Filling the Missing Names of Towns in a Map: A Graph Theoretic Approach

Frank Harary*

Department of Computer Science
New Mexico State University

Aurora Morgana

Dipartimento di Matematica
Università di Roma "La Sapienza"

Bruno Simeone

Dipartimento di Statistica
Probabilità e Statistiche Applicate
Università di Roma "La Sapienza"

ABSTRACT. A map shows only the names of some (but not all) towns in a region. If for each town, the names of all neighboring towns are known, when is it possible to reconstruct from this information the missing names? We deal with a generalization of this problem to arbitrary graphs. For a graph $G = (V, E)$ with n nodes, we give an $O(n^3)$ algorithm to recognize those instances for which the answer is YES, as well as two characterization theorems: NO-instances are determined by the existence of a certain partition of V and YES-instances by the existence of a suitable weak order in V.

1 The Problem

A region consists of n townships. For each township the set of all neighboring townships is known. A map of the region is available, in which only the names of some – but not all – townships are indicated. Making use of the given list of neighborhoods, is it possible to fill the missing names?

*Research supported by a CNR visiting professorship at the University of Rome "La Sapienza" during the month of May 1992.

One of us actually encountered this problem when trying to visualize on a map the output of an algorithm for political redistricting. The map showed only the names of the townships with at least 30,000 inhabitants. On the other hand, the adjacency lists of all townships were available in a computer file.

To illustrate, consider the map of Figure 1, where only the names of the townships 2, 3, 4, 11 and 12 are known to be A, B, E, D and C respectively.

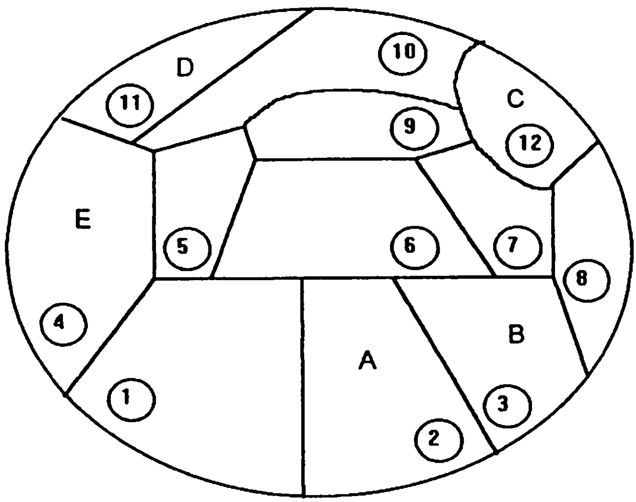

Figure 1

Name	Adjacency List
A	B F H
B	A H L M
C	J K L M
D	E K
E	D F G H
F	A E G H
G	E F H J K
H	A B F G J M
J	C G H K M
K	C D E G J
L	B C M
M	B C H J L

Table 1

We suppose that the adjacency list of each township is available as shown in Table 1. Then it is possible to reconstruct the missing names in the following way:

Township 6 is adjacent to A and B. An inspection of Table 1 shows that the only township which is adjacent to both A and B is H. Hence the name of township 6 is H.

Township 7 is adjacent to B and C and H. From Table I, the only township neighboring with B, C and H is M. Hence 7 must have the name M.

Continuing, one can sequentially identify the remaining names as follows:

$$L \to 8;\ J \to 9;\ F \to 1;\ G \to 5;\ K \to 10.$$

On the other hand, in the map of Figure 2 the townships 1 and 4 cannot be identified as there is not enough information to decide which is A and which is D, but if we had been given only the names of townships 1 and 2, then 3 and 4 could have been identified.

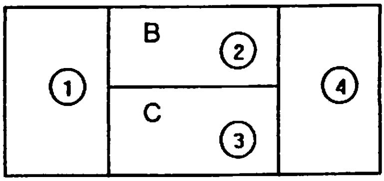

Figure 2

Name	Adjacency List
A	B C
B	A C D
C	A B D
D	B C

Table 2

In order to give a graph theoretic formulation of the problem, let us introduce some notation and definitions, following [1] in general. Let $G = (V, E)$ be a graph with $n = |V|$ nodes and $m = |E|$ edges. For $x \in V$, we define its *neighborhood* to be $N(x) = \{z \in V : xz \in E\}$. Given $x \in V$ and any subset $S \subset V$, we write $N(x, S) = \{z \in S : xz \in E\} = N(x) \cap S$.

A node $x \notin S$ is called *S-identifiable* if for each $y \notin S$, $y \neq x$, we have $N(y, S) \neq N(x, S)$. Let $I(S)$ be the set of all S-identifiable nodes of G.

Two nodes x, y are *S-twins* if $N(x, S) = N(y, S)$.

With reference to the example of Figure 1, let $S_0 = \{2, 3, 4, 11, 12\}$ be the set of townships whose set of corresponding names $\{A, B, E, D, C\}$ is initially known.

Then nodes 7 and 8 are S_0-twins since $N(7, S_0) = N(8, S_0) = \{3, 12\}$ and thus their names cannot be identified at once. On the other hand, nodes 1, 5, 6, 9 and 10 are S_0-identifiable, since $N(1, S_0) = \{2, 4\}$; $N(5, S_0) = \{4\}$; $N(6, S_0) = \{2, 3\}$; $N(9, S_0) = \{12\}$; $N(10, S_0) = \{11, 12\}$ and their names are uniquely determined to be F, G, H, J and K, by inspection of Table 1.

Problem: Given a graph G and $S_0 \subset V$ we want to know if there is a permutation $(x_1, \ldots, x_n)$ of V such that letting $q = |S_0|$ and $S_k = \{x_1, \ldots, x_k\}$, $k = 1, 2, \ldots, n$,

(i) $S_q = S_0$,

(ii) x_{k+1} is S_k-identifiable, $q \leq k < n$.

If the answer is YES, G will be said to be S_0-*identifiable* and the sequence $(x_1, \ldots, x_n)$ will be called an S_0-*identifying sequence.*

We next give a simple general algorithm for deciding whether or not a given subset of nodes enables all the remaining nodes to be identified and then we shall prove its correctness.

Algorithm 1.

Step 0. Set $S = S_0$ and $k = q + 1$. Let the elements of S_0 be $x_1, \ldots, x_q$.
Step 1. While $I(S) \neq \emptyset$ do
begin
Select $x \in I(S)$ and replace S by $S \cup \{x\}$. Set $x_k = x$.
Increase k by 1.
endwhile
Step 2. If $S = V$ then G is S_0-identifiable; else G is not S_0-identifiable.
End.

Clearly if the algorithm answers YES then G is S_0-identifiable and the sequence $(x_1, \ldots, x_i)$ produced by the algorithm is an S_0-identifying one. However it is not obvious that if the answer of the algorithm is NO there is no identifying sequence at all.

To prove this, we start from the following trivial remark:

Lemma 0. *Let $S \subset T \subset V$ and $x \notin T$. If x is S-identifiable then x is also T-identifiable.*

Theorem 1. *Algorithm 1 is correct.*

Proof: As remarked above, it is enough to prove that when the set S output by the algorithm is properly contained in V, then G is not S_0-identifiable. So suppose that $(x_1, \ldots, x_n)$ is an S_0-identifying sequence. Let k be the smallest index such that $x_{k+1} \notin S$. Then $S_k \subset S$. On the other hand, x_{k+1} is S_k-identifiable and hence by Lemma 0 is also S-identifiable, contradicting the fact that the set S obtained by the algorithm satisfies $I(S) = \emptyset$. □

By this theorem we note that the algorithm works regardless of the choice of $x \in I(S)$ in Step 1. Hence one can consider the following special version.

Algorithm 2.

Step 0. Set $S = S_0$.
Step 1. While $I(S) \neq \emptyset$ do replace S by $S \cup I(S)$.
Step 2. If $S = V$ then G is S_0-identifiable else G is not S_0-identifiable.
End.

Notice that, if $I(S) \neq \emptyset$ and $S \subset T \subset S \cup I(S)$, then $I(T) \neq \emptyset$ by Lemma 0. Hence in Step 1 there is no need to test the condition $I(T) \neq \emptyset$ for all such intermediate subsets T.

Algorithm 2 runs in polynomial time. In fact, the implementation given below is seen to run in $O(n^3)$ time. The input graph is given by means of adjacency lists while, for the representation of sets, one can make use of efficient data structures supporting Union-Find operations [2].

Algorithm 2 (implementation).

Step 0. Set $S = Q = S_0$.
for each $i, j = 1, \ldots, n$ do set $a_{ij} = 1$.
{Since the end of the first execution of Step 1, for all $i, j \notin S$ one has $a_{ij} = 1$ or 0 according as i and j are S-twins or not w.r.t. the current S}
Step 1. While $Q \neq \emptyset$ do

```
begin
  for each h ∈ Q do
    for each {i,j} V − S do
       if i ∈ N(h) and j ∉ N(h) or i ∉ N(h) and j ∈ N(h)
       then set a_ij = 0.
    endfor
  endfor
  Set Q = ∅.
  for each i ∉ S do
    let d_i = Σ{a_ij : j ∉ S}.
    if d_i = 1 then add i to Q. {d_i = 1 iff i ∈ I(S)}
  endfor
  Replace S by S ∪ Q.
endwhile
```

Step 2. If $S = V$ then G is S_0-identifiable else G is not S_0-identifiable.
End.

We present two characterizations of an S_0-identifiable graph G.

Let $G = (V, E)$ and let $X, Y \subset V$ be disjoint nonempty subsets of nodes. Then we write $B(X, Y)$ for the bipartite subgraph of G spanned by those edges of G joining a node of X with a node of Y.

Theorem 2. *Graph G is not S_0-identifiable if and only if there is a partition $\{X_0, X_1, \ldots, X_t\}$ of V such that*

(i) $X_0 \supset S_0$;

(ii) $|X_i| \geq 2$, $i = 1, \ldots, t$;

(iii) *for every $i = 1, \ldots, t$ the bipartite subgraph $B(X_i, X_0)$ is either empty or complete.*

Proof: To demonstrate the 'only if' assertion, let

$$V_0 = S_0 \text{ and } V_{k+1} = V_k \cup l(V_k), \; k = 0, 1, \ldots \tag{1}$$

If G is not S_0-identifiable then there is an index k such that $I(Vk) = \emptyset$.

Define in $V - V_k$ an equivalence relation by $x \approx y$ meaning that x and y are V_k-twins.

Let $X_0 = V_k$ and let $X_1, \ldots, X_t$ be the equivalence classes of $\approx$. Then $\{X_0, X_1, \ldots, X_t\}$ is a partition of V. Moreover, $|X_i| \geq 2$ for all $t \geq 1$, else $I(V_k) = \emptyset$. Finally $B(X_i, X_0)$ is either empty or complete.

We now show the "if part". Let $\{X_0, X_1, \ldots, X_t\}$ be a partition satisfying (i), (ii), (iii) and assume that G is S_0-identifiable. Let $(x_1, \ldots, x_n)$ be an S_0-identifying sequence and let k be the smallest index such that $x_{k+1} \notin X_0$. Then $S_k = \{x_1, \ldots, x_k\} \subset X_0$ and since x_{k+1} is S_k-identifiable it is also X_0-identifiable by Lemma 0. But then x_{k+1} has no X_0-twins, contradicting (ii) and (iii). □

Recall that a binary relation $\geq$ on a set V is *complete* if for all $x, y \in V$, $x \geq y$ or $y \geq x$.

A *weak order* on a set V is a complete and transitive binary relation $\geq$. We use the standard notation:

$$x > y \text{ means } x \geq y \text{ and } x \neq y.$$

$x > -y$ or in words, x covers y, means $x > y$ and there is no z such that $x > z > y$.

Theorem 3. *Graph G is S_0-identifiable if and only if there is a weak order on V such that for all $x, y \notin S_0$ there is a node $z < x$, y which is adjacent to exactly one of x and y.*

Proof: We first prove "only if". Let G be S_0-identifiable. If $V_0, V_1, \ldots$ are defined by the recursion (1), then in view of Algorithm 2 there is a smallest index p such that $V_p = V$.
Define $L_0 = S_0$ and $L_k = I(L_0 \cup L_1 \cup \cdots \cup L_{k-1})$ for $k = 1, \ldots, p$.
Notice that $V_k = L_0 \cup L_1 \cup \cdots \cup L_{k-1}$ for $k = 0, 1, \ldots, p$.
Hence $\{L_0, L_1, \ldots, L_p\}$ is a partition of V. Define $\geq$ in V by

$$y \geq x \text{ means } x \in L_h \text{ and } y \in L_k \text{ and } h \leq k.$$

Clearly $\geq$ is a weak order.
Now let $x, y \notin S_0$. Assume that $x \in L_h$ and $y \in L_k$, where $h, k \geq 1$ and, without loss of generality, $h \leq k$. Then x is V_{h-1}-identifiable and thus there exists a node z which is adjacent to exactly one of x and y, and belongs to some L_i, $i < h \leq k$, implying $z < x, y$.

To show the "if part", let $\geq$ be a weak order in V satisfying the hypothesis of the theorem.

Define $L_0 = S_0$ and $L_k = \{x\colon x \in V - (L_0 \cup L_1 \cup \cdots \cup L_{k-1})$ and $\exists$ $y \in L_0 \cup L_1 \cup \cdots \cup L_{k-1}$, such that $x > -y\}$ for each $k = 1, 2, \ldots$.

Clearly, there exists an index p such that $V = L_0 \cup L_1 \cup \cdots \cup L_p$.

Moreover $\{L_0 \cup L_1 \cup \cdots \cup L_p\}$ is a partition of V. Let $h \geq 1$ and let x be an arbitrary element of L_h. If $k \geq h$ and y is an arbitrary element of L_k, then there exists a $z < x$, y which is adjacent to exactly one of x, y. Since z must belong to some L_i, with $i < h$, it follows that x is $(L_0 \cup L_1 \cup \cdots \cup L_{h-1})$-identifiable. Hence L_h $I(L_0 \cup L_1 \cup \cdots \cup L_{h-1})$. Since $\{L_0 \cup L_1 \cup \cdots \cup L_p\}$ is a partition of V, the theorem follows. □

Notice that combining Theorems 2 and 3 one gets a good characterization of those pairs (G, S_0) such that G is S_0-identifiable.

References

[1] F. Harary, *Graph Theory*, Addison-Wesley, Reading, 1969.

[2] T. Cormen, C. Leiserson and R. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA,1990.