Available online at www.sciencedirect.com
SCIENCE DIRECT®
Decision Support Systems 38 (2004) 473–487
Decision Support Systems
www.elsevier.com/locate/dsw

Fighting organized crimes: using shortest-path algorithms to identify associations in criminal networks

Jennifer J. Xu*, Hsinchun Chen

Department of Management Information Systems, University of Arizona, Tucson, AZ 85721, USA

Received 1 January 2003; accepted 21 July 2003
Available online 7 October 2003

Abstract

Effective and efficient link analysis techniques are needed to help law enforcement and intelligence agencies fight organized crimes such as narcotics violation, terrorism, and kidnapping. In this paper, we propose a link analysis technique that uses shortest-path algorithms, priority-first-search (PFS) and two-tree PFS, to identify the strongest association paths between entities in a criminal network. To evaluate effectiveness, we compared the PFS algorithms with crime investigators' typical association-search approach, as represented by a modified breadth-first-search (BFS). Our domain expert considered the association paths identified by PFS algorithms to be useful about 70% of the time, whereas the modified BFS algorithm's precision rates were only 30% for a kidnapping network and 16.7% for a narcotics network. Efficiency of the two-tree PFS was better for a small, dense kidnapping network, and the PFS was better for the large, sparse narcotics network.

Keywords: Link analysis; Shortest-path algorithm; Concept space; Law enforcement; Crime investigation; Organized crime

1. Introduction

Organized crimes such as terrorism, narcotics violations, armed robbery, and kidnapping often involve multiple offenders who are connected through various relationships (e.g., kinship, friendship, co-workers, or business associates) [16]. These criminals can be treated as a network in which they interact and play different roles in illegal activities [22]. For instance, a narcotics network may consist of interrelated criminals who are responsible for handling the supply, distribution, sale, and smuggling of drugs, or even money laundering. Members in a terrorist network may have shared religious beliefs or attended terrorist training together previously so that they trust each other and cooperatively plan and commit terrorist attacks [20]. In a broader sense, a criminal network may be composed of a variety of entities (e.g., organizations, locations, vehicles, weapons, properties, bank accounts, etc.) in addition to persons. Learning associations between these entities is a critical part of uncovering criminal activities and fighting crimes. To achieve this goal, crime investigators often employ a method called link analysis [8,16,24], which can help generate investigative leads and uncover missing information that may be buried in a criminal network. In a

* Corresponding author. Tel.: +1-520-626-9239.
E-mail addresses: jxu@eller.arizona.edu (J.J. Xu), hchen@eller.arizona.edu (H. Chen).

0167-9236/$ - see front matter © 2003 Published by Elsevier B.V.
doi:10.1016/S0167-9236(03)00117-9

narcotics network, for example, link analysis may reveal that a group of offenders actually belong to the same drug supply chain. In a homicide crime, link analysis may find "hidden", intermediate persons connecting the victim with the suspect who denies knowing the victim.

Link analysis usually consists of two major tasks: extracting information about entity associations from raw data (e.g., telephone records, surveillance logs, and crime reports) and constructing a network representation, and identifying associations between seemingly unrelated entities in a network. Both tasks can be very time-consuming and labor-intensive. Current link analysis practice in law enforcement is mainly an ad-hoc manual process. To solve a crime, investigators may spend a large amount of time performing extensive database searches, reading crime reports, and looking for clues of criminal associations. Although some software packages have been labeled with "link analysis tools", they provide only visual representations of criminal networks and are "still not doing the analysis" [24]. Because of these problems, link analysis is used only for high-profile cases. Effective and efficient link analysis techniques are needed to help fight crime [22].

To address the lack-of-technique problem, we propose using two variations of the classical shortest-path algorithms [11] for link analysis. Our evaluation studies assess both the effectiveness and efficiency of the proposed algorithms. The effectiveness issue concerns whether association paths found by the proposed algorithms are more useful for uncovering investigative leads than those found by a modified breadth-first-search (BFS) algorithm. The modified BFS algorithm to a large extent simulated the manual approach of association search by crime investigators and was used as a benchmark technique for effectiveness comparison. The efficiency issue concerns which shortest-path algorithm is faster in what type of networks.

The rest of the paper is organized as follows. Section 2 reviews the literature on link analysis and the shortest-path algorithms. Section 3 presents the modified BFS algorithm. The two proposed shortest-path algorithms are introduced in Section 4. Evaluation and results are presented and discussed in Section 5. In Section 6, we conclude the paper and suggest directions for future work.

2. Literature review

In this section, we review network construction techniques proposed in previous research and existing link analysis tools. We then introduce the algorithms for computing shortest paths in a graph.

2.1. Link analysis

2.1.1. Network construction

To entail link analysis, an indispensable task is to extract information about entities and their associations from large amounts of raw data and convert the information into a network representation. Usually, entities are represented by nodes and associations between them are represented by links in a network. Different network construction methods may be needed, depending on whether the raw data are structured database records or unstructured textual documents.

Several techniques have been developed for constructing network representations of structured data records. For example, Goldberg and Senator [14] suggested that consolidation and link formation operations be performed on transactional data records during investigations of financial crimes. Consolidation is a process of "disambiguating and combining identification information into a unique key which refers to specific individuals" [14]. Links or associations between consolidated individuals are formed based on a set of heuristics such as whether the individuals have shared addresses, shared bank accounts, or related transactions. This technique has been employed by the U.S. Department of the Treasury to detect money laundering transactions and activities [15]. A different network construction method used by COPLINK Detect [17] is based on the concept space approach developed by Chen and Lynch [6]. A concept space can be treated as a network in which nodes represent domain-specific concepts and links represent weighted co-occurrence associations between concepts [17]. In COPLINK Detect, nodes are records of entities (persons, organizations, vehicles, and locations) stored in crime databases. In such a network, an association exists between a pair of entities if they appear together in the same criminal incident. The more frequently they occur together, the stronger the association. The

concept space approach is primarily a statistic-based approach and differs from the heuristic-based one in Ref. [14].

Some other techniques can build networks based on information extracted from unstructured data or textual documents. Lee [21] developed a technique to construct criminal networks from free texts. This approach can extract entities and events from textual crime reports by applying a large collection of predefined patterns. Associations among extracted entities and events are formed using relation-specifying words and phrases. For example, the phrase "member of" indicates an entity-to-entity association between an individual and an organization; the word "arrest" may suggest an entity-to-event association between an individual and an arrest event. This approach relies heavily on a fixed set of predefined patterns and rules and thus has a limited scope of application. The concept space approach [6,17], as mentioned earlier, can also be used to construct networks from textual documents. Instead of using structured data from databases, it uses noun phrases extracted from crime reports as entities to build a criminal network. An association or co-occurrence relationship exists between a pair of entities as long as they appear together in the same report. However, the noun phrases extracted may not necessarily be the entities that interest the crime investigators. Success of this type of network construction approaches, to a large extent, depends on the development of named-entity extraction technique [7], which is the automatic identification from text documents of the names of entities of interest, such as date, time, number expression, person, location, and organization [5,7].

2.1.2. Link analysis tools

In addition to network construction, another important link analysis task is searching for possible associations between entities. However, most existing link analysis tools can only visualize criminal networks and do not offer much help with association search. This section will provide a review of existing link analysis tools.

The earliest link analysis tool is the Anacapa charting system [16], which has been used extensively in law enforcement since its introduction. Based on human-extracted association information, the system can generate a two-dimensional visual representation of a network with different symbols representing different types of entities. However, this tool does not facilitate association search and an investigator must manually examine the network display to find association paths between entities or confirm initial suspicions about specific suspects [24]. Other link analysis tools such as Netmap [15] and Analyst's Notebook [19] are also designed for network visualization rather than for association search.

A link analysis tool called Watson [2] can search and identify direct associations between entities by querying databases. Given a specific entity such as a person's name, Watson automatically forms a query to search for other records that are related to the person. For example, an analyst may want to find out who is related to a kidnapped child. The related records found by Watson, which may include the child's relatives, friends, or other acquaintances, will be linked to this child and presented in a link chart. COPLINK Detect [17] can also be treated as a link analysis tool which provides direct association search functionality.

In Section 2.2, we review shortest-path algorithms, which we propose to address the problem of identifying the strongest associations between entities that are not directly related. Although these algorithms have been studied and employed widely in other domains, their importance and relevance to link analysis have not yet been recognized in law enforcement.

2.2. Shortest-path algorithms

Shortest-path algorithms are a type of graph search algorithms. They can identify the optimal paths between nodes in a graph (i.e., a network) by examining link weights. Conventional shortest-path algorithms have been used in many applications such as robot motion planning [4], computer network routing [23], transportation and traffic control [26], critical path computation in PERT charts, etc. Recently, a neural network approach in artificial intelligence has been proposed for shortest-path computation [1,3]. In this section, we review the conventional approaches and briefly introduce the neural network approach.

The Dijkstra algorithm [11] is the classical method for computing the shortest paths from a single source node to every other node in a weighted graph. Most other algorithms for solving this problem are based on this algorithm but have improved data structures for implementation [12]. For example, the priority-first-search (PFS) algorithm [9] is faster than the Dijkstra algorithm because of the use of a priority queue.

Unlike the classical Dijkstra algorithm, the two-tree Dijkstra algorithm computes the shortest path from a single source node to a single destination node, rather than to every other node in a graph. Previous studies have demonstrated that the two-tree Dijkstra algorithm can be much faster than the Dijkstra algorithm. According to Helgason et al. [18], in most cases, the Dijkstra algorithm generated a shortest-path tree containing approximately 50% of the nodes in a graph before the shortest path between a source node and a destination node was found. Shortest-path trees generated by the two-tree Dijkstra algorithm, in contrast, contained only 6% of the nodes in the graph. This might save a substantial amount of computational time.

Some researchers have proposed neural network approaches to solving the shortest-path problem. Araujo et al. [3] extended Ali and Kamoun's study [1] and applied a two-layer Hopfield net to the shortest-path problem. In their Hopfield net, each neuron corresponds with a link in a graph. The value of a neuron is 1 if the link it represents participates in the shortest path and 0 otherwise. It has been found that the two-layer Hopfield net could be faster than conventional shortest-path algorithms because of its parallel architecture. However, these proposed Hopfield net approaches work only for networks of small size (e.g., 40 in Ref. [3]).

In summary, previous studies have proposed some techniques for network construction in link analysis. However, little research has been done to address the association search problem. Specifically, an effective and efficient link analysis technique is needed to find association paths between two or more source entities not directly related. Moreover, the paths found should reveal strong associations between entities so that important investigative leads can be uncovered. We propose to use the shortest-path algorithms to achieve this goal. To compare the proposed algorithms with current link analysis practices, in our pilot study, we recorded and analyzed the association search processes of crime investigators experienced in link analysis. We found that the typical association search approach can be described as a breadth-first search [9]. However, such an approach cannot guarantee finding the strongest associations between entities and thus may not successfully generate investigative leads. In the next section we present the modified BFS algorithm, which simulates the typical association search.

3. The modified BFS algorithm

Since existing link analysis tools are limited to direct association search, crime investigators must explore links manually when they have entities that are not directly related. We found that a typical search starts with a single source entity and incrementally builds up an association path during link exploration. For example, a crime investigator may need to find associations between two seemingly unrelated drug offenders. In this case, the crime investigator may start with one offender's name and use a link analysis tool to find all entities that are associated with the offender in previous crimes. By reading each crime report, the investigator can determine whether a link is useful for generating a new lead to connect the two offenders. He then selects those useful links and does further searches, in which entities associated with the newly selected entities from the previous round are examined. He keeps exploring new entities until an association path that connects the two offenders is found.

Such a search process is very similar to a graph traversal algorithm called BFS [9], except that an investigator may consider link usefulness during exploration. To describe this algorithm, we use the notation found in Ref. [18]. The input of this algorithm is a weighted directed graph $G=(N, A)$ with a node set N and a link set A, $|N|=n$, $|A|=m$. A nonnegative number, l_{ij}, is used to represent the weight of the link $(i, j)\in A$. Each node $u\in N$ has an incoming link set, $\mathrm{In}(u)$, and an outgoing link set, $\mathrm{Out}(u)$. Since our criminal networks are undirected graphs, $\mathrm{In}(u)=\mathrm{Out}(u)$.

Starting at a source node s, BFS can find paths leading to a target node t. It works by maintaining a

traversal tree T rooted at the node s. In this tree, the child nodes of a specific node u are u's outgoing neighbors in the graph G. Initially, T contains only s. The algorithm then collects all the outgoing neighbors of s in G and sets them as the child nodes of s. For each child node of s, the algorithm further finds its children and adds them to the tree. This procedure is repeated until the target node t is reached. The time complexity of a BFS algorithm is $O(n+m)$ [9].

As indicated earlier, a crime investigator may not explore all entities associated with a specific entity but selects only those having strong associations. We therefore modified the BFS algorithm so that when it finds the children of a node, it selects only those neighbors that have a link weight greater than a predefined threshold value. The modified BFS algorithm is presented in Fig. 1.

Notice that multiple paths may exist between the source entities s and t. BFS simply finds one such path and does not guarantee to identify the strongest associations between source entities. This suggests that the shortest-path algorithms may be a better option.

4. Shortest-path algorithms

To find the strongest associations between two or more source entities, we propose to employ conventional shortest-path algorithms. However, to apply the algorithms, a network representation transformation must be made.

4.1. Network representation transformation

In our criminal networks, the strength of an association between two directly connected nodes is represented by their link weight, which is a number between zero and one. A link weight can be treated as a probability measure indicating how likely it is that two nodes are related. In general, the probability of a set of mutually independent events occurring together is the product of the probabilities of the individual events. Therefore, if two nodes are not connected directly but by a path consisting of a sequence of intermediate links, the strength of the association between these two nodes should be the product of the weights of these intermediate links. For example,

Modified BFS algorithm
{*This modified BFS algorithm computes the paths from the first node in K to every other node in K. K may contain multiple source nodes*}

```
Begin
   Initialize:
   s = the 1st element of K; p_s = s; {p_i is the parent node of i}
    p_i = 0  for all i ∈ N, i ≠ s;
   T = {s}, L_0 = {s}; {L_i stores the current nodes}
   i = 0;
   while (L_i ≠ Ø)
       L_{i+1} = Ø;  {L_{i+1} stores the child nodes of the current nodes in L_i}
       for each u ∈ L_i do
           {Explore a link only if its length is less than the threshold value 1, which
           corresponds to link weight of 0.5 in the original, untransformed graph}
           for each (u, v) ∈ Out(u) such that l_uv < 1 do
               if v ∈ T then {Include v into the tree and set u as the parent of v}
                   T = T ∪ {v};
                   p_v = u;
                   L_{i+1} = L_{i+1} ∪ {v};
           end
       end
       i = i + 1;
       if v ∈ K, then K = K-{v};
       if (K = Ø) break; {Stop when all source nodes in K are included in the tree}
   endwhile;
end
```

Fig. 1. The modified BFS algorithm.

if node A and node C are connected through node B, and the weights of the intermediate links (A–B) and (B–C) are 0.5 and 0.8, respectively, then the weight of the path (A–B–C) would be 0.4. To find the strongest association between a pair of nodes, therefore, is to find the path with the largest weight product. Fig. 2 presents an illustrative example.

In this figure, the number beside each link is that link's weight or association strength. Two paths, (A–B–C–D) and (A–E–D), exist between the source node A and the destination node D. The association strength of path (A–B–C–D) is 0.28 ($0.5 \times 0.8 \times 0.7$), and the association strength of path (A–E–D) is 0.24 (0.8×0.3). Therefore, path (A–B–C–D) has a stronger association between node A and node D than path (A–E–D).

Although the shortest-path algorithms can identify the optimal path between a pair of nodes, they cannot be used directly to identify the strongest association between the two nodes. This is because of the following two representation problems:

(a) In a general weighted graph, the weight of a link represents the distance or cost of traveling from one end of the link to the other. Therefore, a low weight is preferred to a high weight. However, a link weight in a criminal network is an indicator of how strongly the two nodes are related to each other. Thus, a high weight is preferred to a low weight.
(b) The shortest path is often computed based on the minimum total weight, which is the sum of the weights of the links along this path. However, our objective is to find a path with the maximum weight product.

In order to address the two representation problems, we transformed the link weight in a criminal network to a distance measure in a new graph representation. In this new graph, the nodes are the same as those in the original network, but the new link weights are computed based on the original weights using a simple logarithmic transformation:

$$l = -\ln w \quad 0 < w \leq 1$$

where l is the link weight in the new graph and w is the corresponding link weight in the original network. Given this transformation, we postulate the following axioms:

(1) All link weights in the new graph are nonnegative numbers.
(2) A lower link weight in the new graph corresponds with a higher link weight in the original network.
(3) The shortest path (using summation of link weights) between a pair of nodes in the new graph generates a path with the maximum link weight product among all the alternative paths between these two nodes in the original network.

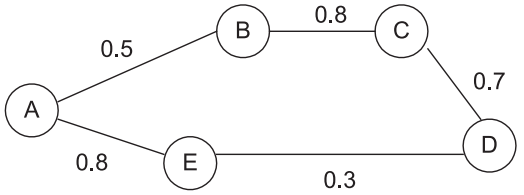

Fig. 2. Two indirectly connected nodes (A and D).

Proof. Proofs of these three axioms are fairly straightforward, following the transformation equation directly:

Axiom 1. Since $0 < w \leq 1$, thus $\ln w \leq 0$, which suggests that $-\ln w \geq 0$.

Axiom 2. Let $l_1 < l_2$, then $-\ln w_1 < -\ln w_2$ or $\ln w_1 > \ln w_2$. Since $\ln w$ is a monotonic increasing function, it follows that $w_1 > w_2$.

Axiom 3. Consider the shortest path, say P, between a pair of nodes A and B. P consists of a set of links with weight $(l_1, l_2, \ldots, l_p)$, $1 \leq p \leq n$, where n is the total number of nodes in this graph. The total length of this path is $\sum_{i=1}^{p} l_i$. Consider another path between node A and node B, say Q, consisting of another set of links with weight $(l_1', l_2', \ldots, l_q')$, $1 \leq q \leq n$. The total length is $\sum_{i=1}^{q} l_i'$.

Because P is the shortest path between node A and node B, we know that

$$\sum_{i=1}^{p} l_i < \sum_{i=1}^{q} l_i'.$$

Since $l_i = -\ln w_i$ and $l_i' = -\ln w_i'$ by definition, we have $\sum_{i=1}^{p} \ln w_i > \sum_{i=1}^{q} \ln w_i'$.

It follows that $\exp(\sum_{i=1}^{p} \ln w_i) > \exp(\sum_{i=1}^{q} \ln w_i')$, which suggests that $\prod_{i=1}^{p} w_i > \prod_{i=1}^{q} w_i'$. □

Axiom 1 ensures that the new graph does not contain negative-weight links, which is a necessary condition for the shortest-path algorithms [12]. Axioms 2 and 3, respectively, address the two representation problems. Therefore, with such a transformation, we are able to use conventional shortest-path algorithms to identify the strongest associations between a pair of nodes or entities in a criminal network.

4.2. Shortest-path algorithms

We propose using the PFS [9] and the two-tree Dijkstra algorithm [18]. Both algorithms can compute the shortest path between two source nodes. Considering the situation where an investigator needs to find associations between more than two entities, we repeatedly use the algorithms to identify the strongest associations among multiple source nodes.

We assume that a group of nodes is strongly associated if each pair of nodes in the group is strongly associated. That is, given k source nodes $(u_1,u_2,\ldots,u_k)$, we first find the shortest paths between u_1 and every other source node (u_2 through u_k). Then, we find the shortest paths between u_2 and the remaining source nodes (u_3 through u_k). Such a process is repeated until the shortest paths between all possible pairs of the k source nodes are found. The total number of these shortest paths is $k(k-1)/2$. It is possible that some of these paths share common links. If this happens, we combine the common links to avoid redundancy.

4.2.1. The modified PFS algorithm

The PFS algorithm [9] is a variation of the classical Dijkstra algorithm [11]. The algorithm works by maintaining a shortest-path tree T rooted at a source node s. T contains nodes whose shortest distances from s are already known. Each node u in T has a parent, which is represented by p_u. A set of labels, d_u, is used to record the distances from the node u to s. Initially, T contains only s. At each step, we select from the candidate set Q a node with the minimum distance to s and add this node to T. Once T includes all nodes in the graph, the shortest paths from the source node s to all the other nodes have been found. PFS differs from the Dijkstra algorithm because it uses an efficient priority queue for the candidate set Q.

With modifications, PFS can be used to compute the shortest paths from a single source node to a set of specified nodes in the graph. That is, given a set of nodes $K \subseteq N$, $|K| = k \geq 2$ and a source node $s \in K$,

Modified PFS algorithm
{This modified PFS algorithm computes the shortest paths from the first node in K to every other node in K}

Begin
  Initialize:
  s = the 1st element of K; $d_s = 0$, $p_s = s$; $d_i = \infty$,
  $p_i = 0$ for all $i \in N$, $i \neq s$;
  $T = \{s\}$; $Q = \{s\}$.
  while ($K \neq \emptyset$) *{Search Q for the node with minimum distance to s}*
    $u = \{i: d_i \leq d_j, i, j \in Q, i \neq j\}$;
    $Q = Q - \{u\}$;
    {The shortest path between u and s has been found and u is added to T}
    $T = T \cup \{u\}$;
    for each $(u, v) \in Out(u)$ **such that** $d_u + l_{uv} < d_v$ **do**
      {Update the distance label of v}
      $d_v = d_u + l_{uv}$;
      $p_v = u$;
      if $v \notin Q$ **then** $Q = Q \cup \{v\}$;
    end
    if $u \in K$, **then** $K = K - \{u\}$;
  endwhile;
end.

Fig. 3. The modified PFS algorithm.

the modified PFS algorithm can compute the shortest paths from s to all $u \in K$ and $u \neq s$. We therefore modify the algorithm so that it stops as soon as all $u \in K$ are included in the shortest-path tree T. Note that when K contains only two nodes, the problem is reduced to a one-to-one shortest-path problem [18]. The modified PFS algorithm is presented in Fig. 3.

When computing the shortest paths from K's second node to every other node in K, we repeat this procedure. Note that we do not need to compute the shortest path from the second node to the first node again, since it has already been computed. This procedure is repeated $k-1$ times until the shortest paths between all possible pairs of the nodes in K have been found.

We implement the priority queue using a heap tree for the candidate set Q. At each iteration of the *while* loop, it takes $O(\log n)$ time to search for the minimum element u from Q, and $O(|\mathrm{Out}(u)|\log n)$ time to examine and update the distances of incident links of u. Thus, the execution time for the *while* loop is $\Sigma_{u \in N}(1 + |\mathrm{Out}(u)| = \log n$ or $O((n+m)\log n)$, because $\Sigma_{u \in N}|\mathrm{Out}(u)| = m$. As a result, the overall time complexity for computing all shortest paths for k nodes is $O(k(n+m)\log n)$. PFS is faster than the Dijkstra algorithm, whose time complexity is $O(k(n^2+m))$ [12].

Two-Tree PFS algorithm
{*Two-tree PFS computes the shortest path between node s and node t*}

Begin
Initialize:
$d_s^s = 0, p_s^s = s, T^s = \{s\}; Q^s = \{s\};\ p_i^s = 0, d_i^s = \infty$ *for all* $i \in N$;
$d_t^t = 0, p_t^t = t, T^t = \{t\}; Q^t = \{t\};\ p_i^t = 0, d_i^t = \infty$ *for all* $i \in N$.
while $(T^s \cap T^t = \emptyset)$ **do** {*Search* Q^s *for the node with minimum distance to s*}
$u = \{i:\ d_i^s \le d_j^s, i, j \in Q^s, i \neq j\}$;
$Q^s = Q^s - \{u\}$;
{*The shortest path between u and s has been found and u is added to* T^s}
$T^s = T^s \cup \{u\}$;
{*Examine outgoing links of u*}
for each $(u, v) \in Out(u)$ **such that** $d_u^s + l_{uv} < d_v^s$ **do**
$d_v^s = d_u^s + l_{uv}$;
$p_v^s = u$;
if $v \notin Q^s$ **then** $Q^s = Q^s \cup \{v\}$;
end

{*Search* Q^t *for the node with minimum distance to t*}
$v = \{i:\ d_i^t \le d_j^t, i, j \in Q^t, i \neq j\}$;
$Q^t = Q^t - \{v\}$;
{*The shortest path between v and t has been found and v is added to* T^t}
$T^t = T^t \cup \{v\}$;
{*Examine incoming links of v*}
for each $(u, v) \in In(v)$ **such that** $d_v^t + l_{uv} < d_u^t$ **do**
$d_u^t = d_v^t + l_{uv}$;
$p_v^t = v$;
if $u \notin Q^t$ **then** $Q^t = Q^t \cup \{u\}$;
end
{*Stopping criterion*}
$\beta = \min\{d_i^s + d_i^t : i \in T^s \cup T^t\}$;
$J = \{i \in T^s \cup T^t : d_i^s + d_i^t = \beta\}$;
endwhile;

end.

Fig. 4. The two-tree PFS algorithm.

4.2.2. The two-tree Dijkstra/PFS algorithm

No modification is made to the two-tree Dijkstra algorithm because it can find the shortest path only between two nodes. The two-tree Dijkstra algorithm works by searching from both ends of the shortest path simultaneously [18]. A shortest-path tree rooted at the source node s and a shortest-path tree rooted at another source node t grow in alternate steps. The two trees are analogous except that the tree rooted at s expands a node by examining its outgoing links, and the tree rooted at t expands a node by examining its incoming links. A shortest path is found when both trees have a common node, say r, such that $d_r^s + d_r^t$ is a minimum, where d_r^s is the distance between r and s, and d_r^t is the distance between r and t, respectively. We define β as the minimum distance and J as the set of nodes that can be used to identify the shortest path. The following two-tree Dijkstra algorithm is provided in Ref. [18]. Assuming a priority queue is used for the candidate set Q, we call this algorithm two-tree PFS (Fig. 4).

Because the two-tree PFS algorithm computes the shortest path only between two nodes, it must be used $k(k-1)/2$ times to identify the shortest paths for all possible node pairs in K. As a result, the overall time complexity is $O(k^2(n+m)\log n)$.

We did not use Floyd's [13] or Dantzig's [10] all-pair shortest-path algorithms, which compute the shortest path for every pair of nodes in a graph. These algorithms require a substantial execution time of $O(n^3)$ [12]. However, the execution time of the two proposed algorithms will not exceed $O(k^2n^2)$, which is less than $O(n^3)$ as long as $k^2 < n$. In most situations where k is rather small compared with n, these two proposed algorithms will work faster than all-pair shortest-path algorithms.

5. System evaluation

We conducted a user evaluation and a simulation experiment in order to assess the performance of the proposed shortest-path algorithms. The user evaluation was aimed at addressing the effectiveness issue, namely, whether association paths identified by the shortest-path algorithms are more likely to generate investigative leads than those identified by the modified BFS algorithm, which is representative of the typical association search approach. The purpose of the simulation experiment, on the other hand, was to determine which shortest-path algorithm was more efficient for what type of networks. Crime investigators often encounter the efficiency issue when they work on a large network [15]. In this section, we first briefly describe our network construction process and then present the evaluation results.

5.1. Network construction

5.1.1. COPLINK concept space and AZNP

The criminal networks used in our experiment were constructed based on the same concept space approach [6] used in COPLINK Detect [17]. In such networks, the strength of an association is indicated by a co-occurrence weight. As reviewed previously, the nodes in COPLINK Detect are structured database records of entities. COPLINK Detect allows for link analysis with depth 1, that is, only nodes directly associated with source nodes can be found.

Rather than using structured database records, our criminal networks were constructed from unstructured textual documents. This is because law enforcement agencies often rely on crime report narratives to obtain detailed criminal association information that may not otherwise be available in structured data. We used our automated noun-phrasing tool called AZNP to extract noun phrases from texts based on part-of-speech tagging and noun phrasing rules [25]. The extracted noun phrases included various entity types such as persons, locations, vehicles, and properties. Co-occurrence weights between these entities were calculated to generate association strength measures.

5.1.2. Data set

The Phoenix Police Department provided us with one-year's worth of crime reports. The size of the dataset is 1 GB. These reports described various types of crimes ranging from shop-lifting to auto theft, from credit card fraud to narcotics possession and sales. We selected two samples as our test bed, namely, kidnapping and narcotics, both of which are organized crimes. The size of the kidnapping report collection is 4.5 MB, and the size of the narcotics report collection is 38 MB.

The crime reports varied substantially in length. For example, in the kidnapping sample, some documents simply contained a few lines about a phoned-in kid-

napping report, while others had hundreds of lines detailing a kidnapping investigation. Since the length of a document can affect the co-occurrence weights of the concepts it contains [6], we removed from our data sets those reports containing fewer than five lines of text. The noun phrases were extracted from the resulting document collections and irrelevant terms were filtered out based on a 3400-item stop word list. The noun phrases left after filtering were used as network nodes and their co-occurrence weights were calculated. Two networks were constructed: one for the kidnapping sample and the other for the narcotics sample. Table 1 presents the statistics for the two samples.

Table 1
Sample statistics of two networks

	Number of reports	Number of noun phrases extracted	Network size (n)	Number of links (m)	Average number of links a node has
Kidnapping	271	95,328	280	25,862	92.4
Narcotics	3572	861,516	4257	733,572	172.3

5.2. Results and discussions

5.2.1. User evaluation: effectiveness issue

In the user evaluation, we compared the effectiveness of the association paths identified by the shortest-path algorithms and those identified by the modified BFS algorithm. The purpose of the evaluation was to ascertain whether the shortest-path algorithms would be more useful for uncovering crime investigative leads.

The paths identified by an algorithm may consist of links that are not useful for crime investigations. With the concept space approach, a link between two entities is created if they co-occur in crime reports. However, a co-occurring association may not necessarily mean an important relationship between entities. For example, the shortest path algorithms identified three association paths for a kidnapping case with three source nodes: Juan (person), Jose (person), and West Van Buren (location):[1]

(1) Juan–Jose
(2) Juan–Maria–West Van Buren
(3) Jose–Maria–West Van Buren

Path (1) is useful because both Juan and Jose are listed in a report as victims in a kidnapping crime. Path (2) is considered to be non-useful. Two reports describe the association between Juan and Maria: one records that Juan Balderaz's ex-wife was Maria Palma; the other indicates that Juan Rodriguez kidnapped Maria Molina's daughter. The association between Maria and West Van Buren is recorded in another report, which indicates that Maria Dillon lived at 3100 West Van Buren. Notice that the three Maria's are different persons. Thus, the association path with Maria as the intermediate node cannot provide information about how Juan and West Van Buren are related. Path (3) is a useful path because one report indicates that Jose Carrasco's friend was Maria Dillon, who lived at 3100 West Van Buren.

To measure the effectiveness of our algorithms, we used a precision rate defined as follows:

$$\text{precision} = \frac{\text{number of useful paths selected by experts}}{\text{total number of paths identified by the algorithm}} \times 100\%$$

Because the modified BFS algorithm did not guarantee to identify the strongest association paths between entities, we predicted that the shortest-path algorithms could achieve a higher precision than the modified BFS algorithm.

We randomly selected 30 pairs of source nodes from each of the kidnapping network and the narcotics network. Association paths were computed using both a shortest-path algorithm and the modified BFS algorithm. As shown in Table 2, the paths found by the modified BFS algorithm generally contain more intermediate links than a shortest-path algorithm. Which shortest-path algorithm was used is not important here because they always generate the same paths.

A domain expert from the Tucson Police Department evaluated the resulting association paths. The expert had been serving in law enforcement for more than 30 years and had a substantial amount of expe-

[1] All entity names are scrubbed to ensure data confidentiality.

Table 2
Effectiveness evaluation results

Algorithm	Average number of links in association paths		Precision	
	Kidnapping	Narcotics	Kidnapping	Narcotics
Shortest-path algorithms	1.40	2.06	66.7%	71.4%
Modified BFS	1.73	12.50	30.0%	16.7%

rience in link analysis. For the results produced by an algorithm, he examined the 30 paths from each network by reading the original crime reports. He determined whether an association path was useful for generating investigative leads based on his past experience investigating similar crimes. It took 2.5–3 h to complete the evaluation task for each network. The results show that on average the shortest-path algorithms identified more useful association paths than the modified BFS algorithm. Around 70% of the paths found by the shortest-path algorithms were considered useful for both networks. For modified BFS, in contrast, only 30% of the paths from the kidnapping network and 16.7% of the paths from the narcotics network were considered to be useful. Table 2 shows the precision rate of each algorithm.

The shortest-path algorithms can achieve a higher precision because they always select associations with high co-occurrence weights during link exploration. As discussed previously, a co-occurrence weight is a measure of how frequently two entities are related. Therefore, the more frequently two entities are associated, the less likely they are to be related by chance, and the more likely such an association will be useful for investigations. In contrast, the modified BFS algorithm produces arbitrary paths between entities. It is very likely that these paths contain unimportant associations, resulting in a low precision rate.

Although promising, the shortest-path algorithms still failed to identify useful paths about 30% of the time. Based on our analysis of the non-useful paths found by the shortest path algorithms, we categorized the reasons for the failures as follows (using the kidnapping network as an example):

- *Some nodes in the networks do not represent unique entities*. This situation often occurs for the person type. Usually, after a person's full name is provided at the beginning of a crime report narrative, he/she is referred to only by the first name in later parts of the report. During network construction, the same first names extracted by the noun phraser from different reports are indiscriminately treated as one single node. As a result, a node (e.g., Maria) may not refer to a unique person but to different people with the same first name (e.g., Maria Palma, Maria Molina, Maria Dillon, etc.). This problem also exists for other types of entities such as vehicles, locations, and properties. For example, "white car" may refer to different white cars owned by different persons; "North 7th Street" includes a number of addresses on that particular street. A non-useful association path may result if it contains such intermediate nodes. In our test bed, 54.2% of the non-useful association paths fell into this category.
- *Whether an entity is relevant or not depends on specific contexts*. This problem seldom affects entities such as persons and addresses, because their presence in a crime report usually implies that they are relevant to that particular crime. Indeed, any person mentioned in a report has a role descriptor. For example, "sp" means suspect, "v" means victim, and "w" means witness. However, property entities may include any physical object that a person possesses. It is much more difficult to determine whether or not a property is relevant to a particular crime without considering the specific context of a crime. When a property is the target of a crime, it usually is considered to be relevant. However, if a physical object is mentioned simply to describe the environment or a situation, it is often treated as irrelevant. For example, a "cell phone" is a relevant property if it is stolen in a crime; it is irrelevant if a witness used his or her cell phone to report a crime to the police. Unlike a human, who can determine an entity's relevance based on contextual clues, the noun phraser cannot examine texts semantically to distinguish between relevant and irrelevant entities. As a result, an association path will be non-useful if it happens to include an irrelevant entity. Over 37% of the non-useful paths had this problem.
- *Two entities may have a fake relationship even though they are listed in the same report*. A link is established when two entities appear together in the

same document. However, this link may be a trivial association between the two entities. Usually, associations between a person and other entities (e.g., another person, vehicles, addresses, etc.) are less frequently subject to this problem. However, associations between entities other than persons are often less informative. For example, a link exists between "white Toyota" and "North 7th Street" because they are listed in the same report narrative. In this report, we found that a male driving a white Toyota car kidnapped the daughter of a person, who lived on North 7th Street. Such a link does not imply a useful relationship between these two entities but a "fake" one. Around 5% of the non-useful paths fell into this category.

Result of this analysis suggests that the effectiveness of our algorithms may be improved if more appropriate entities and associations are extracted and used.

5.2.2. Simulation experiment: efficiency issue

Our simulation experiment focused on the efficiency of the two shortest-path algorithms (modified PFS and two-tree PFS). We define the efficiency of an algorithm as its average execution time. The experiment was intended to ascertain which algorithm is more efficient for what type of networks in terms of network size and other structural characteristics.

To compare the efficiency of these two algorithms in the case of multiple source nodes, we varied the number of source nodes, k, from 2 to 5 in the simulations. We chose these numbers based on the observation from our pilot studies in which investigators usually used less than five source entities during an association search. Given a specific k, we randomly generated 100 cases using both algorithms for each network. The execution time for the algorithms was recorded and is presented in Table 3.

For all four values of k, the pairwise t-tests for the mean execution time suggest that two-tree PFS is significantly faster than PFS ($p < 0.001$) in the kidnapping network. However, PFS is significantly faster than the two-tree PFS algorithm ($p < 0.01$) in the narcotics network. Fig. 5 presents the execution time plot with $k = 5$ for the kidnapping and narcotics networks, respectively.

The result from the kidnapping network is consistent with the findings in Ref. [18]. According to Helgason et al. [18], a two-tree algorithm usually is faster than one-tree algorithms. In their study, a shortest-path tree in a one-tree algorithm contains about 50% of the nodes in a network before the shortest path is found; whereas a two-tree algorithm can find the shortest path when its trees contain only 6% of the nodes. We found similar results in terms of the number of nodes contained in the shortest-path trees. For the kidnapping network, the one-tree PFS algorithm generated a tree containing 52% of the nodes, and the two-tree PFS algorithm generated two trees containing 14.7% of the nodes in total. For the narcotics network, the tree in the one-tree PFS algorithm contained 49.6% of the nodes, and the trees in the two-tree PFS algorithm only contained 3.9% of the nodes.

However, the one-tree algorithm outperformed its two-tree counterpart in the narcotics network. Based on our analysis of the structural characteristics of both networks, we found that two factors might have caused this discrepancy.

- *Network size*. As the size of a network increases, the size of the candidate set Q, which contains

Table 3
Mean execution time (in seconds) for the two shortest-path algorithms (Numbers in parentheses are standard deviations)

(a) Results for the kidnapping network				
Algorithm	$k=2$	$k=3$	$k=4$	$k=5$
Modified PFS	1.00 (0.54)	2.89 (0.97)	6.00 (1.26)	10.67 (2.09)
Two-tree PFS	0.35 (0.19)	0.95 (0.28)	1.94 (0.37)	3.45 (0.65)
(b) Results for the narcotics network				
Modified PFS	66.75 (27.06)	194.05 (53.97)	419.47 (61.91)	661.10 (132.22)
Two-tree PFS	239.00 (132.00)	709.50 (263.75)	1350.56 (348.70)	2,322.28 (546.25)

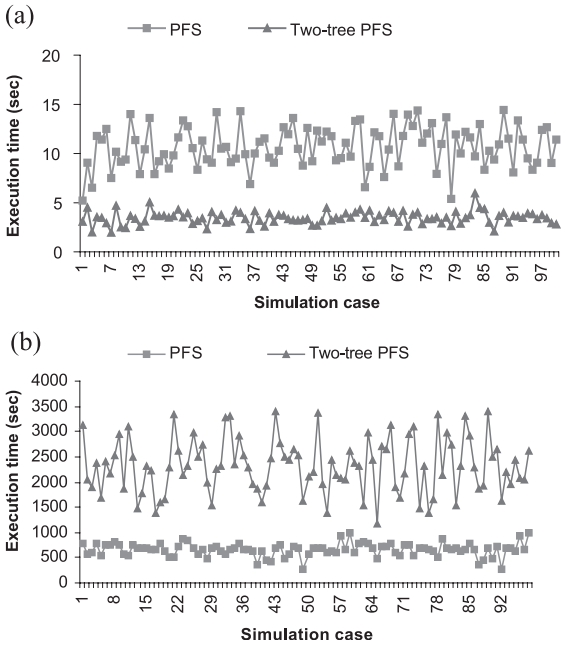

Fig. 5. Execution time scatter plot ($k=5$). (a) Results for the kidnapping network. (b) Results for the narcotics network.

temporarily labeled nodes, also increases. It takes time to search and update the labels in Q when incident links of a node are explored. Therefore, when a network is large and the computational cost of processing the candidate sets becomes high, the two-tree algorithm will be inefficient. For the narcotics network ($n=4257$), the two candidate sets in the two-tree PFS algorithm together contained 120% of the total nodes, whereas the candidate set in the one-tree PFS algorithm contained only 47.9% of the total nodes. Thus, the time for processing the candidate sets in the two-tree PFS algorithm was much longer than the time spent in the one-tree PFS algorithm, causing the two-tree PFS algorithm to be slower.

- *Network density.* The density of a network is defined as the ratio of the total number of links to the possible number of links [27]. Thus, the density of an undirected network consisting of n nodes and m links is $2m/n(n-1)$. Network density may have an impact on the efficiency of a two-tree algorithm, which can find a shortest path only if the two trees have overlapping nodes. The lower the density of a network, the less likely two trees will overlap. In our experiment, the density of the narcotics network is 0.08. This means that the two trees have overlapping nodes only 8% of the time and that the algorithm must spend more time growing the trees. The kidnapping network, in contrast, has a much higher density (0.66), causing the two-tree algorithm to be faster than the one-tree algorithm.

Based on the analysis, we suggest that the two-tree PFS algorithm be used for small and dense networks. For large and sparse networks, the one-tree PFS algorithm is faster.

6. Conclusions

Effective and efficient link analysis techniques can assist investigation of organized crimes. With the help of such techniques, crime investigators may acquire better understanding of the interrelationships between offenders, thereby discovering new leads for investigation.

In this paper, we proposed a link analysis technique that employs shortest-path algorithms (PFS and two-tree PFS) to identify the strongest associations between two or more entities in a criminal network. Modifications were made to the algorithms to solve the shortest-path computation problem for multiple source nodes. After a logarithmic transformation of the link weights, these shortest paths could identify the strongest associations between given entities.

Our evaluation study focused on the approach's effectiveness and efficiency, both of which are desirable features of a sophisticated decision-support system. The results show that the shortest-path algorithms outperformed the typical association search approach (as represented by the modified BFS algorithm) of crime investigators in terms of effectiveness. The association paths identified using the shortest-path algorithms were considered as useful about 70% of the time, as opposed to precision rates of 30% (for the kidnapping network) and 16.7% (for the narcotics network) with the modified BFS algorithm. The two shortest-path algorithms always produced identical

results but the two-tree PFS algorithm was faster for the small and dense kidnapping network and the PFS algorithm was faster for the large and sparse narcotics network.

Analysis of the evaluation results suggests that the effectiveness might be improved by extracting more appropriate entities from texts and using them as network nodes. In our future research, we will apply effective named-entity extraction techniques to replace our current noun phraser. We will also incorporate some domain-specific heuristics to help the system select only entities and associations that are considered useful by crime investigators.

Acknowledgements

This project has primarily been funded by the National Science Foundation, Digital Government Program, ''COPLINK Center: Information and Knowledge Management for Law Enforcement,'' #9983304, July 2000–June 2003. We would like to thank the following people for their support and assistance during the entire project development and evaluation process: Dr. Homa Atabakhsh, Michael Chau, JoAnna Davis, and other members of the University of Arizona Artificial Intelligence Lab, Detective Tim Petersen, and other personnel from the Tucson Police Department and the Phoenix Police Department.

References

[1] M. Ali, F. Kamoun, Neural networks for shortest path computation and routing in computer networks, IEEE Transactions on Neural Networks 4 (5) (1993) 941–953.
[2] T. Anderson, L. Arbetter, A. Benawides, A. Longmore-Etheridge, Security works, Security Management 38 (17) (1994) 17–20.
[3] F. Araujo, B. Ribeiro, L. Rodrigues, A neural network for shortest path computation, IEEE Transactions on Neural Networks 12 (5) (2001) 1067–1073.
[4] T. Asano, D. Kirkpatrick, C. Yap, Pseudo approximation algorithms, with applications to optimal motion planning, Proceedings of the Eighteenth Annual Symposium on Computational Geometry (Barcelona, Spain), 2002, pp. 170–178.
[5] M. Chau, J. Xu, H. Chen, Extracting meaningful entities from police narrative reports, Proceedings of the National Conference on Digital Government Research (Los Angeles, CA), 2002, pp. 271–275.
[6] H. Chen, K.J. Lynch, Automatic construction of networks of concepts characterizing document databases, IEEE Transactions on Systems, Man and Cybernetics 22 (5) (1992) 885–902.
[7] N.A. Chinchor, Overview of MUC-7/MET-2, Proceedings of the Seventh Message Understanding Conference (MUC-7), 1998.
[8] W.F. Coady, Automated link analysis: artificial intelligence-based tool for investigators, Police Chief 52 (9) (1985) 22–23.
[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1991.
[10] G. Dantzig, On the shortest route through a network, Management Science 6 (1960) 187–190.
[11] E. Dijkstra, A note on two problems in connection with graphs, Numerische Mathematik 1 (1959) 269–271.
[12] J. Evans, E. Minieka, Optimization Algorithms for Networks and Graphs, Marcel Dekker, New York, 1992.
[13] R.W. Floyd, Algorithm 97: shortest path, Communications of the ACM 5 (6) (1962) 345–370.
[14] H.G. Goldberg, T.E. Senator, Restructuring databases for knowledge discovery by consolidation and link formation, Proceedings of 1998 AAAI Fall Symposium on Artificial Intelligence and Link Analysis, AAAI Press, Menlo Park, CA, 1998.
[15] H.G. Goldberg, R.W.H. Wong, Restructuring transactional data for link analysis in the FinCen AI System, Proceedings of 1998 AAAI Fall Symposium on Artificial Intelligence and Link Analysis, AAAI Press, Menlo Park, CA, 1998.
[16] W.R. Harper, D.H. Harris, The application of link analysis to police intelligence, Human Factors 17 (2) (1975) 157–164.
[17] R.V. Hauck, H. Atabakhsh, P. Ongvasith, H. Gupta, H. Chen, Using coplink to analyze criminal-justice data, IEEE Computer 35 (3) (2002) 30–37.
[18] R.V. Helgason, J.L. Kennington, B.D. Stewart, The one-to-one shortest-path problem: an empirical analysis with the two-tree dijkstra algorithm, Computational Optimization and Applications 1 (1993) 47–75.
[19] P. Klerks, The network paradigm applied to criminal organizations: theoretical nitpicking or a relevant doctrine for investigators? Recent developments in The Netherlands, Connections 24 (3) (2001) 53–65.
[20] V.E. Krebs, Mapping networks of terrorist cells, Connections 24 (3) (2001) 43–52.
[21] R. Lee, Automatic information extraction from documents: a tool for intelligence and law enforcement analysts, Proceedings of 1998 AAAI Fall Symposium on Artificial Intelligence and Link Analysis, AAAI Press, Menlo Park, CA, 1998.
[22] D. McAndrew, The structural analysis of criminal networks, in: D. Canter, L. Alison (Eds.), The Social Psychology of Crime: Groups, Teams, and Networks, Offender Profiling Series, Aldershot, Dartmouth, vol. III, 1999.
[23] C.E. Perkins, P. Bhagwat, Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers, Proceedings of SIGCOMM Symposium on

Communications Architectures and Protocols, London, UK, 1994, pp. 212–225.
[24] M.K. Sparrow, The application of network analysis to criminal intelligence: an assessment of the prospects, Social Networks 13 (1991) 251–274.
[25] K.M. Tolle, H. Chen, Comparing noun phrasing techniques for use with Medical Digital Library tools, Journal of the American Society for Information Science 51 (4) (2000) 352–370.
[26] Z. Wang, J. Crowcroft, Analysis of shortest-path routing algorithms in a dynamic network environment, ACM Computer Communication Review 22 (2) (1992) 63–71.
[27] S. Wasserman, K. Faust, Social Network Analysis: Methods and Applications, Cambridge Univ. Press, Cambridge, 1994.

Jennifer J. Xu is a doctoral candidate in Management Information Systems (MIS) at the University of Arizona, where she is a member of the Artificial Intelligence Lab. Her research interests include knowledge management, social network analysis, computer-mediated communication, and information visualization. She received her MS in Computer Science and MA in Economics from the University of Mississippi in 1999 and 2000, respectively, and her BS in Engineering from Beijing Institute of Technology, China, in 1995.

Dr. Hsinchun Chen is McClelland Professor of Management Information Systems and head of Artificial Intelligence Lab at the Management Information Systems (MIS), Department at the University of Arizona. He received his PhD degree in Information Systems from New York University in 1989. He is author of more than 70 articles about semantic retrieval, search algorithms, knowledge discovery, and collaborative computing in leading information technology publications. He serves on the editorial board of *Journal of the American Society for Information Science* and *Decision Support Systems*. He is an expert in digital library and knowledge management research whose work has been featured in various scientific and information technologies publications including *Science*, *The New York Times*, *NCSA Access Magazine*, *WEBster*, and *HPCWire*. Since 1990, Dr. Chen has received more than US$10 M in research funding from various government agencies and major corporations including NSF, DARPA, NASA, NIH, NIJ, NLM, NCI, NCSA, HP, SAP, 3COM, and AT&T. Dr. Chen founded The University of Arizona Artificial Intelligence Lab in 1990, which employs a staff of more than 40 full-time research scientists, research assistants, and programmers. Dr. Chen also was the founding director of The University of Arizona's Mark and Susan Hoffman eCommerce Laboratory (October 2000). That laboratory has received more than US$2 M in infrastructure funding from Mark Hoffman (founder of Commerce One), from HP, and more than US$10 M in software donations from major IT software companies including SAP, Oracle, Microsoft, and IBM.