A Proposal for an Algorithm to Generate Consistent STNs

Merrick Chang

June 26, 2020

I have a proposal for an algorithm for the random generation of consistent STNs. I *think* that this algorithm is truly random in the sense that it is able to generate any consistent STN. I'll go through the necesary proofs in the Preliminaries section and propose a sketch of the algorithm in the Proposal section. I haven't sketched out the psuedo-code specifically, but it ought to work and work quite quickly.

1 Preliminaries

We start with the following definition:

Definition 1.1. We say that a digraph $G = (V, E)$ is a **diforest** if it is accylic. We call diforest a **ditree** if additionally there exists some (unique) vertex x such that for all $v \in V$ such that $v \neq x$. We call this vertex the **source**.

Remark. That x is unique is an obvious consequence of its property. If there exists a second vertex y with this property, then there exists a path from x to y and from y to x and the union of those paths is thusly a cycle.

Much like forests can be described as a number of trees for undirected graphs, diforrests can be described as compositions of ditrees as well, albeit less neatly.

Theorem 1.1. *Every diforest, $G = (V, E)$, can be written as the union of edge-wise dijoint (but not necessarily vertex-wise disjoint) of ditrees.*

Proof. The proof proceeds by induction. Suppose $|V| = 1$. Then obviously the statement is trivially true. Suppose that the statement is true for all $|V| \leq n$. Consider the case where $|V| = n + 1$. Then, choose some v in V. The graph $H = (V - \{v\}, E)$ is acyclic, so it is a diforrest and thus the union of ditrees $T_1, T_2, \ldots T_m$. Now, for each inedge, e, from some vertex $u \in V$ to v, abitrarily choose ditree containing u and add v and e to the ditree. Then likewise, create a new ditree with v in it and for outedge, e, from v to $w \in V$ add e and w to this new ditree. Call the resulting trees $T'_1, T'_2, \ldots T'_{m+1}$. G is a union of these ditrees. Thus the statement is true for $|V| = n + 1$. □

Remark. Obviously the converse statement, every edgewise dijsoint union of ditrees is a diforest is untrue and has trivial counterexamples. There is certainly a more precise statement for which the converse is true but it is unimportant for the purposes of these proofs.

Definition 1.2. Let $G = (V, E)$ be a digraph. We say that a function $f : V \to \mathbb{N}$ is an **ordering** iff it is injective.

Theorem 1.2. *For every diforest $G = (V, E)$, there exists some ordering $f : V \to \mathbb{N}$ such that for each edge e_{uv} from u to v, $f(u) < f(v)$.*

Proof. The proof proceeds by induction. For the case, $|V| = 1$, this result is trivial. Let the statement of the theorem hold for $|V| \leq n$. Suppose $|V| = n + 1$. Let $v \in G$. Then $G - \{v\}$ is a graph with n vertices so there exists some ordering $g : V - \{v\} \to \mathbb{N}$ such that for all edges e_{ab} from a to b where $a, b \in V - \{v\}$, $g(a) < g(b)$. Then let

$$f(x) = \begin{cases} g(x) & g(x) < M \\ M & v = x \\ g(x) + 1 & g(x) > M \end{cases}$$

where $M = \max\{g(x) \in \mathbb{N} : \text{there exists some edge } e_{xv} \in E \text{ from } x \text{ to } v\}$. □

Corollary 1.2.1. For every ditree $G = (V, E)$, there exists some ordering $f : V \to \mathbb{N}$ such that for each edge e_{uv} from u to v, $f(u) < f(v)$.

2 Proposal

Now, each STN can be represented by weighted graph $G = (V, E, w)$. Now, by definition G has no negative cycles so G_- the unweighted digraph containing only edges in G with negative weights is a diforest. The premise is that we can generate a diforrest first which minimizes the amount of computation you need to do to ensure the STN is consistent. My suggested algorithm goes roughly as follows:

1. Generate a list of nodes
2. Generate an empty hashtable of source vertices and their respective distances to each other vertex
3. For each node u,
   (a) for each node v with index greater than u, create an edge from v to u with some specified probability; generate a random negative weight for this edge; if an edge is created this way and update the distance from each source node
   (b) if edges were just created and u has no predecessors, mark it as a source node; for each source node keep track of the minimum (in terms of absolute value) distance from the source node and each other node; if there is no path from the source node to another node, mark it as being infinite; mark the distance to itself as 0

4. For each node u and each other node v generate an edge from u to v with some specified probablity. If u is a child of v in some ditree, take the difference, δ between the distance from the source to v and the distance from the source to u; assign the edge a wieght of at least δ; else generate a random positive weight

By Theorem 1.2, any consistent STN should be able to be generated in this manner. Furthermore, this method saves the trouble of generating random STNs and checking if they are consistent or not.