OPTIMIZATION OF CONNECTION STRUCTURES IN GRAPHS

Edited by

A. Iwainsky

Central Institute of Cybernetics and Information Processes
Academy of Sciences
of the
German Democratic Republic

Peter Richter, Oleg Taraszow

Random Graph Generation

Abstract

In this paper two methods of graph generation are presented by which it is possible to produce random graphs embedded in a two- or threedimensional rectilinear grid. For the two-dimensional case we describe a graphical presentation which is suitable not only for the output on graphic displays but also on low-cost printers and alphanumeric displays. The presented methods have been the basis for the development of a graph generator written in FORTRAN 77.

1. Introduction

Many algorithms deal with solutions of graph-theoretical problems. If these algorithms are implemented, adequate graphs for performance analysis will be necessary. The reasons for the application of a graph generator are the following:

- Manual preparation, input, and checking of graphs is very time-consuming.
- Only the use of a large number of different graphs of a certain class makes it possible to judge the quality and power of graph algorithms.

The two random generation methods described here yield finite, connected, and weighted graphs embedded in two- or threedimensional rectilinear grids and containing neither self-loops nor multiedges. The user can sepcify the limitation of the graph extent by a window or cuboid.

Such graphs are especially useful as fictive models of layout possibilities in industrial sites, and, in fact, have been used during the research process reported in this volume. With respect to an expressive illustration of layout problems and their inherent difficulties the graphic representation of graph models is of importance. Such a representation can be easily generated on the basis of the methods described in this paper.

Another example for the generation of fictive models of practical routing possibilities is that described in /1/. But in contrast to our approach a whole twodimensional grid and some further edges between randomly chosen grid vertices are generated. In models of traffic networks these additional edges may stand for highways.

Finally we only mention /2/ as an example for the generation of random graphs which are not intended to be embedded in the two- or threedimensional Euclidean space.

2. Definitions, Notations and Symbolism

In this paper the following terminology will be used:

- $Q = (V,E)$:= any graph Q with vertex set V and edge set E
- $V(Q)$:= vertex set of graph Q
- $E(Q)$:= edge set of graph Q
- $P_Q(p,q)$:= any path in graph Q between vertices p and q
- $\mathbb{N}$:= set of natural numbers
- $(\mathbb{N}^k,d)$:= ordered pair to denote a metric space with the underlaying set $\mathbb{N}^k$, the distance function d, and the dimensionality $k = 2$ or 3
- $d(p,q)$:= $\sum_{i=1}^{k} |\xi_i - \eta_i|$ - rectilinear distance between two points $p = (\xi_1,\ldots,\xi_k) \in \mathbb{N}^k$ and

$q = (\eta_1,\ldots,\eta_k) \in \mathbb{N}^k$ with $k = 2$ or 3

- $\overline{G} = (\overline{V},\overline{E})$:= virtual grid within the space $(\mathbb{N}^k,d)$
- $\overline{V}$:= $\bigtimes_{i=1}^{k} K_i \subset \mathbb{N}^k$ ($\times$ denotes the Cartesian product); $K_i = \{1,2,\ldots,l_i\}$; l_i - input parameter
- $\overline{E}$:= $\{(p,q) : p,q \in \overline{V} \wedge d(p,q) = 1\} \subseteq \overline{V} \times \overline{V}$ (edge set $E(\overline{G})$)
- $G = (V,E)$:= actual generated subgraph $G \subseteq \overline{G}$
- $deg(i)$:= degree of vertex i with respect to G
- $deg^*(i)$:= degree of vertex i with respect to $\overline{G}$
- $\widehat{deg}(i)$:= $\min\{p_2, deg^*(i)\}$ (p_2 see below)
- $i \hat{\in} I$:= denotes the selection of an element i out of set I by a pseudo-random-number generator approximating a choice with the same probability for each element of I. In the following we often write shortly "randomly chosen".
- $N_Q(i)$:= set of neighbours of vertex i of graph Q
- n := $|V(G)|$
- m := $|E(G)|$

Furthermore we introduce the following parameters for the control of algorithms describes below

- p_1 := number of vertices in the generated graph
- p_2 := maximum vertex degree; p_1 and p_2 are valid only in the case of method B (s. below)
- p_3 := number of initial vertices in the case of method A
- $p_4 = \begin{cases} 2 & \text{- generation in the 2-dimensional grid } \overline{G} \\ 3 & \text{- generation in the 3-dimensional grid } \overline{G} \end{cases}$
- $p_5 = \begin{cases} 1 & \text{- generation of a tree} \\ 0 & \text{- generation of an arbitrary graph} \end{cases}$
- $p_6 = \begin{cases} 0 & \text{- no costs assigned to edges} \\ 1 & \text{- Weights are assigned to the edges of G according to a uniform distribution between } p_7 \text{ and } p_8 \\ 2 & \text{- Weights are assigned to the edges of G according to a normal distribution with mean value} \end{cases}$

equal to the distance d between adjacent vertices and with standard deviation p_9. Negative values are rejected.

- p_7, p_8, p_9 ⇒ see p_6
- p_{10} = { 0 - no costs assigned to vertices; 1 - Weights are assigned to vertices of G according to a uniform distribution between p_{11} and p_{12}.
- p_{11}, p_{12} ⇒ see p_{10}
- l_1 := length
- l_2 := width
- l_3 := height

} of the demanded output window or the output cuboid

3. Generation of an Undirected Graph G which is Embedded in the Grid $\bar{G}$

This generation can be carried out in several ways, among them the following two methods A and B:

Method A:
At first a set W containing p_3 different nodes $i \hat{\in} \bar{V}$ is generated. Then the vertices of W are connected by a tree. If $p_5 = 1$ the graph generation is terminated. Otherwise a further step is started, namely the extension of the tree by the generation of additional paths between some pairs of vertices $x,y \hat{\in} W$.

Method B:
By this method graph G is generated iteratively starting with one randomly selected vertex $v \hat{\in} \bar{V}$. In each iteration a randomly chosen edge incident to at least one of the vertices already belonging to G is added to this graph until the number of vertices in G is equal to p_1. That is carried out in such a way that the degrees of vertices in G do not exceed p_2.

3.1. Graph Generation by Method A

3.1.1. The Generation of a Tree G (Algorithm I)

Method A generates trees in the following steps:

Initialisation:
After the generation of vertex set W (s. above) one vertex of W is randomly selected and constitutes the tree G.

Extension Step:
In each extension step two vertices $y \hat{\in} V(G) \cap W$ and $x \hat{\in} W \setminus V(G)$ are selected. Then the generation of a shortest path from x to y is carried out iteratively (starting with x), until a vertex y^* already belonging to the tree is reached. In each iteration the optimal path is randomly extended by one element of $\bar{E}$. Tree G is extended by the path from x to y^*. This extension is repeated as long as not all vertices of W belong to V(G).

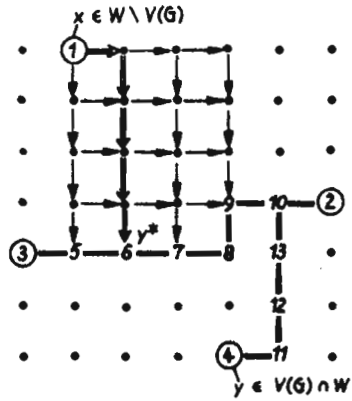
Termination:
If all vertices of W belong to V(G) the procedure is terminated.

The tree extension is illustrated in Fig. 1.
Fig. 2 shows a tree generated by method A as an example.

3.1.2. Random Generation of Cycles (Algorithm II)

For $p_5 = 0$ method A generates a further graph extension in the following steps:

Extension Step:
In each extension step first a vertex $x \hat{\in} W$ is randomly chosen but only accepted, if for a random degree $\hat{d}$ ($\hat{d} \hat{\in} \{1,2,..,\deg^*(x)\}$) holds $\hat{d} > \deg(x)$. That means, x is rejected with a probability $\frac{\deg(x)}{\deg^*(x)}$ and removed from W.
If x has been accepted another vertex $y \hat{\in} W \setminus \{x\}$ is randomly



○ initial vertex w ∈ W

—— current graph G = (V,E)

═══ a possible random path $P_{\overline{G}}(x,y^*)$

• vertices of the set $\overline{V}$
(edges of $\overline{E}$ are not presented)

⟶ possible directions of the considered shortest path

Fig. 1:: Illustration of the tree generation by method A (Algorithm I)

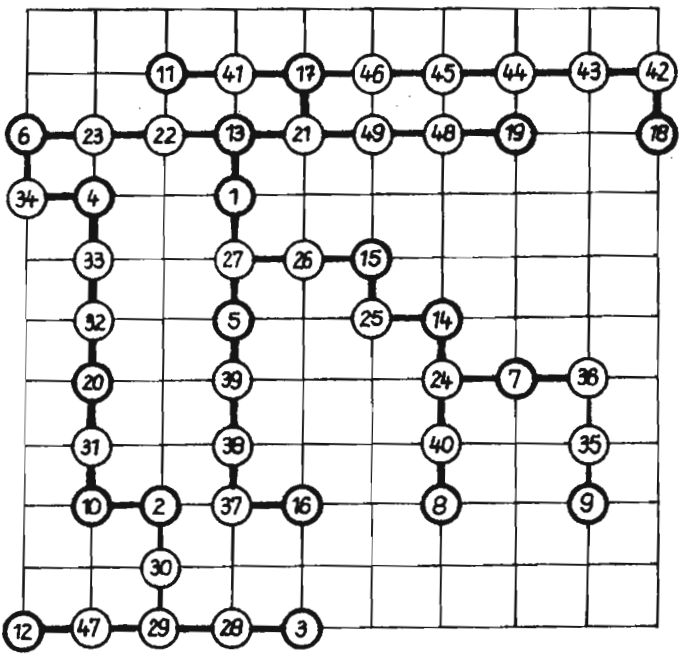


Fig. 2: A tree randomly generated by algorithm I

○ initial points

○ additional generated vertices

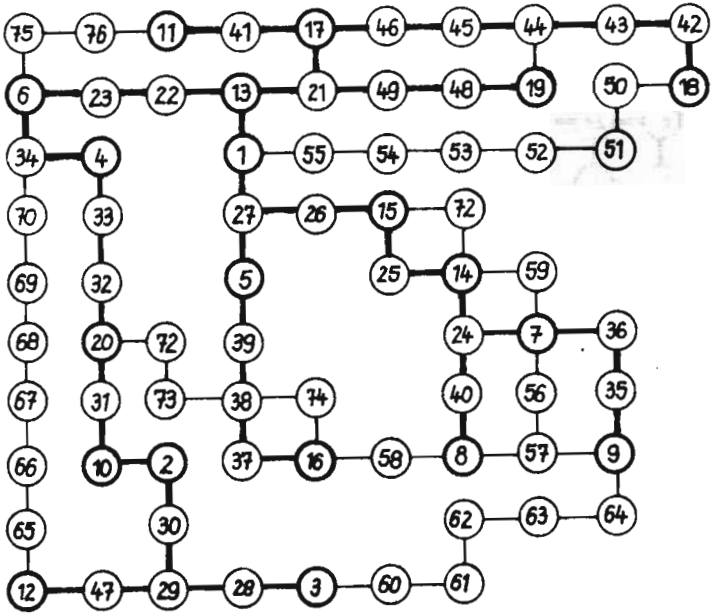


Fig. 3: Graph extension by method A (Algorithm II)

——— edges of the existing tree

——— additional edges created by algorithm II

chosen. Then the generation of a shortest path from x to y is iteratively carried out (starting with x) until a vertex $y^* \in V(G)$ is reached and the next iteration would certainly extend the path along an edge which already belongs to G. Again, in each iteration the optimal path is randomly extended by one element of $\bar{E}$.

Termination:
The extension is terminated if $W = \emptyset$.

Fig. 3 presents a graph layout produced by algorithm II.

3.2. Graph Generation by Method B (Algorithm III)

Method B generates graphs in the following steps:

Initialisation:
Only one vertex $x_0 \hat{\in} \bar{V}$ randomly chosen serves as initial point so that $G = (\{x_0\}, \emptyset)$ is the starting graph.

Extension Step:
In each extension step one vertex $x \hat{\in} \{i \in V(G) \mid deg(i) < \widehat{deg}(i)\}$ is randomly chosen. Then, a further vertex $y \hat{\in} \{i \in \bar{V} \mid d(x,i) = 1 \wedge deg(i) < \widehat{deg}(i) \wedge (x,i) \notin E(G) \wedge \wedge (p_5 = 0 \vee i \notin V(G))\}$ is randomly chosen. The new edge (x,y) and the possibly new vertex y are added to G.

Termination:
The extension is terminated if $|V(G)| = p_1$, or if $\{i \in V(G) : deg(i) < \widehat{deg}(i)\} = \emptyset$.

Fig. 4 presents an example of a graph generated by method B.

Remark 1: By a proper choice of the input parameters, in contrast to method A, the number of nodes and the maximum vertex degree can be controlled.

Remark 2: A special random preference of vertices $x \in V(G)$ with a certain degree (e. g. $deg(x) = 2$) can be useful for influencing the graph layout ("stretching").

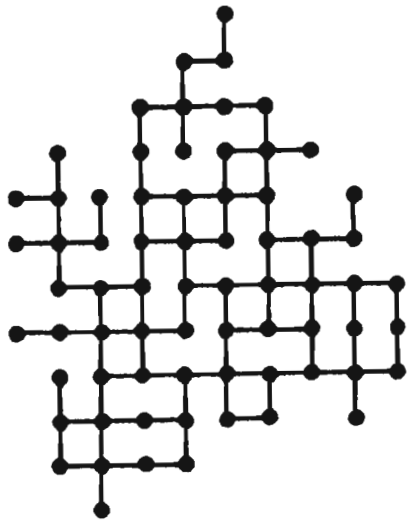


Fig. 4: Layout of a graph randomly generated by method B (Algorithm III)

Remark 3: If required it is easy to eliminate in the graphs generated by method A or B a certain percentage of vertices with degree two and to combine the adjacent edges into one edge. Of course, then the resulting edges of G are assigned to more than one edge of the grid.

4. The Complexity of Algorithms I, II, and III

In the following the worst-case computational effort required by the three algorithms described in Section 3 is presented. For the twodimensional case l_3 is equal to zero.

Algorithm I has the time complexity

$$O((l_1 + l_2 + l_3)(p_3 - 1))$$

Proof: Algorithm I produces a tree connecting p_3 initial vertices by the construction of at most $p_3 - 1$ paths. No path can have more than $l_1 + l_2 + l_3$ edges.

Algorithm II has a time complexity of

$$O((\alpha - 1)p_3(l_1 + l_2 + l_3)),$$

where $\alpha = 4$ or 6 in the case of 2- or 3-dimensional grids, respectively.

Proof: At most $(\alpha - 1)$ paths are constructed for each of the p_3 vertices of W being source nodes of the paths and each path contains at most $l_1 + l_2 + l_3$ edges.

Algorithm III has a time complexity of

$$O(p_1 p_2).$$

Proof: During the generation of the graph each of its p_1 vertices can be selected at most p_2-times.

5. The Output of a Graphic Representation

The generation of a graphic representation is carried out in the following steps:

In the first step the minimal window or cuboid is determined which is required for the graphic representation of the actual generated graph.

With the aim to generate a very compact graphic representation, especially by low-cost alphanumeric printers, in the second step a renumbering of the graph vertices is performed: The vertices are numbered line per line from left to right, whereby only the number of the first vertex in each line is printed (s. Fig. 5 and 6).

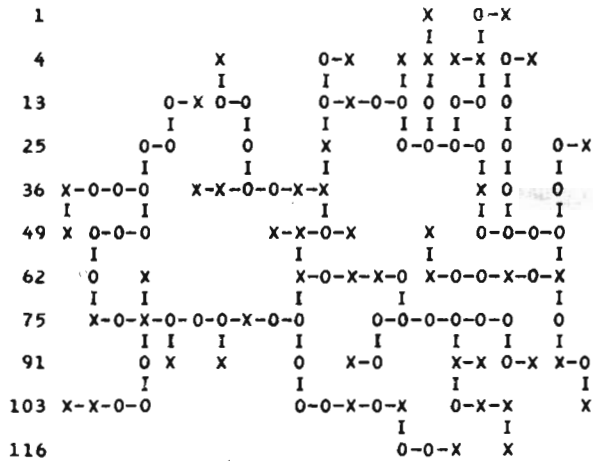


Fig. 5: An example of a generated tree
X - vertex of W
O - further vertices
I } edges
- }

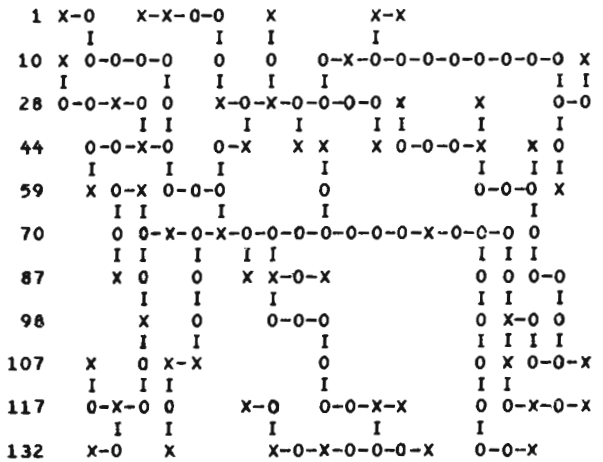


Fig. 6: Another example of a generated tree (same input parameters as in the case of the example presented in Fig. 5)

In the third step the generation of a graphic representation is performed line by line, whereby each horizontal line of the grid together with the information regarding the connections of the graph to the lower part of the grid is transformed into two lines of the printer.

References

/1/ Glover, F.: Computational Study of an Improved Shortest Path Algorithm. Networks, Vol. 14 (1984), 25 - 36.

/2/ Heck, J.: Zufallsgraphen. Ein Algorithmus und seine Anwendung in SIMULA 67. TU Berlin, 1975.