04	23.59	1.76	0.0702	0.0101	0.0101
3	300	8.97E+04	181	8.98	0.238	0.0266	0.0264
4	400	1.60E+05	604	24.56	0.585	0.0537	0.0533
5	1000	9.99E+05	>3600	>3600	10.01	0.629	0.629
6	10000	1.00E+08	>3600	>3600	>3600	128.31	127.62

unique way, so each call to CLUSTER(v) is done in $O(1)$. Then, MERGE-CLUSTER(u, v) involves affecting the value $c(u)$ to each node having the value $c(v)$, and this can be done in $O(n)$. So, the practical time complexity of the algorithm is $O(n) + O(|E| \cdot \log(|E|)) + O(|E| \cdot n) = \text{MAX}\{O(n), O(|E| \log(|E|)), O(|E| \cdot n)\} = O(|E| \cdot n) = O(n^3)$. Compared to $O(n^2 \cdot \log(n))$, this complexity is worst in theory. However, the run time is faster in practice, as we will see in the next section. For this purpose, we name the algorithm using the disjoint-set data structure with union by rank and path compression, the *low-complexity* MST-Pathfinder; and the index-based disjoint-set variant, the *practical* MST-Pathfinder.

Concerning the space complexity, we need to store three lists of links with their weights: F, T, and H. The record of the cluster index can be done with a single additional attribute for each node, so the total space complexity of the algorithm is $3 \cdot n^2 + n$. The only drawback is that this algorithm cannot be applied on directed networks, because the sorting process deals with undirected links.

Experiments

The goal of this section is to present an experimental study showing the run time improvement of the MST-Pathfinder algorithm in comparison with the other Pathfinder variants when dealing with two case studies: a laboratory problem where networks of medium and large sizes are randomly generated, and a real world problem of information visualization with medium-sized networks obtained by a cocitation analysis of 20 scientific domains.

Case Study 1: Random Networks

In this experiment, our aim is twofold. First, we compare the execution time of the five algorithms (the original Pathfinder, the Binary Pathfinder, the Fast Pathfinder, the low-complexity MST-Pathfinder, and the practical MST-Pathfinder) on random matrices, from sizes 100 to 10'000. Second, we prove that the practical MST-Pathfinder is faster than the low-complexity MST-Pathfinder. Recall that the two variants use two different internal algorithms to manage the disjoint-set data structure. The former uses the most known algorithm in the literature for this structure, the union by rank and path compression (Cormen et al., 2001), while the latter uses a trivial algorithm based on the update of a simple index.

In order to measure the run time of all the algorithms, we used 20 different matrices for each of the six networks sizes considered (100, 200, 300, 400, 1000, 10'000), randomly filled with real numbers from 1 to 1000, and we averaged the results in order to make a more fair comparison (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 goal was to compare the run time of all the algorithms using laboratory cases and very large matrices. Only symmetric matrices were considered in this experiment; they represent fully connected networks and the parameters were set to $q = n - 1$ and $r = \infty$, when applicable. The algorithms have been written in C, and compiled on Linux with the GNU *GCC* compiler with the $-O3$ option on an Intel dual-core Pentium 3.2 GHz with 2 GB of memory.

The obtained results are shown in Table 1. The first conclusion is that this experiment shows the important run time improvement achieved by the MST-Pathfinder algorithm for medium[2] and large networks. Networks containing 100'000'000 links are pruned in around 2 minutes instead of time greater than 1 hour spent by the remaining algorithms.[3] This allows us to process medium networks online and large networks in a reasonable time. Another point is that the improvement of MST-Pathfinder, in comparison to Fast-Pathfinder, is clearly demonstrated here: From a complexity of $O(n^3)$ to a complexity of $O(n^2 \cdot \log(n))$, the run time changes from more than 1 hour (around 3 hours according to some tests we performed without considering the 1 hour run time threshold) to only 2 minutes.

The second conclusion is that the practical MST-Pathfinder algorithm is slightly faster than the low-complexity variant, as explained in the previous section: The improvement is around 0.54% for a network of 10'000 nodes.

Case Study 2: A Method to Generate Scientograms for Vast Scientific Domains

The graphical representation of information for its later visualization is a very common activity in the most of scientific disciplines. However, its combination with computer science is a rather new task.

[2]We use the term *medium* to describe networks having 100 to 1000 nodes (Börner et al., 2007).

[3]A run time of 1 hour was our imposed threshold in order to stop the algorithms in a reasonable time.

DOI: 10.1002/asi

The achievement of a vast scientogram is a recurrent idea in the modern age. In 1998, Chen (1998a, 1998b) was the first researcher to bring forth the use of PFNETs in citation analysis. This is due to the fact that scientograms are the most appropriate means to represent the spatial distribution of research areas, while also affording information on their interactions (Small & Garfield, 1985). Taking the latter as a base, Moya-Anegón et al. (2004) proposed a method for the visualization and analysis of vast scientific domains using the ISI[4]-JCR category cocitation information. They represented it as a social network, simplified that network by means of the Pathfinder algorithm considering $q = n - 1$ and $r = \infty$, and graphically depicted its layout using the Kamada-Kawai's algorithm (Kamada & Kawai, 1989), thus getting a structural model of the scientific research in a vast domain. Note that $q = n - 1$ and $r = \infty$ 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).

The different method stages are briefly described as follows.

Category Cocitation Measure

Cocitation is a widely used and generally accepted technique for obtaining relational information about documents belonging to a domain. Because we strive to represent and analyze the structure of vast domains, whether they be thematic, geographic, or institutional, we fall back on to ISI-JCR cocitation categories (Moya-Anegón et al., 2004) as a tool for this purpose.

Hence, once the rough information of the ISI-JCR cocitation for the categories present in the domain to be analyzed is obtained, a cocitation measure *CM* is computed for each pair of categories i and j as follows:

$$CM(ij) = Cc(ij) + \frac{Cc(ij)}{\sqrt{c(i) \cdot c(j)}}$$

where *Cc* is the cocitation frequency and c is the citation frequency.

Notice that the aim of this scientogram generation method is that the final scientogram obtained is a tree. Hence, in order to avoid the existence of cycles in the pruned network, the considered measure of association adds the normalized cocitation (divided by the square root of the product of the frequencies of the cocited documents' citations; Salton & Bergmark, 1979) to the rough category cocitation frequency. In this way, the network weights become real numbers, allowing us to create small differences between similar values for the cocitation frequency, thus avoiding the occurrence of cycles and achieving the optimal prune of each link considering the citing conditions of each category.

[4]Currently registered as Thomson Reuters.

Network Pruning by Pathfinder

The Pathfinder algorithm is then applied to the cocitation matrix to prune the network. We should take into account the fact that the networks resulting from citation, cocitation, or term co-occurrence analysis are usually dense when the categories are used as the unit for each node. Due to this fact, and especially in the case of vast scientific domains with a high number of entities (categories in our case) in the network, Pathfinder is usually parameterized to $r = \infty$ and $q = n - 1$, obtain an schematic representation of the most outstanding existing information by means of a network showing just the most salient links.

Network Layout by Kamada-Kawai

Kamada and Kawai's (1989) algorithm is then used to automatically produce representations of the pruned network resulting from the Pathfinder run on a plane, starting from a circular position of the nodes. It generates social networks with aesthetic criteria such as the maximum use of available space, the minimum number of crossed links, the forced separation of nodes, building balanced maps, etc.

In general, the weights of the links belongs to R and are all different, so the final result is a tree as the one shown in Figure 6.

Comparison of the Run Time

The goal of this section is to conduct the experimental procedure described above to measure the run time of all the Pathfinder variants considered in this article and show the run time improvement obtained on 20 real world networks. The networks were obtained from the ISI-JCR category cocitation information available at the SCImago research group's Atlas of Science.[5] Their sizes range from 212 to 263 nodes and from 8485 to 23430 links. Notice that the link weights in these medium networks correspond to similarities instead of distance measurements.[6] The original Pathfinder, the Binary, the Fast Pathfinder, and the MST-Pathfinder algorithms have been compared to prune the latter networks to design the scientograms. Pathfinder parameters have been set to $q = n - 1$ and $r = \infty$ (when considered), the typical values in vast domain scientogram design. The same computer considered for the previous case study has been used: an Intel dual-core Pentium 3.2 GHz with 2 GB of memory.

Fifty independent runs have been performed for each algorithm and each network, and the global run time has been averaged for each to obtain more precise statistics. The obtained results are shown in Table 2.

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

[6]According to the Moya's method (Moya-Anegón et al., 2004), the normalized cocitation coefficients are used and correspond to similarities. More details can be found in that paper. Actually, using similarities or distances has no influence at all in our proposal. In case of using similarities, we would need only to sort the set of the links of G in a reverse way in the MST-Pathfinder algorithm (see Figure 5).

DOI: 10.1002/asi

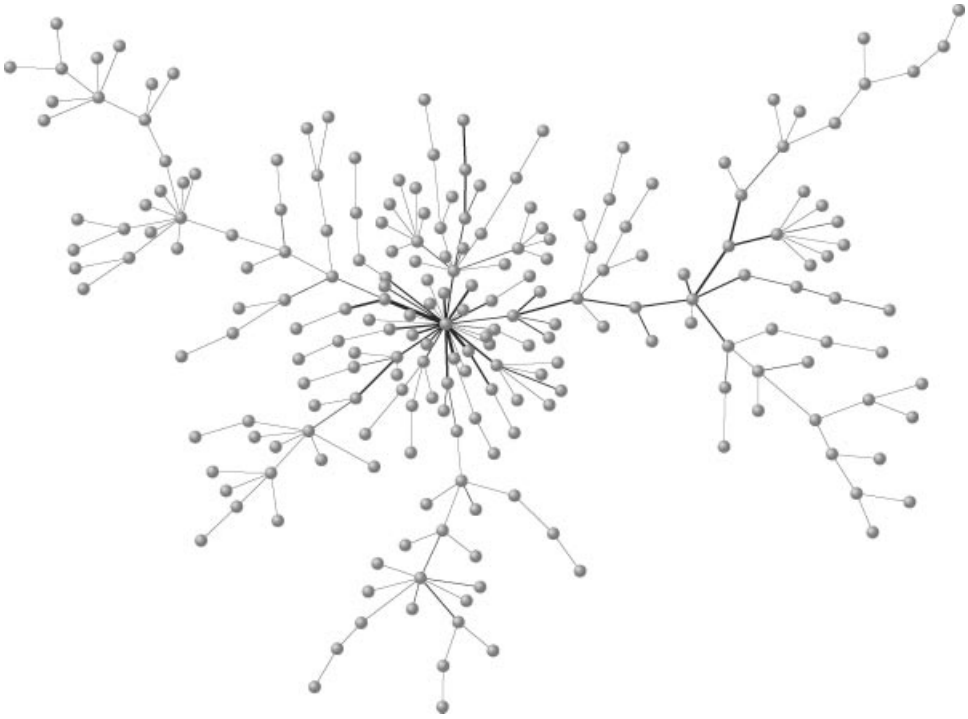

FIG. 6. An example of a scientogram corresponding to the Europe scientific domain in 2002.

TABLE 2. Comparison of the run times (expressed in seconds) of all the algorithms for the scientograms case study.

#	Domain (year)	#Nodes	#Links	Original PF	Binary PF	Fast PF	MST-PF (low-complexity)	MST-PF (practical)
1	China (2002)	212	8541	28.31	1.87	0.180	0.00398	0.00400
2	Japan (2002)	213	9028	28.70	1.69	0.179	0.00422	0.00423
3	France (2002)	216	10087	31.79	2.14	0.187	0.00473	0.00470
4	Peru (2002)	218	8485	30.97	2.20	0.200	0.00396	0.00397
5	Germany (2002)	218	11745	35.53	2.20	0.214	0.00555	0.00552
6	UK (2002)	218	13567	37.74	2.21	0.196	0.00651	0.00646
7	Europe (2002)	218	17242	40.10	2.19	0.193	0.00852	0.00845
8	USA (2002)	218	18132	40.59	2.21	0.196	0.00909	0.00904
9	World (2002)	218	20154	40.19	2.20	0.195	0.01031	0.01020
10	Cuba (2004)	219	10644	34.15	2.15	0.200	0.00500	0.00499
11	Spain (1994)	219	13478	37.65	2.21	0.194	0.00644	0.00639
12	Cuba (2006)	221	11286	35.71	2.08	0.204	0.00532	0.00529
13	Spain (1998)	223	16226	47.61	2.94	0.211	0.00789	0.00786
14	Venezuela (2005)	239	15415	53.86	3.12	0.265	0.00750	0.00747
15	Spain (2002)	240	19183	57.59	3.43	0.258	0.00976	0.00973
16	Spain (2004)	240	23430	60.21	3.42	0.257	0.01216	0.01208
17	Chile (2004)	242	17914	59.09	3.04	0.269	0.00901	0.00897
18	Mexico (2005)	250	21264	75.24	4.11	0.300	0.01104	0.01089
19	Portugal (2005)	254	22179	84.45	4.99	0.321	0.01153	0.01143
20	Argentina (2005)	263	19562	82.84	4.10	0.346	0.00998	0.00994

As shown, the MST-Pathfinder variant is some orders of magnitude faster than the Fast Pathfinder variant that itself outperforms the Binary and the original Pathfinder algorithms. More precisely, the MST-Pathfinder is 6200 times faster in average than the original Pathfinder, 360 times faster than the Binary algorithm, and 30 times faster than the Fast Pathfinder. In this sense, the comparison with the original algorithm, having a complexity of $O(n^4)$, the Binary variant, $O(n^3 \cdot \log n)$, and the Fast one, $O(n^3)$, clearly demonstrates how the $O(n^2 \cdot \log n)$ variant speeds up the pruning of the networks.

Concerning the second point, the comparison between the low-complexity and the practical variants of the MST-Pathfinder, we can see that the variant having the largest

DOI: 10.1002/asi

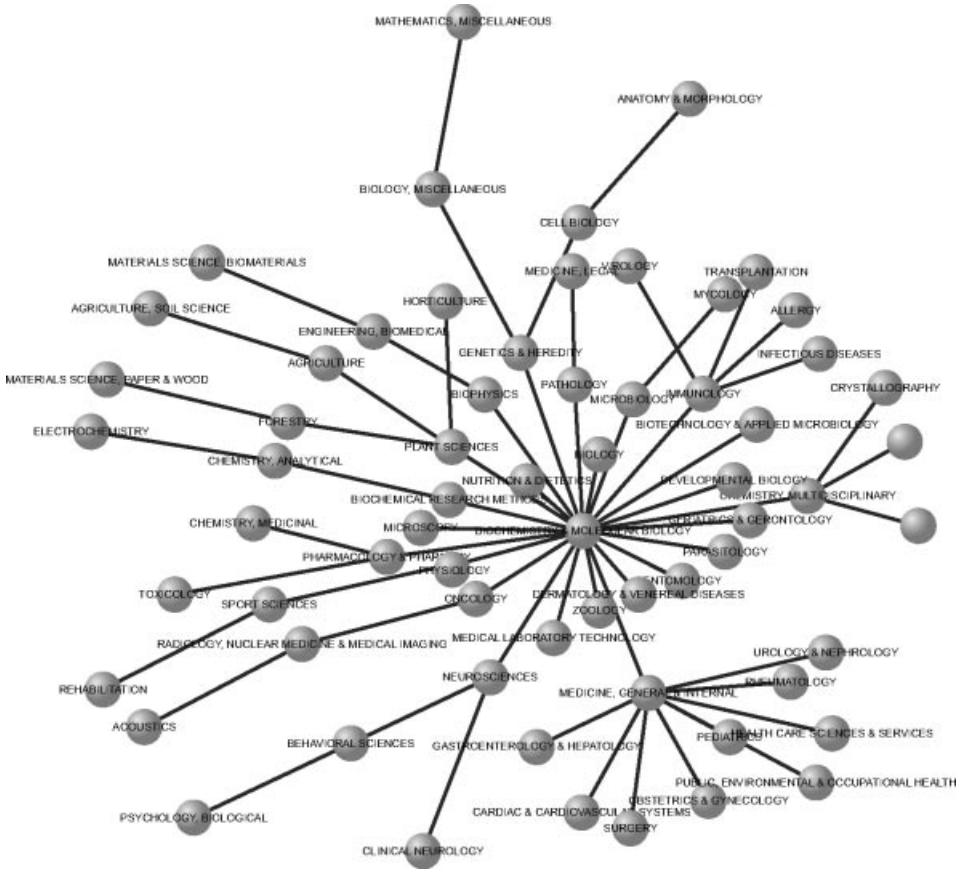

FIG. 7. A detailed view of the center of the Europe scientogram, processed by the original Pathfinder algorithm.

theoretical time complexity ($O(n^3)$ instead of $O(n^2 \cdot \log n)$) obtains the fastest results. In fact, the run time is slightly better and the improvement is more or less the same on these medium networks—0.5% faster in average—than for the case of larger networks (see the section of the first case study).

Description of Some Implementation Issues

In this example, we have used the Europe data of 2002. The data are directly extracted from a database of cocitation measures. Some specific criteria are set up in order to select a subset of the whole database restricted to the countries, the journals, or the authors. The resulting file encodes a fully connected network, with labeled nodes and weighted links, ready to be pruned. Thus, the first step is to use the MST-Pathfinder algorithm to prune this network. The computing time for this step is roughly 9 ms (see Table 2). From 218 nodes (ISI-JCR categories) and 17242 links, the network is pruned until having only 217 links, becoming a tree. So, with this map, a highly simplified network has been obtained in a fast way.

The next step is to print the map in a convenient way. Many graphical libraries can be used for this purpose, but the best we have found so far is the GraphViz library. GraphViz is an open source network drawing software, freely provided by AT & T Labs, and available at *http://www.graphviz.org*. It integrates the Kamada-Kawai algorithm in the form of the *neato* utility. This utility exports in diverse graphical formats a description of a network done with a proprietary language, the DOT language (Gansner & North, 2000). Thus, the second step was to convert the previous network in the format accepted by this library. The computing time for this step is 0.6 ms.

The last step is to generate the graphical output from the DOT description using *neato*. Actually, as these maps are designed for online consultation, the Scalable Vector Graphics (SVG) format was chosen. The time to generate the SVG image is 1300 ms. The following command was used to generate this map, once the library is installed:

```
dot -Kneato -Tsvg -o Europe.svg Europe.dot
```

With this procedure, we are able to generate the full Europe scientific domain scientogram without exceeding 1.5 seconds. This enables the generation of these maps in an online way providing a faster interaction for the user. This also enables the generation of a high number of maps in a short time to compare them among a time line or geographically, which is one of our aims in the short future.

The final result is shown in Figure 9. Two detailed views, produced by the original Pathfinder algorithm and the MST-Pathfinder algorithm are presented in Figures 7 and 8 to show that they are equal. In fact, the two files used to produce these pictures were identical in a binary way.

DOI: 10.1002/asi

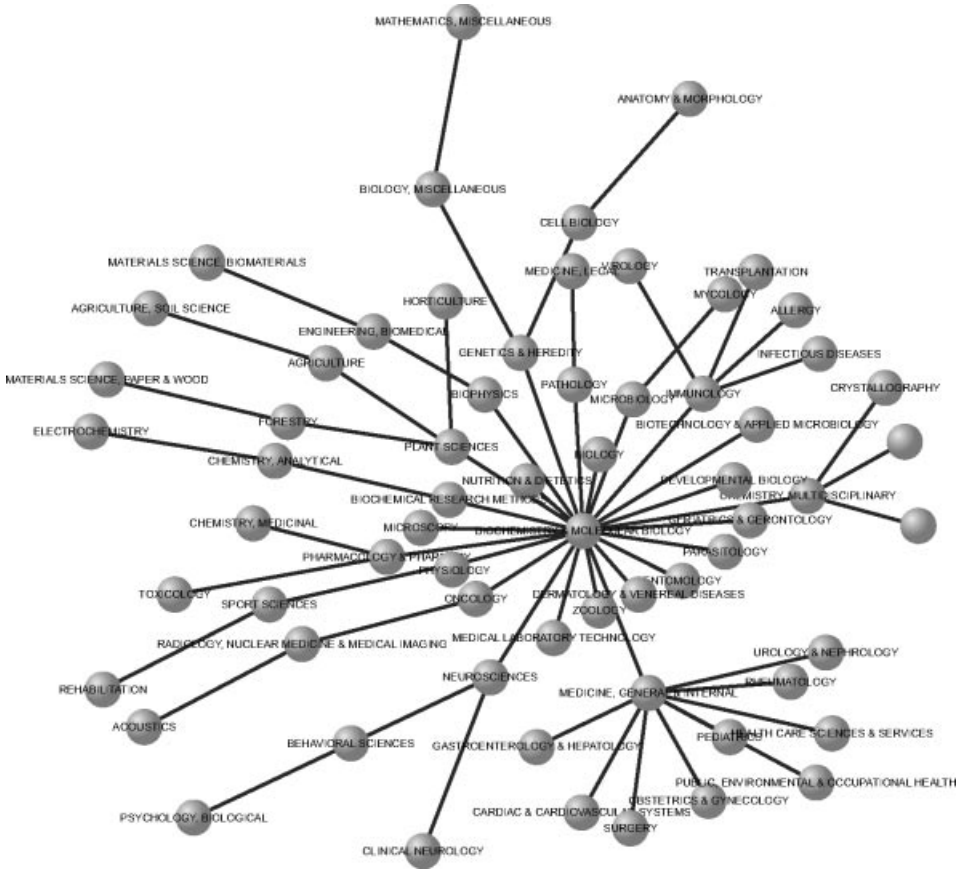

FIG. 8. A detailed view of the center of the Europe scientogram, processed by the MST-Pathfinder algorithm.

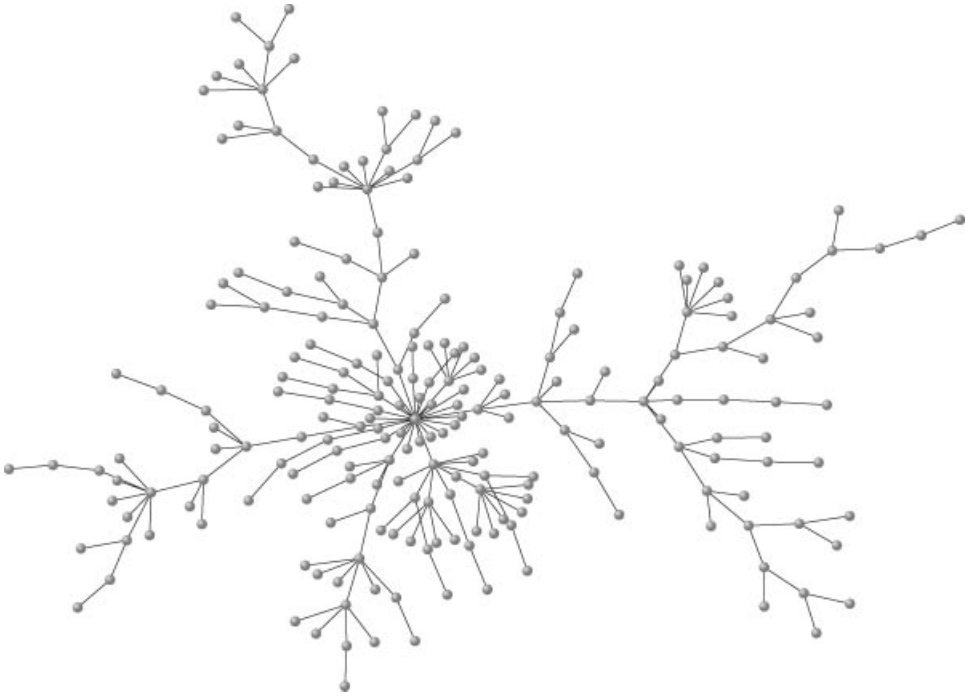

FIG. 9. An example of the output corresponding to the Europe data, computed by the Kamada-Kawai algorithm.

DOI: 10.1002/asi

TABLE 3. Summary of the properties of the Pathfinder variants.

Name of the algorithm	Application domain	Time complexity (for $q = n - 1$)	Space complexity	Approach in algorithm theory
Original PF	Any valid values for q and r, (un-)directed graphs	$O(q \cdot n^3) = O(n^4)$	$2 \cdot q \cdot n^2 = 2 \cdot n^3 - 2 \cdot n^2$	Dynamic programming
Binary PF	Any valid values for q and r, (un-)directed graphs	$O(\log q \cdot n^3) = O(n^3 \cdot \log n)$	$4 \cdot n^2$	Dynamic programming
Fast PF	Any valid values for r, $q = n - 1$, (un-)directed graphs	$O(n^3)$	$2 \cdot n^2$	Dynamic programming
MST-PF (low-complexity)	$r = \infty$, $q = n - 1$, undirected graphs	$O(n^2 \cdot \log(n))$	$3 \cdot n^2 + n$	Greedy approach
MST-PF (practical)	$r = \infty$, $q = n - 1$, undirected graphs	$O(n^3)$	$3 \cdot n^2 + n$	Greedy approach

Conclusion

In this article, we have introduced a new variant of the Pathfinder algorithm for the specific parameter setting used in many applications (q must be fixed to $n - 1$ and r to ∞), enabling us to drastically decrease the run time of the original algorithm. Our proposal is based on the fact that the union of the MSTs is equal to PFNET(∞, $n - 1$), what allows us to reduce the time complexity from $O(n^4)$ to $O(n^2 \cdot \log n)$. Notice that MST-Pathfinder saves also a noticeable amount of memory in comparison to the original and the Binary variant of Pathfinder. See Table 3 to have a global view of the properties of the Pathfinder variants.

The experimental comparison on the laboratory problem conducted on large networks has demonstrated the fast run time when networks are fully connected. The other experimental comparison of the run time for 20 medium networks from real world domains has proved the ability of the new proposal to prune medium networks in real time.

Acknowledgments

This work was funded by the Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica 2004–2007 and the Fondo Europeo de Desarrollo Regional (FEDER) as part of research projects SEJ-2004-08358-C02-01 and SEJ2004-08358-C02-02. We 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. (1965). Dynamic programming and modern control theory. New York: Academic Press.

Börner, K., Sanyal, S., & Vespignani, A. (2007). Networks science. Annual Review of Information Science and Technology (ARIST), 41, 537–606.

Breiger, R.L. (2004). Handbook of data analysis. In (pp. 505–526). London: Sage Publications.

Buzydlowski, J. (2002). A comparison of self-organizing maps and pathfinder networks for the mapping of co-cited authors. Unpublished doctoral dissertation. 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. (2004). Information visualization: Beyond the horizon. Berlin, Germany: Springer.

Chen, C., Kuljis, J., & Paul, R. (2001). Visualizing latent domain knowledge. IEEE Transactions on Systems, Man and Cybernetics, Part C, 31(4), 518–529.

Chen, C., & Morris, S. (2003). Visualizing evolving networks: Minimum spanning trees versus pathfinder networks. In Proc. IEEE symposium on information visualization (INFOVIS) (pp. 67–74).

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

De Moya-Anegón, F., Vargas-Quesada, B., Chinchilla-Rodríguez, Z., Herrero-Solana, V., Corera-Álvarez, E., & Munoz-Fernández, F. J. (2005). Domain analysis and information retrieval through the construction of heliocentric maps based on ISI-JCR category cocitation. Information Processing & Management, 41(6), 1520–1533.

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.

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

Floyd, R. (1962). Agorithm 97: Shortest path. Communications of the ACD, 5(6), 345.

Gansner, E.R., & North, S.C. (2000). An open graph visualization system and its applications to software engineering. Software – Practice and Experience, 30(11), 1203–1233.

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

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

Kravitz, D. (2007). Two comments on minimum spanning trees. The Bulletin of the ICA, 49, 7–10.

Kruskal, J. (1956). On the shortest spanning subtree and the traveling salesman problem. Proc. of the American Mathematical Society, 7(1), 48–50.

Kudikyala, U., & Vaughn, R. (2005). Software requirement understanding using pathfinder networks: Discovering and evaluating mental models. Journal of Systems and Software, 74(1), 101–108.

Martino, F., & Spoto, A. (2006). Social network analysis: A brief theoretical review and further perspectives in the study of information technology. Psychology Journal, 4(1), 53–86.

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

DOI: 10.1002/asi

Moya-Anegón, F. de, Vargas-Quesada, B., Chinchilla-Rodríguez, Z., Herrero-Solana, V., Corera-Álvarez, E., & Munoz-Fernández, F.J. (2007). Visualizing the marrow of science. Journal of the American Society for Information Science and Technology, 58(14), 2167–2179.

Prim, R. (1957). Shortest connection networks and some generalizations. Bell System Technical Journal, 36(6), 1389–1401.

Quirin, A., Cordón, O., Santamaría, J., Vargas-Quesada, B., & Moya-Anegón, F. (2008). A new variant of the pathfinder algorithm to generate large visual science maps in cubic time. Information Processing & Management, 44(4), 1397–1409.

Salton, G., & Bergmark, D. (1979). A citation study of computer science literature. IEEE Transaction on Professional Communication, 22, 146–158.

Schvaneveldt, R.W. (1990). Pathfinder associative networks. Norwood, NJ: Ablex.

Shope, S., DeJoode, J., Cooke, N., & Pedersen, H. (2004). Using pathfinder to generate communication networks in a cognitive task analysis. In Proc. of the human factors and ergonomics society, 48th annual meeting (pp. 678–682).

Small, H., & Garfield, E. (1985). The geography of science: Disciplinary and national mappings. Journal of Information Science, 11(4), 147–159.

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

Warshall, S. (1962). A Theorem on Boolean matrices. Journal of the ACM, 9(1), 11–12.

DOI: 10.1002/asi