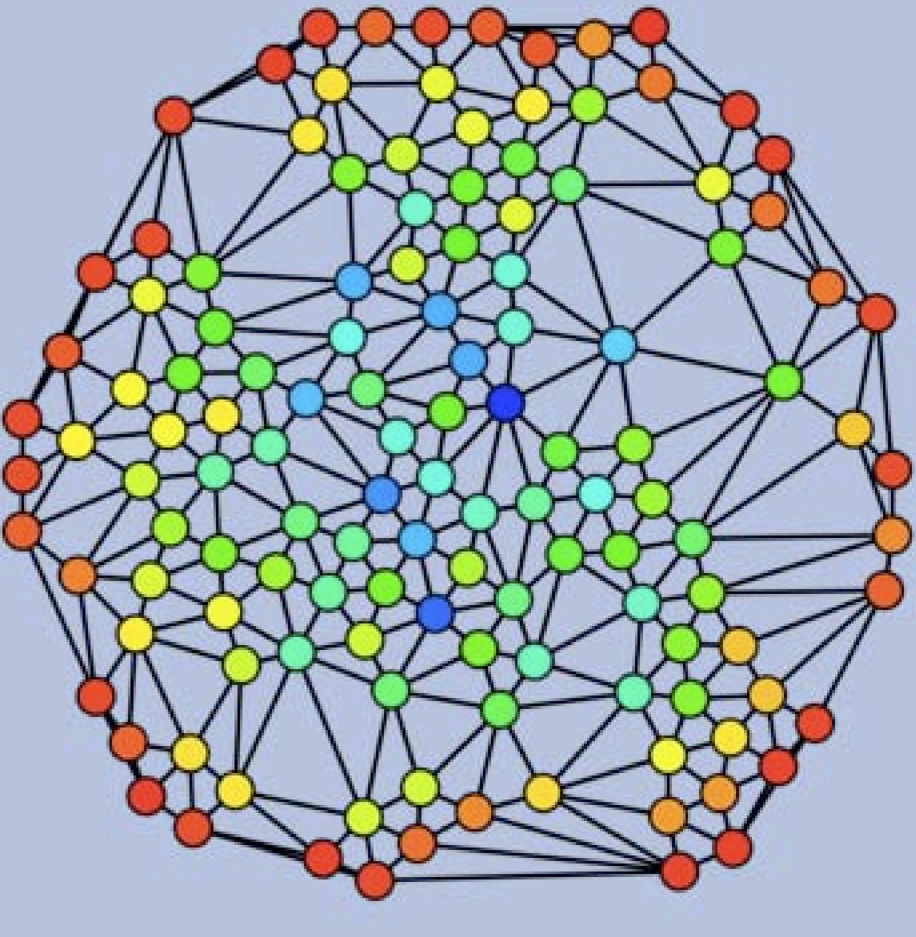
Social Network Analysis

Theory and Applications



Social Network Analysis

Theory and Applications

Contents

Theory

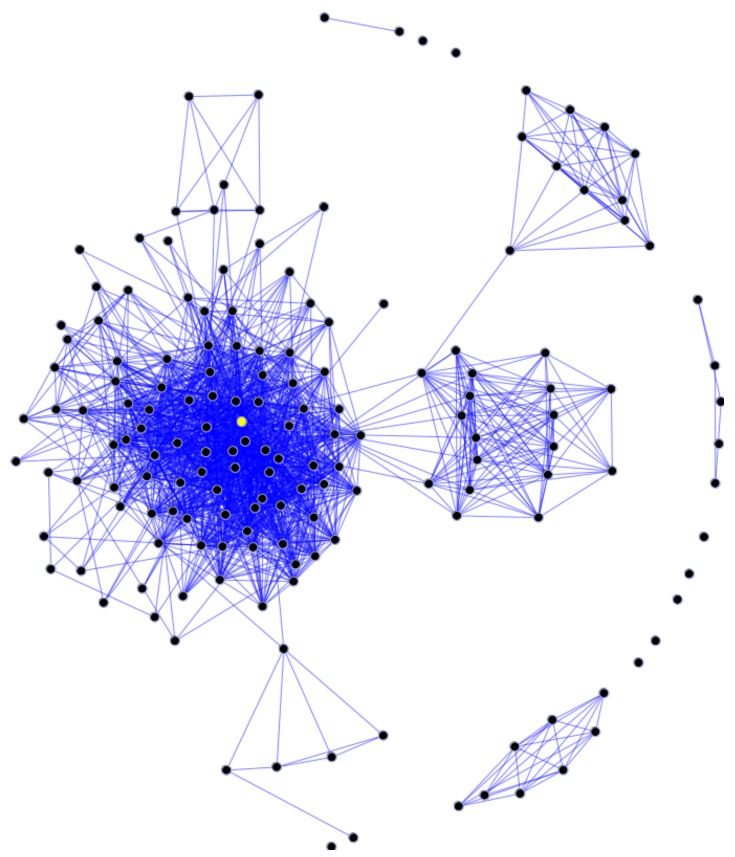
Social network

A **social network** is a social structure made up of individuals (or organizations) called "nodes", which are tied (connected) by one or more specific types of interdependency, such as friendship, kinship, common interest, financial exchange, dislike, sexual relationships, or relationships of beliefs, knowledge or prestige.

Social network analysis views social relationships in terms of network theory consisting of *nodes* and *ties* (also called *edges*, *links*, or *connections*). Nodes are the individual actors within the networks, and ties are the relationships between the actors. The resulting graph-based structures are often very complex. There can be many kinds of ties between the nodes. Research in a number of academic fields has shown that social networks operate on many levels, from families up to the level of nations, and play a critical role in determining the way problems are solved, organizations are run, and the degree to which individuals succeed in achieving their goals.

In its simplest form, a social network is a map of specified ties, such as friendship, between the nodes being studied. The nodes to which an individual is thus connected are the **social contacts** of that individual. The network can also be used to measure social capital – the value that an individual gets from the social network. These concepts are often displayed in a social network diagram, where nodes are the points and ties are the lines.

Social network analysis


An example of a social network diagram. The node with the highest betweenness centrality is marked in yellow.

Social network analysis (related to *network theory*) has emerged as a key technique in modern sociology. It has also gained a significant following in anthropology, biology, communication studies, economics, geography, information science, organizational studies, social psychology, and sociolinguistics, and has become a popular topic of speculation and study.

People have used the idea of **"social network"** loosely for over a century to connote complex sets of relationships between members of social systems at all scales, from interpersonal to international. In 1954, J. A. Barnes started using the term systematically to denote patterns of ties, encompassing concepts traditionally used by the public and those used by social scientists: bounded groups (e.g., tribes, families) and social categories (e.g., gender, ethnicity). Scholars such as S.D. Berkowitz, Stephen Borgatti, Ronald Burt, Kathleen Carley, Martin Everett, Katherine Faust, Linton Freeman, Mark Granovetter, David Knoke, David Krackhardt, Peter Marsden, Nicholas Mullins, Anatol Rapoport, Stanley Wasserman, Barry Wellman, Douglas R. White, and Harrison White expanded the use of systematic social network analysis.[1]

Social network analysis has now moved from being a suggestive metaphor to an analytic approach to a paradigm, with its own theoretical statements, methods, social network analysis software, and researchers. Analysts reason from whole to part; from structure to relation to individual; from behavior to attitude. They typically either study *whole networks* (also known as *complete networks*), all of the ties containing specified relations in a defined population, or *personal networks* (also known as *egocentric networks*), the ties that specified people have, such as their "personal communities".[2] The distinction between whole/complete networks and personal/egocentric networks has depended largely on how analysts were able to gather data. That is, for groups such as companies, schools, or membership societies, the analyst was expected to have complete information about who was in the network, all participants being both potential egos and alters. Personal/egocentric studies were typically conducted when identities of egos were known, but not their alters. These studies rely on the egos to provide information about the identities of alters and there is no expectation that the various egos or sets of alters will be tied to each other. A *snowball network* refers to the idea that the alters identified in an egocentric survey then become egos themselves and are able in turn to nominate additional alters. While there are severe logistic limits to conducting snowball network studies, a method for examining *hybrid networks* has recently been developed in which egos in complete networks can nominate alters otherwise not listed who are then available for all subsequent egos to see.[3] The hybrid network may be valuable for examining whole/complete networks that are expected to include important players beyond those who are formally identified. For example, employees of a company often work with non-company consultants who may be part of a network that cannot fully be defined prior to data collection.

Several analytic tendencies distinguish social network analysis:[4]

> There is no assumption that groups are the building blocks of society: the approach is open to studying less-bounded social systems, from nonlocal communities to links among websites.
>
> Rather than treating individuals (persons, organizations, states) as discrete units of analysis, it focuses on how the structure of ties affects individuals and their relationships.
>
> In contrast to analyses that assume that socialization into norms determines behavior, network analysis looks to see the extent to which the structure and composition of ties affect norms.

The shape of a social network helps determine a network's usefulness to its individuals. Smaller, tighter networks can be less useful to their members than networks with lots of loose connections (weak ties) to individuals outside the main network. More open networks, with many weak ties and social connections, are more likely to introduce new ideas and opportunities to their members than closed networks with many redundant ties. In other words, a group of friends who only do things with each other already share the same knowledge and opportunities. A group of individuals with connections to other social worlds is likely to have access to a wider range of information. It is better for individual success to have connections to a variety of networks rather than many connections within a single network. Similarly, individuals can exercise influence or act as brokers within their social networks by bridging two networks that are not directly linked (called filling structural holes).[5]

The power of social network analysis stems from its difference from traditional social scientific studies, which assume that it is the attributes of individual actors—whether they are friendly or unfriendly, smart or dumb, etc.—that matter. Social network analysis produces an alternate view, where the attributes of individuals are less important than their relationships and ties with other actors within the network. This approach has turned out to be useful for explaining many real-world phenomena, but leaves less room for individual agency, the ability for individuals to influence their success, because so much of it rests within the structure of their network.

Social networks have also been used to examine how organizations interact with each other, characterizing the many informal connections that link executives together, as well as associations and connections between individual employees at different organizations. For example, power within organizations often comes more from the degree to which an individual within a network is at the center of many relationships than actual job title. Social networks also play a key role in hiring, in business success, and in job performance. Networks provide ways for companies to gather information, deter competition, and collude in setting prices or policies.[6]

History of social network analysis

A summary of the progress of social networks and social network analysis has been written by Linton Freeman.[7]

Precursors of social networks in the late 1800s include Émile Durkheim and Ferdinand Tönnies. Tönnies argued that social groups can exist as personal and direct social ties that either link individuals who share values and belief (*gemeinschaft*) or impersonal, formal, and instrumental social links (*gesellschaft*). Durkheim gave a non-individualistic explanation of social facts arguing that social phenomena arise when interacting individuals constitute a reality that can no longer be accounted for in terms of the properties of individual actors. He distinguished between a traditional society – "mechanical solidarity" – which prevails if individual differences are minimized, and the modern society – "organic solidarity" – that develops out of cooperation between differentiated individuals with independent roles.

Georg Simmel, writing at the turn of the twentieth century, was the first scholar to think directly in social network terms. His essays pointed to the nature of network size on interaction and to the likelihood of interaction in ramified, loosely-knit networks rather than groups (Simmel, 1908/1971).

After a hiatus in the first decades of the twentieth century, three main traditions in social networks appeared. In the 1930s, J.L. Moreno pioneered the systematic recording and analysis of social interaction in small groups, especially classrooms and work groups (sociometry), while a Harvard group led by W. Lloyd Warner and Elton Mayo explored interpersonal relations at work. In 1940, A.R. Radcliffe-Brown's presidential address to British anthropologists urged the systematic study of networks.[8] However, it took about 15 years before this call was followed-up systematically.

Social network analysis developed with the kinship studies of Elizabeth Bott in England in the 1950s and the 1950s–1960s urbanization studies of the University of Manchester group of anthropologists (centered around Max Gluckman and later J. Clyde Mitchell) investigating community networks in southern Africa, India and the United Kingdom. Concomitantly, British anthropologist S.F. Nadel codified a theory of social structure that was influential in later network analysis.[9]

In the 1960s-1970s, a growing number of scholars worked to combine the different tracks and traditions. One group was centered around Harrison White and his students at the Harvard University Department of Social Relations: Ivan Chase, Bonnie Erickson, Harriet Friedmann, Mark Granovetter, Nancy Howell, Joel Levine, Nicholas Mullins, John Padgett, Michael Schwartz and Barry Wellman. Also independently active in the Harvard Social Relations department at the time were Charles Tilly, who focused on networks in political and community sociology and social movements, and Stanley Milgram, who developed the "six degrees of separation" thesis.[10] Mark Granovetter and Barry Wellman are among the former students of White who have elaborated and popularized social network analysis.[11]

Significant independent work was also done by scholars elsewhere: University of California Irvine social scientists interested in mathematical applications, centered around Linton Freeman, including John Boyd, Susan Freeman, Kathryn Faust, A. Kimball Romney and Douglas White; quantitative analysts at the University of Chicago, including Joseph Galaskiewicz, Wendy Griswold, Edward Laumann, Peter Marsden, Martina Morris, and John Padgett; and communication scholars at Michigan State University, including Nan Lin and Everett Rogers. A substantively-oriented University of Toronto sociology group developed in the 1970s, centered on former students of Harrison White: S.D. Berkowitz, Harriet Friedmann, Nancy Leslie Howard, Nancy Howell, Lorne Tepperman and Barry Wellman, and also including noted modeler and game theorist Anatol Rapoport.In terms of theory, it critiqued methodological individualism and group-based analyses, arguing that seeing the world as social networks offered more analytic leverage.[12]

Research

Social network analysis has been used in epidemiology to help understand how patterns of human contact aid or inhibit the spread of diseases such as HIV in a population. The evolution of social networks can sometimes be modeled by the use of agent based models, providing insight into the interplay between communication rules, rumor spreading and social structure.

SNA may also be an effective tool for mass surveillance – for example the Total Information Awareness program was doing in-depth research on strategies to analyze social networks to determine whether or not U.S. citizens were political threats.

Diffusion of innovations theory explores social networks and their role in influencing the spread of new ideas and practices. Change agents and opinion leaders often play major roles in spurring the adoption of innovations, although factors inherent to the innovations also play a role.

Robin Dunbar has suggested that the typical size of an egocentric network is constrained to about 150 members due to possible limits in the capacity of the human communication channel. The rule arises from cross-cultural studies in sociology and especially anthropology of the maximum size of a village (in modern parlance most reasonably understood as an *ecovillage*). It is theorized in evolutionary psychology that the number may be some kind of limit of average human ability to recognize members and track emotional facts about all members of a group. However, it may be due to economics and the need to track "free riders", as it may be easier in larger groups to take advantage of the benefits of living in a community without contributing to those benefits.

Mark Granovetter found in one study that more numerous weak ties can be important in seeking information and innovation. Cliques have a tendency to have more homogeneous opinions as well as share many common traits. This homophilic tendency was the reason for the members of the cliques to be attracted together in the first place. However, being similar, each member of the clique would also know more or less what the other members knew. To find new information or insights, members of the clique will have to look beyond the clique to its other friends and acquaintances. This is what Granovetter called "the strength of weak ties".

Guanxi is a central concept in Chinese society (and other East Asian cultures) that can be summarized as the use of personal influence. Guanxi can be studied from a social network approach.[13]

The small world phenomenon is the hypothesis that the chain of social acquaintances required to connect one arbitrary person to another arbitrary person anywhere in the world is generally short. The concept gave rise to the famous phrase six degrees of separation after a 1967 *small world experiment* by psychologist Stanley Milgram. In Milgram's experiment, a sample of US individuals were asked to reach a particular target person by passing a message along a chain of acquaintances. The average length of successful chains turned out to be about five intermediaries or six separation steps (the majority of chains in that study actually failed to complete). The methods (and ethics as well) of Milgram's experiment was later questioned by an American scholar, and some further research to replicate Milgram's findings had found that the degrees of connection needed could be higher.[14] Academic researchers continue to explore this phenomenon as Internet-based communication technology has supplemented the phone and postal systems available during the times of Milgram. A recent electronic small world experiment at Columbia University found that about five to seven degrees of separation are sufficient for connecting any two people through e-mail.[15]

Collaboration graphs can be used to illustrate good and bad relationships between humans. A positive edge between two nodes denotes a positive relationship (friendship, alliance, dating) and a negative edge between two nodes denotes a negative relationship (hatred, anger). Signed social network graphs can be used to predict the future evolution of the graph. In signed social networks, there is the concept of "balanced" and "unbalanced" cycles. A balanced cycle is defined as a cycle where the product of all the signs are positive. Balanced graphs represent a group of people who are unlikely to change their opinions of the other people in the group. Unbalanced graphs represent a group of people who are very likely to change their opinions of the people in their group. For example, a group of 3 people (A, B, and C) where A and B have a positive relationship, B and C have a positive relationship, but C and A have a negative relationship is an unbalanced cycle. This group is very likely to morph into a balanced cycle, such as one where B only has a good relationship with A, and both A and B have a negative relationship with C. By using the concept of balances and unbalanced cycles, the evolution of signed social network graphs can be predicted.

One study has found that happiness tends to be correlated in social networks. When a person is happy, nearby friends have a 25 percent higher chance of being happy themselves. Furthermore, people at the center of a social network tend to become happier in the future than those at the periphery. Clusters of happy and unhappy people were discerned within the studied networks, with a reach of three degrees of separation: a person's happiness was associated with the level of happiness of their friends' friends' friends.[16] (See also Emotional contagion.)

Some researchers have suggested that human social networks may have a genetic basis.[17] Using a sample of twins from the National Longitudinal Study of Adolescent Health, they found that in-degree (the number of times a person is named as a friend), transitivity (the probability that two friends are friends with one another), and betweenness centrality (the number of paths in the network that pass through a given person) are all significantly heritable. Existing models of network formation cannot account for this intrinsic node variation, so the researchers propose an alternative "Attract and Introduce" model that can explain heritability and many other features of human social networks.[18]

Metrics (measures) in social network analysis

Betweenness

The extent to which a node lies between other nodes in the network. This measure takes into account the connectivity of the node's neighbors, giving a higher value for nodes which bridge clusters. The measure reflects the number of people who a person is connecting indirectly through their direct links.[19]

Bridge

An edge is said to be a bridge if deleting it would cause its endpoints to lie in different components of a graph.

Centrality

This measure gives a rough indication of the social power of a node based on how well they "connect" the network. "Betweenness", "Closeness", and "Degree" are all measures of centrality.

Centralization

The difference between the number of links for each node divided by maximum possible sum of differences. A centralized network will have many of its links dispersed around one or a few nodes, while a decentralized network is one in which there is little variation between the number of links each node possesses.

Closeness

The degree an individual is near all other individuals in a network (directly or indirectly). It reflects the ability to access information through the "grapevine" of network members. Thus, closeness is the inverse of the sum of the shortest distances between each individual and every other person in the network. (See also: Proxemics) The shortest path may also be known as the "geodesic distance".

Clustering coefficient

A measure of the likelihood that two associates of a node are associates themselves. A higher clustering coefficient indicates a greater 'cliquishness'.

Cohesion

The degree to which actors are connected directly to each other by cohesive bonds. Groups are identified as ‘cliques’ if every individual is directly tied to every other individual, ‘social circles’ if there is less stringency of direct contact, which is imprecise, or as structurally cohesive blocks if precision is wanted.[20]

Degree

The count of the number of ties to other actors in the network. See also degree (graph theory).

(Individual-level) Density

The degree a respondent's ties know one another/ proportion of ties among an individual's nominees. Network or global-level density is the proportion of ties in a network relative to the total number possible (sparse versus dense networks).

Flow betweenness centrality

The degree that a node contributes to sum of maximum flow between all pairs of nodes (not that node).

Eigenvector centrality

A measure of the importance of a node in a network. It assigns relative scores to all nodes in the network based on the principle that connections to nodes having a high score contribute more to the score of the node in question.

Local bridge

An edge is a local bridge if its endpoints share no common neighbors. Unlike a bridge, a local bridge is contained in a cycle.

Path length

The distances between pairs of nodes in the network. Average path-length is the average of these distances between all pairs of nodes.

Prestige

In a directed graph prestige is the term used to describe a node's centrality. "Degree Prestige", "Proximity Prestige", and "Status Prestige" are all measures of Prestige. See also degree (graph theory).

Radiality

Degree an individual’s network reaches out into the network and provides novel information and influence.

Reach

The degree any member of a network can reach other members of the network.

Structural cohesion

The minimum number of members who, if removed from a group, would disconnect the group.[21]

Structural equivalence

Refers to the extent to which nodes have a common set of linkages to other nodes in the system. The nodes don’t need to have any ties to each other to be structurally equivalent.

Structural hole

Static holes that can be strategically filled by connecting one or more links to link together other points. Linked to ideas of social capital: if you link to two people who are not linked you can control their communication.

Network analytic software

Network analytic tools are used to represent the nodes (agents) and edges (relationships) in a network, and to analyze the network data. Like other software tools, the data can be saved in external files. Additional information comparing the various data input formats used by network analysis software packages is available at NetWiki. Network analysis tools allow researchers to investigate large networks like the Internet, disease transmission, etc. These tools provide mathematical functions that can be applied to the network model.

Visualization of networks

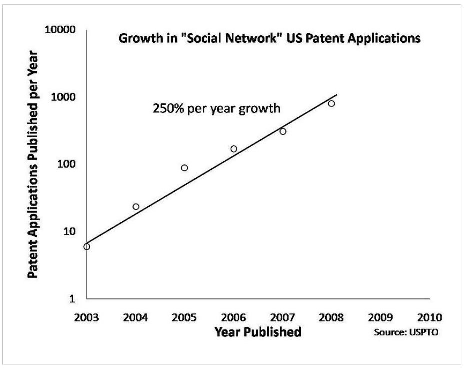
Visual representation of social networks is important to understand the network data and convey the result of the analysis [22]. Many of the analytic software have modules for network visualization. Exploration of the data is done through displaying nodes and ties in various layouts, and attributing colors, size and other advanced properties to nodes.

Typical representation of the network data are graphs in network layout (nodes and ties). These are not very easy-to-read and do not allow an intuitive interpretation. Various new methods have been developed in order to display network data in more intuitive format (e.g. Sociomapping).

Especially when using social network analysis as a tool for facilitating change, different approaches of participatory network mapping have proven useful. Here participants / interviewers provide network data by actually mapping out the network (with pen and paper or digitally) during the data collection session. One benefit of this approach is that it allows researchers to collect qualitative data and ask clarifying questions while the network data is collected.[22] Examples of network mapping techniques are Net-Map (pen-and-paper based) and VennMaker [24] (digital).

Patents

There has been rapid growth in the number of US patent applications that cover new technologies related to social networking. The number of published applications has been growing at about 250% per year over the past five years. There are now over 2000 published applications.[23] Only about 100 of these applications have issued as patents, however, largely due to the multi-year backlog in examination of business method patents and ethical issues connected with this patent category [24]



References

[1] Linton Freeman, *The Development of Social Network Analysis.* Vancouver: Empirical Press, 2006.
[2] Wellman, Barry and S.D. Berkowitz, eds., 1988. *Social Structures: A Network Approach.* Cambridge: Cambridge University Press.
[3] Hansen, William B. and Reese, Eric L. 2009. Network Genie User Manual (https://secure.networkgenie.com/admin/documentation/Network_Genie_Manual.pdf). Greensboro, NC: Tanglewood Research.
[4] Freeman, Linton. 2006. *The Development of Social Network Analysis.* Vancouver: Empirical Pres, 2006; Wellman, Barry and S.D. Berkowitz, eds., 1988. *Social Structures: A Network Approach.* Cambridge: Cambridge University Press.
[5] Scott, John. 1991. *Social Network Analysis.* London: Sage.
[6] Wasserman, Stanley, and Faust, Katherine. 1994. *Social Network Analysis: Methods and Applications.* Cambridge: Cambridge University Press.
[7] *The Development of Social Network Analysis* Vancouver: Empirical Press.
[8] A.R. Radcliffe-Brown, "On Social Structure," *Journal of the Royal Anthropological Institute:* 70 (1940): 1–12.
[9] Nadel, SF. 1957. *The Theory of Social Structure.* London: Cohen and West.
[10] The Networked Individual: A Profile of Barry Wellman (http://www.semioticon.com/semiotix/semiotix14/sem-14-05.html)
[11] Mullins, Nicholas. *Theories and Theory Groups in Contemporary American Sociology.* New York: Harper and Row, 1973; Tilly, Charles, ed. *An Urban World.* Boston: Little Brown, 1974; Mark Granovetter, "Introduction for the French Reader," *Sociologica* 2 (2007): 1–8; Wellman, Barry. 1988. "Structural Analysis: From Method and Metaphor to Theory and Substance." Pp. 19-61 in *Social Structures: A Network Approach*, edited by Barry Wellman and S.D. Berkowitz. Cambridge: Cambridge University Press.
[12] Mark Granovetter, "Introduction for the French Reader," *Sociologica* 2 (2007): 1–8; Wellman, Barry. 1988. "Structural Analysis: From Method and Metaphor to Theory and Substance." Pp. 19-61 in *Social Structures: A Network Approach*, edited by Barry Wellman and S.D. Berkowitz. Cambridge: Cambridge University Press. (see also Scott, 2000 and Freeman, 2004).
[13] Barry Wellman, Wenhong Chen and Dong Weizhen. "Networking Guanxi." Pp. 221–41 in Social Connections in China: Institutions, Culture and the Changing Nature of Guanxi, edited by Thomas Gold, Douglas Guthrie and David Wank. Cambridge University Press, 2002.
[14] *Could It Be A Big World After All?* (http://www.judithkleinfeld.com/ar_bigworld.html): Judith Kleinfeld article.
[15] Six Degrees: The Science of a Connected Age, Duncan Watts.
[16] James H. Fowler and Nicholas A. Christakis. 2008. " Dynamic spread of happiness in a large social network: longitudinal analysis over 20 years in the Framingham Heart Study. (http://www.bmj.com/cgi/content/full/337/dec04_2/a2338)" *British Medical Journal*. December 4, 2008: doi:10.1136/bmj.a2338. Media account for those who cannot retrieve the original: Happiness: It Really is Contagious (http://www.npr.org/templates/story/story.php?storyId=) Retrieved December 5, 2008.
[17] Shishkin, Philip (January 27, 2009). "Genes and the Friends You Make" (http://online.wsj.com/article/SB123302040874118079.html). *Wall Street Journal*. .
[18] Fowler, J. H.; Dawes, CT; Christakis, NA (10 February 2009). "Model of Genetic Variation in Human Social Networks" (http://jhfowler.ucsd.edu/genes_and_social_networks.pdf) (PDF). *Proceedings of the National Academy of Sciences* **106** (6): 1720–1724. doi:10.1073/pnas.0806746106. PMID 19171900. PMC 2644104. .
[19] The most comprehensive reference is: Wasserman, Stanley, & Faust, Katherine. (1994). Social Networks Analysis: Methods and Applications. Cambridge: Cambridge University Press. A short, clear basic summary is in Krebs, Valdis. (2000). "The Social Life of Routers." *Internet Protocol Journal*, 3 (December): 14–25.
[20] Cohesive.blocking (http://intersci.ss.uci.edu/wiki/index.php/Cohesive_blocking) is the R program for computing structural cohesion according to the Moody-White (2003) algorithm. This wiki site provides numerous examples and a tutorial for use with R.
[21] Moody, James, and Douglas R. White (2003). "Structural Cohesion and Embeddedness: A Hierarchical Concept of Social Groups." *American Sociological Review* 68(1):103–127. Online (http://www2.asanet.org/journals/ASRFeb03MoodyWhite.pdf): (PDF file).

[22] Bernie Hogan, Juan-Antonio Carrasco and Barry Wellman, "Visualizing Personal Networks: Working with Participant-Aided Sociograms," *Field Methods* 19 (2), May 2007: 116-144.

[23] USPTO search on published patent applications mentioning “social network” (http://appft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=/netahtml/PTO/search-adv.html&r=0&p=1&f=S&l=50&Query=spec/"social+network"&d=PG01)

[24] USPTO search on issued patents mentioning “social network” (http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=/netahtml/PTO/search-adv.htm&r=0&p=1&f=S&l=50&Query=spec/"social+network"&d=PTXT)

Further reading

- Barnes, J. A. "Class and Committees in a Norwegian Island Parish", *Human Relations* 7:39–58
- Berkowitz, Stephen D. 1982. *An Introduction to Structural Analysis: The Network Approach to Social Research.* Toronto: Butterworth. ISBN 0-409-81362-1
- Brandes, Ulrik, and Thomas Erlebach (Eds.). 2005. *Network Analysis: Methodological Foundations* (http://www.springeronline.com/3-540-24979-6/) Berlin, Heidelberg: Springer-Verlag.
- Breiger, Ronald L. 2004. "The Analysis of Social Networks." Pp. 505–526 in *Handbook of Data Analysis,* edited by Melissa Hardy and Alan Bryman. London: Sage Publications. ISBN 0-7619-6652-8 *Excerpts in pdf format* (http://www.u.arizona.edu/~breiger/NetworkAnalysis.pdf)
- Burt, Ronald S. (1992). *Structural Holes: The Structure of Competition.* Cambridge, MA: Harvard University Press. ISBN 0-674-84372-X
- **(Italian)** Casaleggio, Davide (2008). *TU SEI RETE. La Rivoluzione del business, del marketing e della politica attraverso le reti sociali.* ISBN 88-901826-5-2
- Carrington, Peter J., John Scott and Stanley Wasserman (Eds.). 2005. *Models and Methods in Social Network Analysis.* New York: Cambridge University Press. ISBN 978-0-521-80959-7
- Christakis, Nicholas and James H. Fowler "The Spread of Obesity in a Large Social Network Over 32 Years," *New England Journal of Medicine* 357 (4): 370–379 (26 July 2007)
- Doreian, Patrick, Vladimir Batagelj, and Anuška Ferligoj. (2005). *Generalized Blockmodeling.* Cambridge: Cambridge University Press. ISBN 0-521-84085-6
- Freeman, Linton C. (2004) *The Development of Social Network Analysis: A Study in the Sociology of Science.* Vancouver: Empirical Press. ISBN 1-59457-714-5
- Hill, R. and Dunbar, R. 2002. "Social Network Size in Humans." (http://www.liv.ac.uk/evolpsyc/Hill_Dunbar_networks.pdf) Human Nature, Vol. 14, No. 1, pp. 53–72.
- Jackson, Matthew O. (2003). "A Strategic Model of Social and Economic Networks". *Journal of Economic Theory* **71**: 44–74. doi:10.1006/jeth.1996.0108. pdf (http://merlin.fae.ua.es/fvega/CourseNetworks-Alicante/Artículos del curso/Jackson-Wolinsky-JET.pdf)
- Huisman, M. and Van Duijn, M. A. J. (2005). Software for Social Network Analysis. In P J. Carrington, J. Scott, & S. Wasserman (Editors), *Models and Methods in Social Network Analysis* (pp. 270–316). New York: Cambridge University Press. ISBN 978-0-521-80959-7
- Krebs, Valdis (2006) Social Network Analysis, A Brief Introduction. (Includes a list of recent SNA applications Web Reference (http://www.orgnet.com/sna.html).)
- Ligon, Ethan; Schechter, Laura, "The Value of Social Networks in rural Paraguay" (http://are.berkeley.edu/seminars/network value.pdf), University of California, Berkeley, Seminar, March 25, 2009 , Department of Agricultural & Resource Economics, College of Natural Resources, University of California, Berkeley
- Lima, Francisco W. S., Hadzibeganovic, Tarik, and Dietrich Stauffer (2009). Evolution of ethnocentrism on undirected and directed Barabási-Albert networks. Physica A, 388(24), 4999–5004.
- Lin, Nan, Ronald S. Burt and Karen Cook, eds. (2001). *Social Capital: Theory and Research.* New York: Aldine de Gruyter. ISBN 0-202-30643-7
- Mullins, Nicholas. 1973. *Theories and Theory Groups in Contemporary American Sociology.* New York: Harper and Row. ISBN 0-06-044649-8

- Müller-Prothmann, Tobias (2006): Leveraging Knowledge Communication for Innovation. Framework, Methods and Applications of Social Network Analysis in Research and Development, Frankfurt a. M. et al.: Peter Lang, ISBN 0-8204-9889-0.
- Manski, Charles F. (2000). "Economic Analysis of Social Interactions". *Journal of Economic Perspectives* **14**: 115–36. doi:10.1257/jep.14.3.115. (http://links.jstor.org/sici?sici=0895-3309(200022)14:3<115:EAOSI>2.0.CO;2-I&size=LARGE&origin=JSTOR-enlargePage) via JSTOR
- Moody, James, and Douglas R. White (2003). "Structural Cohesion and Embeddedness: A Hierarchical Concept of Social Groups." *American Sociological Review* 68(1):103–127. (http://www2.asanet.org/journals/ASRFeb03MoodyWhite.pdf)
- Newman, Mark (2003). "The Structure and Function of Complex Networks". *SIAM Review* **56**: 167–256. doi:10.1137/S003614450342480. pdf (http://www.santafe.edu/files/gems/paleofoodwebs/Newman2003SIAM.pdf)
- Nohria, Nitin and Robert Eccles (1992). *Networks in Organizations.* second ed. Boston: Harvard Business Press. ISBN 0-87584-324-7
- Nooy, Wouter d., A. Mrvar and Vladimir Batagelj. (2005). *Exploratory Social Network Analysis with Pajek.* Cambridge: Cambridge University Press. ISBN 0-521-84173-9
- Scott, John. (2000). *Social Network Analysis: A Handbook.* 2nd Ed. Newberry Park, CA: Sage. ISBN 0-7619-6338-3
- Sethi, Arjun. (2008). *Valuation of Social Networking* (http://fusion.dalmatech.com/~admin24/files/socialnetworkvaluation.pdf)
- Tilly, Charles. (2005). *Identities, Boundaries, and Social Ties.* Boulder, CO: Paradigm press. ISBN 1-59451-131-4
- Valente, Thomas W. (1995). *Network Models of the Diffusion of Innovations.* Cresskill, NJ: Hampton Press. ISBN 1-881303-21-7
- Wasserman, Stanley, & Faust, Katherine. (1994). *Social Network Analysis: Methods and Applications.* Cambridge: Cambridge University Press. ISBN 0-521-38269-6
- Watkins, Susan Cott. (2003). "Social Networks." Pp. 909–910 in *Encyclopedia of Population.* rev. ed. Edited by Paul George Demeny and Geoffrey McNicoll. New York: Macmillan Reference. ISBN 0-02-865677-6
- Watts, Duncan J. (2003). *Small Worlds: The Dynamics of Networks between Order and Randomness.* Princeton: Princeton University Press. ISBN 0-691-11704-7
- Watts, Duncan J. (2004). *Six Degrees: The Science of a Connected Age.* W. W. Norton & Company. ISBN 0-393-32542-3
- Wellman, Barry (1998). *Networks in the Global Village: Life in Contemporary Communities.* Boulder, CO: Westview Press. ISBN 0-8133-1150-0
- Wellman, Barry. 2001. "Physical Place and Cyber-Place: Changing Portals and the Rise of Networked Individualism." *International Journal for Urban and Regional Research* 25 (2): 227–52.
- Wellman, Barry and Berkowitz, Stephen D. (1988). *Social Structures: A Network Approach.* Cambridge: Cambridge University Press. ISBN 0-521-24441-2
- Weng, M. (2007). *A Multimedia Social-Networking Community for Mobile Devices* Interactive Telecommunications Program, Tisch School of the Arts/ New York University
- White, Harrison, Scott Boorman and Ronald Breiger. 1976. "Social Structure from Multiple Networks: I Blockmodels of Roles and Positions." *American Journal of Sociology* 81: 730–80.

External links

- Introduction to Stochastic Actor-Based Models for Network Dynamics - Snijder et al. (http://stat.gamma.rug.nl/SnijdersSteglichVdBunt2009.pdf)
- Social Networking (http://www.dmoz.org/Computers/Internet/On_the_Web/Online_Communities/Social_Networking/) at the Open Directory Project
- The International Network for Social Network Analysis (http://www.insna.org) (INSNA) – professional society of social network analysts, with more than 1,000 members
- Center for Computational Analysis of Social and Organizational Systems (CASOS) at Carnegie Mellon (http://www.casos.cs.cmu.edu)
- NetLab at the University of Toronto, studies the intersection of social, communication, information and computing networks (http://www.chass.utoronto.ca/~wellman/netlab/ABOUT/index.html)
- Netwiki (http://netwiki.amath.unc.edu/) (wiki page devoted to social networks; maintained at University of North Carolina at Chapel Hill)
- Building networks for learning (http://learningforsustainability.net/social_learning/networks.php) – A guide to on-line resources on strengthening social networking.
- Program on Networked Governance (http://www.ksg.harvard.edu/netgov) – Program on Networked Governance, Harvard University
- The International Workshop on Social Network Analysis and Mining (http://www.snakdd.com) (SNAKDD) - An annual workshop on social network analysis and mining, with participants from computer science, social science, and related disciplines.
- Historical Dynamics in a time of Crisis: Late Byzantium, 1204–1453 (a discussion of social network analysis from the point of view of historical studies) (http://www.oeaw.ac.at/byzanz/historicaldynamics.htm)

Analysis Software

Social network analysis software

Social network analysis software is used to identify, represent, analyze, visualize, or simulate nodes (e.g. agents, organizations, or knowledge) and edges (relationships) from various types of input data (relational and non-relational), including mathematical models of social networks. The output data can be saved in external files. Various input and output file formats exist.

Network analysis tools allow researchers to investigate representations of networks of different size - from small (e.g. families, project teams) to very large (e.g. the Internet, disease transmission). The various tools provide mathematical and statistical routines that can be applied to the network model.[1]

Visual representations of social networks are important to understand network data and convey the result of the analysis [2] . Visualization is often used as an additional or standalone data analysis method. With respect to visualization, network analysis tools are used to change the layout, colors, size and other properties of the network representation.

Social network tools are:

- For scholarly research tools like *UCINet* [3] , *Pajek* [4] , *ORA* [5] , the *statnet* [6] suite of packages in *R*, and *GUESS* [7] are popular.
- Examples of business oriented social network tools include *iPoint* [8] , *NetMiner* [9] , *InFlow*[10] , *Keyhubs* [11] , *Sentinel Visualizer* [12] , *KXEN Social Network*[13] , *NodeXL* [14] . For large networks with millions of nodes, try *Sonamine* [15] or ORA [16] . For mobile telecoms *Idiro SNA Plus* [17] is recommended
- An open source package with GUI for Linux, Windows and Mac, is *Social Networks Visualizer* or *SocNetV* [18] , developed in Qt/C++.
- Another generic open source package for Windows, Linux and OS X with interfaces to Python and R is "igraph" [19]
- Another generic open source package with [GUI] for Windows, Linux and OS X is "Tulip"
- *RapidNet* [20] is a generic freely available open source solution for network analysis and interactive visual network exploration and drill-down.
- For Mac OS X a related package installer of *SocNetV* [21] is available.
- For integrated egocentric data collection and visualization [22]

A systematic overview and comparison of a selection of software packages for social network analysis was provided by Huisman and Van Duijn.[23] The International Network for Social Network Analysis (INSNA) maintains a large list of software packages and libraries.[24]

Collection of Social Network Analysis Tools and Libraries

Product	Main Functionality	Input Format	Output Format	Platform	License and cost	Notes
AllegroGraph [25]	Graph Database. RDF with Gruff visualization tool	RDF	RDF	Linux, Mac, Windows	Free and Commercial	AllegroGraph is a graph database. It is disk-based, fully transactional OLTP database that stores data structured in graphs rather than in tables. AllegroGraph includes a Social Networking Analytics library [26]. Gruff [27] is a freely downloadable triple-store browser that displays visual graphs of subsets of a store's resources and their links. By selecting particular resources and predicates, you can build a visual graph that displays a variety of the relationships in a triple-store. Gruff can also display tables of all properties of selected resources or generate tables with SPARQL queries, and resources in the tables can be added to the visual graph.

AutoMap [28]	Network Text Analysis	.txt	DyNetML [29], .csv	Any (it's in Java)	Freeware for non-commercial use	Text mining tool that supports the extraction of relational data from texts. Distills three types of information: content analysis, semantic networks, ontologically coded networks. In order to do this, a variety of Natural Language Processing/ Information Extraction routines is provided (e.g. Stemming, Parts of Speech Tagging, Named-Entity Recognition, usage of user-defined ontologies, reduction and normalization, Anaphora Resolution, email data analysis, feature identification, entropy computation, reading and writing from and to default or user-specified database).
CFinder [30]	Finding and visualizing communities	.txt	.txt, .pfd, .ps, .svg, .svg, .emf, .gif, .raw, .ppm, .bmp, .jpg,.png, .wbmp	Linux, Mac OS X, Windows, Solaris	Freeware for non-commercial use	A software for finding and visualizing overlapping dense communities in networks, based on the clique percolation method. It enables customizable visualization and allows easy strolling over the found communities. The package contains a command line version of the program as well, suitable for scripting.

Commetrix [31]	Dynamic network visualization & analysis	Commetrix-Files, direct import from data sources/DB's, (standard DB and File Specs upcoming)	CSV Tables for SNA Metrics over time,(Graph Videos per Screencast), Keywords, Graphs, etc. in GUI	Any system supporting java (developed for Windows Platform)	Free trial, commercial licenses, free research collaboration (in beta-user group),	Commetrix is a Software Framework and Tool for Dynamic Network Analysis and Visualization. It provides easy exploratory access to network graphs and has been applied to study co-authorship, Instant Messaging, manual SNA surveys, e-mail, newsgroups, etc. Each node and each linking event can have properties, e.g. types of messages or rank of nodes, but also types, topics, or time stamps. This allows animations of network growth, structural change, and topic diffusion. A short introductory video is available on the website.
CoSBiLab Graph [32]	Network visualization, analysis and manipulation	.dot, .txt, .dl(UCINet), .spec(BetaWB), .txt (MRMC)	.dot, .txt, .dl(UCINet), .txt (MRMC), .pm(PRISM), .png	Windows (.NET 3.5 required)	Freeware for non-commercial use	CoSBiLab Graph is an application for visualization analysis and manipulation of networks. It provides a high customizable graphical representation of networks based on local properties. Nodes can be aggregated and arranged on the space manually or by choosing from a list of predefined layouts. A set of indices is provided for measuring the positional importance of nodes in the network and they can be combined together defining new mathematical expressions. The manual and a set of examples are available on the website.

Cytoscape [33]	General complex network data integration, analysis, and visualization.	SIF (Simple Interaction Format, GraphML, XGMML, GML, KGML, SBML, BioPAX, Excel, and text tables (including csv, tab delimited tables)	SIF, XGMML, GML, GraphML, Cytoscape Session(.cys), vector/bitmap images including jpg, png, pdf, ps.	Any system supporting Java	Open source (LGPL)	An open source platform for complex network data integration, analysis, and visualization. Originally Cytoscape was developed for bioinformatics research and now it is a problem domain independent platform. Many plugins are available for users and developers can expand its functionality by writing them.
Detica NetReveal [34]	Social Network Analysis for insurance or banking fraud, crime detection, intelligence, tax evasion, border control and network risk based targeting	csv, txt, XML and databases	csv, txt, XML and native Oracle database	Any system supporting Java	Commercial	A platform that can process billions (often at national scale) of multi-format data sources and builds social networks. In doing so, a single view of entity (customer, business, telephone, bank account, vehicle, address, citizen, etc.) can be generated across multiple, poor quality data sources. Social networks and entities can be scored using a range of powerful analytics and a full free text entity centric search is available across all records. The platform includes network visualization tools, workflow and real time rules engine to score incoming events in real time.
DEX [35]	Graph database for query processing and network analysis.	csv, jdbc	csv, graphml, graphviz	Linux & Windows	Free evaluation version (up to 1 Million nodes, no restriction on edges, 1 concurrent user). For larger graphs or commercial ask for licenses quotation.	DEX is a high-performance graph database written in Java and C++ . One of its main characteristics is its performance storage and retrieval for large graphs, in the order of billions of nodes, edges and attributes, allowing the analysis of large scale networks.

Discourse Network Analyzer [36]	Extract networks from structured text data	Text via copy&paste, .DNA files (a simple XML format)	DL, GraphML, CSV, Commetrix SQL, SON (Sonia)	Any system supporting Java 1.6	Freeware	Discourse Network Analyzer serves two purposes: manually coding text data for statements of actors in a QDA-like fashion, and exporting one- or two-mode networks from these structured data. Dynamic algorithms for the longitudinal analysis of discourses are available.
DyNet [37]	Data analysis	*.agf(proprietary), *.net(pajek), *.txt	*.agf(proprietary), *.net(pajek), *.txt	?	Proprietary(starting from $3000/user)	DyNet SE (Standard Edition) is an innovative software tool to analyse pools of complex data unveiling relations and interconnections via graphical and verbose outputs. DyNet SE is based on social network theory therefore relational data is visualised in terms of networks.
EgoNet Active Development [38] or Explanation [39]	Ego-centric network analysis	Conducts interviews or takes any valid XML file	Output to CSV and convertible to almost any other format	Any system supporting Java	Open Source, seeking contributors	Egonet is a program for the collection and analysis of egocentric network data. Egonet contains facilities to assist in creating the questionnaire, collecting the data and providing general global network measures and data matrixes that can be used in further analysis by other software programs.

EveSim [40]	EvESimulator	XML, SimCase	XML	Any system supporting Java	Open Source	The EvESimulator provides a simulation framework for biologically inspired P2P systems - the EvE as a part of the DBE. Although its focus is on the EvE, the EvESimulator simulates a DBE. Besides from that, the EvESimulator constitutes a collaborative platform for interdisciplinary research acting as a framework for understanding, visualising and presenting the DBE concepts to contributors.
Future Point Systems [41]	Visual analytics platform called Starlight for all-source analysis, including social network analysis (SNA)	Virtually any format, including MSFT Office, PDF, XML, .txt, database, HTML, web services, POP or IMAP mail servers, RSS, ESRI SHP	XML, CSV, ESRI SHP, KML, copy to clipboard, web reports, PDF, .jpg, .bmp, .png	Windows	Government GSA pricing and commercial pricing	Starlight is a comprehensive visual analytics platform that transforms mountains of data into actionable intelligence. SNA capabilities include centrality, path-finding and metrics support.
Financial Network Analyzer [42]	Tool for building and analyzing network time series	*.txt	*.txt, *.net (Pajek)	Any system supporting Java 1.5	Open Source (BSD)	Financial Network Analyzer (FNA) is an application for the statistical analysis of financial networks using methods developed in network science and social network analysis. It differentiates from other tools in that it builds networks from message (payments, trades, etc.) data and that it is geared towards the analysis of network times series.

Gephi [43]	Graph exploration and manipulation software	GraphViz(.dot), Graphlet(.gml), GUESS(.gdf), LEDA(.gml), NetworkX(.graphml, .net), NodeXL(.graphml, .net), Pajek(.net, .gml), Sonivis(.graphml), Tulip(.tlp, .dot), UCINET(.dl), yEd(.gml), Gephi (.gexf), Edge list(.csv), databases	GUESS(.gdf), Gephi(.gexf), .svg, .png	Any system supporting Java 1.6 and OpenGL	Open Source (GPL3), seeking contributors	Gephi is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs. It is a tool for people that have to explore and understand graphs. The user interacts with the representation, manipulate the structures, shapes and colors to reveal hidden properties. It uses a 3D render engine to display large networks in real-time and to speed up the exploration. A flexible and multi-task architecture brings new possibilities to work with complex data sets and produce valuable visual results.
GraphStream [44]	Dynamic Graph Library	GraphStream(.dgs), GraphViz(.dot), Graphlet(.gml), edge list	GraphStream(.dgs), GraphViz(.dot), Graphlet(.gml), image sequence	Any system supporting Java	Open Source	With GraphStream you deal with graphs. Static and Dynamic. You create them from scratch, from a file or any source. You display and render them.
graph-tool [45]	Python module for efficient analysis and visualization of graphs.	GraphViz(.dot), GraphML	GraphViz(.dot), GraphML, .bmp, .canon, .cmap, .eps, .fig, .gd, .gd2, .gif, .gtk, .ico, .imap, .cmapx, .ismap, .jpeg, .pdf, .plain, .png, .ps, .ps2, .svg, .svgz, .tif, .vml, .vmlz, .vrml, .wbmp, .xlib	GNU/Linux, Mac	Free Software (GPL3)	graph-tool is a python module for efficient analysis of graphs. Its core data structures and algorithms are implemented in C++, with heavy use of Template metaprogramming, based on the Boost Graph Library. It contains a comprehensive list of algorithms.

Graphviz	Graph vizualisation software	GraphViz(.dot)	.bmp, .canon, .cmap, .eps, .fig, .gd, .gd2, .gif, .gtk, .ico, .imap, .cmapx, .ismap, .jpeg, .pdf, .plain, .png, .ps, .ps2, .svg, .svgz, .tif, .vml, .vmlz, .vrml, .wbmp, .xlib	Linux, Mac, Windows	Open Source (CPL)	Graphviz is open source graph visualization framework. It has several main graph layout programs. Suitable for social network visualization.
Idiro SNA Plus [46] [46]	Highly scalable Social Network Analysis for Telecoms	All databases e.g. Oracle, DB2, Teradata & flat file	All databases & flat text Files	Linux	Commercial	Idiro SNA Plus is the market leading SNA platform for telecoms with a particular focus on churn prediction, viral marketing, acquisition and family unit identification. Idiro SNA Plus takes social network analysis from academia and into the realm of business where the focus is on deriving real value from the application of SNA to real-world problems. With Idiro SNA Plus users can do the following: **1. Churn** - Predict churners and quantify the damage a person would cause if they were to churn **2. Viral marketing** - Identify key influencers for viral marketing **3. Family units** - Identify family units for marketing purposes **4. Acquisition** - Identify targets for member-get-member campaigns **5. Rotational churn** - Identify rotational churners

igraph [47]	Analysis and visualization of very large networks	.txt (edge list), .graphml, .gml, .ncol, .lgl, .net	.txt (edge list), .graphml, .dot. .gml, .ncol, .lgl, .net	Windows, Linux, Mac OS X	Open source (GNU GPL)	igraph is a C library for the analysis of large networks. It includes fast implementations for classic graph theory problems and recent network analysis methods like community structure search, cohesive blocking, structural holes, dyad and triad census and motif count estimation. Higher level interfaces are available for R [48], Python [49] and Ruby [50].
iPoint [51]	Analysis and visualization of social networks trends, geo-location, age, gender and sentiment	Take any valid XML	XML , Flex	Windows, Linux, Mac OS X	Commercial	iPoint monitors and analyzes Consumer Generated Media, the full privacy of the author is maintained and its reporting dashboard reads from iMediaStreams web services. The analysis is easily viewed and managed from the worldwide, to the state, to the hyperlocal neighborhood level. It is this aggregation of news and topics, overlaid with sentiment and demographics, which provides a unique research tool into the areas that are uppermost on peoples minds.

InFlow [52]	Interactive network mapping and network metrics in one integrated application for social and organizational network analysis.	Easy data import from Microsoft Office[PC/MAC] and CSV files	Export graphics to Microsoft Office [PC/MAC] -- Powerpoint, Word, Visio—and network files to interactive Java applet for WWW	Windows 2000, XP, Vista	Commercial, Site, and Academic licenses available. Training & Mentoring in social network analysis, data gathering, and software application, is also available.	InFlow is intended for business users, and is designed for ease-of-use, multiple networks per node set, and what-if capabilities. Network data can be entered via 1) CSV files, from data bases and spreadsheets, 2) automated survey tools such as NetworkGenie, Optimice, etc. 3) data entry screens with paper surveys, or 4) drawn by hand with mouse, using node & link tools in graphics window. Most popular network metrics included: Density, Geodesics, Freeman Centralities, Watts-Strogatz Small World, Structural Equivalence, Cluster Analysis, Krackhardt E/I Ratio, and Krebs Reach & Weighted Average Path Length. Metrics are executed based on current network view—you measure what is mapped. Many network layouts are possible using automated algorithms and geometric layouts[arcs, lines, etc.] resulting in an unlimited number of custom views. Different actions can be taken on selected nodes vs. unselected nodes. Projects using InFlow can be viewed here [53].

Java Universal Network/Graph (JUNG) Framework	network and graph manipulation, analysis, and visualization	built-in support for GraphML, Pajek, and some text formats; user can create parsers for any desired format	built-in support for GraphML, Pajek, and some text formats; user can create exporters for any desired format	Any platform supporting Java	Open source (BSD license)	JUNG is a Java API and library that provides a common and extensible language for the modeling, analysis, and visualization of relational data. It supports a variety of graph types (including hypergraphs), supports graph elements of any type and with any properties, enables customizable visualizations, and includes algorithms from graph theory, data mining, and social network analysis (e.g., clustering, decomposition, optimization, random graph generation, statistical analysis, distances, flows, and centrality (PageRank, HITS, etc.)). It has been used to analyze networks in excess of 1 million nodes (although visualizations are currently more limited), and is limited only by the amount of memory allocated to Java.
Keyhubs [54]	A Simple, Online Tool for Mapping Informal Networks	www.keyhubs.com	www.keyhubs.com	Any platform supporting Internet Explorer	Free (Limited Version), Monthly Software Subscription (Full Version)	Keyhubs allows managers and business leaders to map human relationships (informal networks) in their organization quickly and easily via a simple, online software service: www.keyhubs.com.
KrackPlot [55]	Network visualization	UCINET, Mathematica	UCINET, Mathematica	?	?	KrackPlot is a program for network visualization designed for social network analysts.

KXEN Social Network (KSN)[56]	Powerful Social Network Analysis	All databases, Text Files, other Input formats	All databases, Text Files, other Output formats, graph structure export : dot format used by GraphViz, link structure.	Unix, Linux, Windows	Commercial	KSN is a Social Network Analysis module designed for extracting many Social Networks from CDRs, extracting many attributes from a Social Network, integrating Social Network attributes into the customers database and exploiting Social Network attributes to build predictive models
libSNA [57]	Basic network statistics	Csv	Csv	Any platform supporting python	Open source(LGPL)	libSNA is the premier open source library for conducting SNA research. Written in the object oriented programming language Python, libSNA provides a simple programming interface for applying SNA to large scale networks. libSNA is built on top of the open source library NetworkX; without NetworkX, libSNA would not be possible.
MetaSight [58]	Email / communication network visualization and analysis	MS Exchange and Lotus email servers	Interactive user interface	Windows Server	Commercial	MetaSight is an enterprise social software application which uses data from routine e-mail to infer and map business expertise and relationships. Applications include expertise location and external relationship management.
NEO4J [59]	Graph Database with several modules such as rdf or visualization	GraphML, rdf, csv, other	?	?	AGPL and commercial	Neo4j is a graph database. It is an embedded, disk-based, fully transactional Java persistence engine that stores data structured in graphs rather than in tables.

Network Overview Discovery Exploration for Excel (NodeXL) [60]	Network overview, discovery and exploration	email, .csv (text), .txt, .xls (Excel), .xslt (Excel 2007), .net (Pajek), .dl (UCINet), GraphML	.csv (text), .txt, .xls (Excel), .xslt (Excel 2007), .dl (UCINet), GraphML	Windows XP/Vista/7	Free (MPL)	NodeXL is a free and open Excel 2007 Add-in and C#/.Net library for network analysis and visualization. It integrates into Excel 2007 and 2010 and adds directed graph as a chart type to the spreadsheet and calculates a core set of network metrics and scores. Supports extracting email, Twitter, YouTube, WWW and flickr social networks. Accepts edge lists and matrix representations of graphs. Allows for easy manipulation and filtering of underlying data in spreadsheet format. Multiple network visualization layouts. Reads and writes UCINet and GraphML files.
NetMiner [61]	All-in-one Software for Network Analysis and Visualization	.xls(Excel), .csv(text), .dl(UCINET), .net(Pajek), .dat(StOCNET), .gml; NMF(proprietary)	.xls(Excel), .csv(text), .dl(UCINET), .net(Pajek), .dat(StOCNET), NMF(proprietary)	Microsoft Windows	Commercial ($16/user ~ $6,600/user) with free trial	NetMiner is a software tool for exploratory analysis and visualization of network data. Main features include : analysis of large networks, comprehensive network measures and models, both exploratory & confirmatory analysis, interactive visual analytics, what-if network analysis, built-in statistical procedures and charts, full documentation(1,000+ pages of User's Manual), expressive network data model, facilities for data & workflow management , and user-friendliness.

Network Genie [62]	Social Network Survey Data collection	Online survey and project design environment	Output to CSV, InFLow [52], NEGOPY [63], MultiNet [64], Pajek [65], Siena [66], and UCINET [67]	Any social web browser	Payment assessed for completed surveys at $.50 per survey. Data collection is free. Payment is required prior to data download.	Network Genie is used to: (1) Design complete, egocentric, and hybrid social network surveys using a wide variety of survey question formats; (2) Manage social network projects, including manage a collaborative team who have privileges defined by a project coordinator; (3) Collect social network data using online forms; and (4) Download and export data to the social network analysis program of your choice. Registration is free.
Network Workbench [68][69]	Modeling, Analysis and Visualization of Large Scale Networks	.net, .mat, .graphml, .nwb, .csv, xgmml	.net, .mat, .graphml, .nwb, .csv, xgmml, .eps, .pdf	Linux, Mac OS X, Windows, Solaris	Open Source (Apache 2.0)	Contains a variety of algorithms and features useful for analyzing networks, including Page Rank, Pathfinder Network Scaling, Small World network generation, and Blondel Community Detection to name a few. The underlying OSGi plugin model allows users to expand on Network Workbench's core functionality.

NetworkX	Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.	GML, Graph6/Sparse6, GraphML, GraphViz (.dot), NetworkX (.yaml, adjacency lists, and edge lists), Pajek (.net), LEDA	GML, Gnome Dia, Graph6/Sparse6, GraphML, GraphViz (.dot), NetworkX (.yaml, adjacency lists, and edge lists), Pajek (.net), LEDA, and assorted image formats (.jpg, .png, .ps, .svg, et al.)	Open source (GPL and similar)	Free	NetworkX (NX) is a rich integrated toolset for graph creation, manipulation, analysis, and visualization. User interface is through scripting/command-line provided by python. NX includes a large set of key algorithms, metrics and graph generators. Visualization is provided through pylab and graphviz. NX is an open-source project, in active development since < 2004 with an open bug-tracking site, and user forums. Development is sponsored by Los Alamos National Lab, and includes extensive unit-testing.
ONA Surveys [70]	Social Network Survey Data collection	Online survey and project design environment	Output to Microsoft Excel, Inflow [52] and Netdraw [71]	Microsoft Internet Explorer	Subscription based. Unlimited number of surveys. Pricing for commercial and academic use.	ONA Surveys is a fast and very user friendly survey tool aimed primarily at ONA/SNA practitioners to help collect data about relationships. Free registration provides full functionality, but export is limited to first 5 nodes. Paid subscription provides unlimited surveys with unlimited respondents. Supports multiple languages.

ORA [72]	Social Network Analysis, Network Visualization, Meta-Network Analysis, Trail Analysis, Geospatial Network Analysis, Network Generation	DyNetML [29], .csv	DyNetML, .csv	Windows	Freeware for non-commercial use	*ORA is a dynamic meta-network assessment and analysis tool containing hundreds of social network, dynamic network metrics, trail metrics, procedures for grouping nodes, identifying local patterns, comparing and contrasting networks, groups, and individuals from a dynamic meta-network perspective. *ORA has been used to examine how networks change through space and time, contains procedures for moving back and forth between trail data (e.g. who was where when) and network data (who is connected to whom, who is connected to where …), and has a variety of geo-spatial network metrics, and change detection techniques. *ORA can handle multi-mode, multi-plex, multi-level networks. It can identify key players, groups and vulnerabilities, model network changes over time, and perform COA analysis. It has been tested with large networks. Distance based, algorithmic, and statistical procedures for comparing and contrasting networks are part of this toolkit.

Pajek [65][73]	Analysis and Visualization of Large Scale Networks	.net, .paj, .dat(UCINET), .ged, .bs, .mac, .mol	.net, .paj, .dat(UCINET), .xml(graphML), .bs	Windows, Linux [74], Mac OS X [75]	Freeware for non-commercial use	A widely used program for drawing networks, Pajek also has significant analytical capabilities, and can be used to calculate most centrality measures, identify structural holes, block-model, and so on. Macros can be recorded to perform repetitive tasks. Data can be sent directly to R, to calculate additional statistics.
R	Social network analysis within the versatile and popular R environment	R will read in almost any format data file	R has write capability for most data formats	Windows, Linux, Mac	Open source	R contains several packages relevant for social network analysis: igraph [76] is a generic network analysis package; sna [77] performs sociometric analysis of networks; network [78] manipulates and displays network objects; tnet [79] performs analysis of weighted networks, two-mode networks, and longitudinal networks; ergm [80] implements exponential random graph models for networks; latentnet [81] has functions for network latent position and cluster models; degreenet [82] provides tools for statistical modeling of network degree distributions; and networksis [83] provides tools for simulating bipartite networks with fixed marginals. Most of these packages are part of the statnet [84] suite, obtainable through the statnet [85] meta-package.

Sentinel Visualizer [86]	Comprehensive network analysis and visualization	Structured XML, and databases such as SQL Server, Oracle, and Access. Also, Excel, Text and HTML formats.	Open database architecture in SQL Server; Structured XML and unstructured documents in Word, PDF, Excel, Text and HTML formats.	Microsoft Windows	Commercial (starting from $2000/user) 45-day free trial available	Sentinel Visualizer is a Windows-based program that provides data visualization [87], analysis [88] and knowledgebase management all within one product. Sentinel Visualizer produces interactive dynamic link charts, timeline and geospatial views, and provides a variety of analysis tools including Social Network Analysis [89], temporal analysis and entity and relationship weighting. Sentinel Visualizer includes a multi-user knowledgebase for efficiently and economically storing analysis data.

Social Networks Visualizer [90]	Social Networks Visualization and Analysis Tool	.xml (GraphML), .net (pajek), .dot (GraphViz), .sm/.net (Sociomatrix), .net (UCINET)	.xml (GraphML), .net (pajek), .dot (GraphViz), .sm/.net (Sociomatrix)	Linux, Windows, Mac (Qt toolkit needed)	Free Software (GPL3)	SocNetV (Social Networks Visualizer) is an open-source graphical application, developed in C++ language and the cross-platform Qt toolkit. The user interface is friendly and simple, allowing the researcher to draw social networks or plain graphs by clicking on a canvas. SocNetV computes basic network properties (i.e. density, diameter, shortest path lengths), as well as more advanced statistics, such as centralities (i.e. closeness, betweeness, graph), clustering coefficient, etc. Various layout algorithms are supported. For instance, nodes can be automatically positioned on circles or levels according to their betweeness centralities. Random networks and small world creation is also supported. SocNetV can handle any number of nodes, although with a speed penalty when nodes are more than 3000 or the graph is quite dense (many edges).

SocioMetrica [91]	EgoNet, LinkAlyzer, and VisuaLyzer applications	DyNetML, Excel, DL, text, UCINET	DyNetML, Excel, DL, text, UCINET, SPSS	Windows	Shareware	A set of applications for interview-based gathering of egocentric data (EgoNet), linking of data records through matching of node attributes (LinkAlyzer), and visualization (VisuaLyzer). VisuaLyzer also provides prototype functionality for analysis using a relational algebra model. A relational programming language, RAlog, derives and analyzes representations in this relation algebra.

SONAMINE [92]	Scalable network scoring and analysis up to hundreds of millions nodes and billion edges	any comma separated text file	comma separate text file	Windows, Linux	Commercial, free eval. Enterprise software license or hosted.	SONAMINE graph scoring engine [93] is a software for analysts that want to score millions of nodes on their windows laptop. Get eigenvector centrality in 3 min for 400,000 nodes and 1 million edges. SONAMINE graph analysis server [94] is a industrial quality enterprise software for companies that want to run massive batch graph analysis jobs. It distributes work over multiple servers, is fault tolerant and horizontally scalable. It is used for node scoring and data mining to improve data mining predictions for churn and marketing. SONAMINE graph query server [95] is a real time high performance graph query engine. It allows your consumer service or ad server to retrieve within 10ms what neighboring nodes of the current user has done. It improves social advertising click-through rates and speeds up initial user base adoption in custom social networks.

SONIVIS [96]	Network visualisation and analysis, especially Wiki-based information spaces	.xml(graphML)	.xml(graphML)	Windows, Linux	open-source (GPL)	SONIVIS:Tool is a Java-based, open-source application, which is based on the Eclipse Rich Client Platform (RCP). The user interface is organized into three main perspectives: Analysis, Manipulation, and Statistics. Besides various Wiki and network analysis metrics, the tool provides predefined and user-definable graphical analyses. It offers a quick overview on current Wiki states or developments
statnet [97]	Social network analysis within the versatile and popular R environment	R will read in almost any format data file	R has write capability for most data formats	Windows, Linux, Mac	Open source (GPL)	A suite of R packages for social network analysis: sna [77] performs sociometric analysis of networks; network [78] manipulates and displays network objects; ergm [80] implements exponential random graph models for networks; latentnet [81] has functions for network latent position and cluster models; degreenet [82] provides tools for statistical modeling of network degree distributions; and networksis [83] provides tools for simulating bipartite networks with fixed marginals; the statnet [85] meta-package allows for package management.

StOCNET [98]	Software package for the advanced statistical analysis of social networks	Text (.dat, .txt)	Text	Windows	Freeware/Open source	StOCNET is a software system for the advanced statistical analysis of social networks, focusing on probabilistic (stochastic) models. The program consists of several statistical models for network analysis. In the present version, six modules are implemented: BLOCKS (stochastic blockmodeling of relational data), p2 (analysis of binary network data with actor and/or dyadic covariates), PACNET (constructing a partial algebraic model for observed multiple complete networks using a statistical approach), SIENA (analysis of repeated measures on social networks and MCMC-estimation of exponential random graphs), ULTRAS (analysis of binary undirected network data using ultrametric measurement models), and ZO (simulation and/or enumeration of graphs with given degrees).
tnet [79]	Social network analysis of weighted, two-mode, and longitudinal networks in R	Edgelist	R has write capability for most data formats	Windows, Linux, Mac	Open source (GPL)	A packages for social network analysis of weighted, two-mode, and longitudinal networks. Possible extensions are discussed here [99]

Tulip	Social Network Analysis tool	Tulip format (.tlp), GraphViz (.dot), GML, txt, adjacency matrix	.tlp, .gml	Windows Vista, XP, 7/ Linux / Mac OS	LGPL	Tulip is an information visualization framework dedicated to the analysis and visualization of relational data. Tulip aims to provide the developer with a complete library, supporting the design of interactive information visualization applications for relational data that can be tailored to the problems he or she is addressing.
UCINET [67]	Social Network Analysis tool	Excel, DL, text, Pajek .net, Krackplot, Negopy, proprietary (##.d & ##.h)	Excel, DL, text, Pajek .net, Krackplot, Mage, Metis, proprietary (##.d & ##.h)	Windows	Shareware	A comprehensive package for the analysis of social network data as well as other 1-mode and 2-mode data. Can handle a maximum of 32,767 nodes (with some exceptions) although practically speaking many procedures get too slow around 5,000 - 10,000 nodes. Social network analysis methods include centrality measures, subgroup identification, role analysis, elementary graph theory, and permutation-based statistical analysis. In addition, the package has strong matrix analysis routines, such as matrix algebra and multivariate statistics.

UNISoN [100]	Download usenet messages and save SNA output files	Reads from free NNTP servers	Creates CSV files and Pajek .net files	Any system supporting Java	Freeware	A java application that can download Usenet messages from free NNTP servers, show the saved messages, then allow filtering of data to save to a Pajek network file or CSV file. It creates networks using the author of each post. If someone replies to a post, there is a unidirectional link created from the author of the post to the author of the message they are replying to. There is also a preview panel that shows the network visually.

UrlNet [101]	Generation of social network analysis program input files	World Wide Web pages, online search engine result sets, and Internet Web Service APIs	currently generates Pajek projects [65] and GUESS .gdf files [102]	Any (it's written in Python, requires v2.5 or higher [103]; source code is included)	Freeware for non-commercial use	UrlNet is a Python class library for generating networks based on Internet linkages. In the simplest case, UrlNet creates a tree by harvesting the outlink URLs from the page referenced by a root URL (level zero); retrieving each of those pages (level 1), harvesting their outlink URLs; retrieving those pages (level 2), harvesting their outlink URLs; et cetera to a caller-specified depth. UrlNet can also create "forests", the union of multiple tree networks. Specialized classes are provided for generation of networks from search engine result sets (6 search engines are currently supported). UrlNet can also utilize URL-based Web Service APIs to generate networks. Current examples include Technorati's Cosmos API and three types of networks utilizing APIs provided by the National Center for Biological Information (NCBI) [104]. More than 20 example programs are provided, along with a 65-page user manual, to help Python programmers get up to speed.

visone [105]	Interactive analysis and visualization of social networks	GraphML, UCINET (.dl), Pajek (.net), Excel (.csv), Matrices, and Edge lists	Same as input and images as PNG, PDF, JPEG	Java (Windows, Linux, MacOS)	Freeware for non-commercial use	Interactive graphical tool for manipulating, analyzing, and visualizing social networks. Analysis methods include centrality indices, clustering, cliques, components, and centralization. Generic graph layout algorithms and tailored network visualizations are available. visone supports many graphical properties and generates high-quality images in PNG, PDF, etc.
VisuaLyzer [91]	Network visualization	Edgelist/Edgearray, Excel or GraphML formats	Edgelist/Edgearray, Excel or GraphML formats	?	Commercial	Interactive tool for entering, visualizing and analyzing social network data. Create nodes and links directly in VisuaLyzer, or import data from Edgelist/Edgearray, Excel or GraphML formats.
WAND [106]	Ecological network analysis	Scor files	Scor files	Microsoft Windows	Open source(GPL)	
Xanalys Link Explorer [107]	Visual analytics combining link analysis with temporal and spatial analysis	ODBC databses, flat files, direct data entry	*.wf, *.xas (Proprietary formats), Excel	Microsoft Windows	Commercial, Free trial	Interactive visual analytics tool combining data acquisition and querying with link analysis, temporal analysis and spatial analysis (GIS) techniques. Integrates with other desktop applications and services such as Excel and Bing mapping
Plotonic [108]	social network analysis	keywords/phrases	graphics, text	PHP	Commercial	Interactive web service for tracking your brands on social networks. Compiles social network data into psychographic, demographic, and geographic charts and maps.

References

[1] "Only connect: Felix Grant looks at the application of data analysis software to social networks", *Scientific Computing World* June 2010: pp 9-10. (http://www.scientific-computing.com/features/feature.php?feature_id=277)
[2] (http://www.cmu.edu/joss/content/articles/volume1/Freeman.html)
[3] (http://www.analytictech.com/ucinet/ucinet.htm)
[4] (http://pajek.imfm.si/doku.php)
[5] (http://www.casos.cs.cmu.edu/projects/ora/)
[6] (http://cran.r-project.org/web/packages/statnet/)
[7] (http://graphexploration.cond.org/)
[8] (http://imediastreams.com/iServe_Demographics.htm)
[9] (http://www.netminer.com/NetMiner/home_01.jsp)
[10] (http://www.orgnet.com/inflow3.html)
[11] (http://www.keyhubs.com)
[12] (http://www.fmsasg.com/Products/SentinelVisualizer/)
[13] (http://www.socialnetworksanalysis.com)
[14] (http://www.codeplex.com/nodexl)
[15] (http://www.sonamine.com/home/index.php?option=com_content&view=article&id=8&Itemid=5)
[16] (http://www.casos.cs.cmu.edu/projects/ora/)
[17] (http://www.idiro.com)
[18] (http://socnetv.sourceforge.net/)
[19] (http://igraph.sourceforge.net)
[20] RapidNet (http://rapid-i.com/content/view/183/193/)
[21] (http://naranja.umh.es/~atg)
[22] SocioMetrica (http://www.mdlogix.com/solutions/additional.html#row1)
[23] Draft(PDF) (http://stat.gamma.rug.nl/snijders/Software for Social Network Analysis CUP_ch13_Oct2003.pdf)
[24] Computer Programs for Social Network Analysis (http://www.insna.org/software/software_old.html)

Notes

- Barnes, J. A. "Class and Committees in a Norwegian Island Parish", *Human Relations* 7:39-58
- Berkowitz, S. D. 1982. *An Introduction to Structural Analysis: The Network Approach to Social Research.* Toronto: Butterworth.
- Brandes, Ulrik, and Thomas Erlebach (Eds.). 2005. *Network Analysis: Methodological Foundations* (http://www.springeronline.com/3-540-24979-6/) Berlin, Heidelberg: Springer-Verlag.
- Breiger, Ronald L. 2004. "The Analysis of Social Networks." Pp. 505–526 in *Handbook of Data Analysis,* edited by Melissa Hardy and Alan Bryman. London: Sage Publications. *Excerpts in pdf format* (http://www.u.arizona.edu/~breiger/NetworkAnalysis.pdf)
- Burt, Ronald S. (1992). *Structural Holes: The Structure of Competition. Cambridge, MA: Harvard University Press.*
- Carrington, Peter J., John Scott and Stanley Wasserman (Eds.). 2005. *Models and Methods in Social Network Analysis.* New York: Cambridge University Press.
- Christakis, Nicholas and James H. Fowler "The Spread of Obesity in a Large Social Network Over 32 Years," *New England Journal of Medicine* 357 (4): 370-379 (26 July 2007)
- Doreian, Patrick, Vladimir Batagelj, and Anuska Ferligoj. (2005). *Generalized Blockmodeling.* Cambridge: Cambridge University Press.
- Freeman, Linton C. (2004) *The Development of Social Network Analysis: A Study in the Sociology of Science.* Vancouver: Empirical Press.
- Hansen, William B. and Reese, Eric L. 2009. Network Genie Users Manual (https://secure.networkgenie.com/admin/documentation/Network_Genie_Manual.pdf). Greensboro, NC: Tanglewood Research.
- Hill, R. and Dunbar, R. 2002. "Social Network Size in Humans." Human Nature, Vol. 14, No. 1, pp. 53–72. Google (http://www.google.com/search?q=cache:sZ_e9TbhRboJ:www.liv.ac.uk/evolpsyc/Hill_Dunbar_networks.pdf+social+network+size&hl=en&gl=ca&ct=clnk&cd=1)

- Jackson, Matthew O. (2003). "A Strategic Model of Social and Economic Networks". *Journal of Economic Theory* **71**: 44–74. doi:10.1006/jeth.1996.0108. pdf (http://merlin.fae.ua.es/fvega/CourseNetworks-Alicante/Artículos del curso/Jackson-Wolinsky-JET.pdf)
- Huisman, M. and Van Duijn, M. A. J. (2005). Software for Social Network Analysis (http://stat.gamma.rug.nl/Software for Social Network Analysis CUP_ch13_Oct2003.pdf). In P J. Carrington, J. Scott, & S. Wasserman (Editors), *Models and Methods in Social Network Analysis* (pp. 270–316). New York: Cambridge University Press.
- Krebs, Valdis (2002) Uncloaking Terrorist Networks, *First Monday*, volume 7, number 4 (Application of SNA software to terror nets Web Reference (http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/941/863/).)
- Krebs, Valdis (2008) A Brief Introduction to Social Network Analysis (Common metrics in most SNA software Web Reference (http://www.orgnet.com/sna.html).)
- Krebs, Valdis (2008) Various Case Studies & Projects using Social Network Analysis software Web Reference (http://www.orgnet.com/cases.html).)
- Lin, Nan, Ronald S. Burt and Karen Cook, eds. (2001). *Social Capital: Theory and Research.* New York: Aldine de Gruyter.
- Mullins, Nicholas. 1973. *Theories and Theory Groups in Contemporary American Sociology.* New York: Harper and Row.
- Müller-Prothmann, Tobias (2006): Leveraging Knowledge Communication for Innovation. Framework, Methods and Applications of Social Network Analysis in Research and Development, Frankfurt a. M. et al.: Peter Lang, ISBN 0-8204-9889-0.
- Manski, Charles F. (2000). "Economic Analysis of Social Interactions". *Journal of Economic Perspectives* **14**: 115–36. doi:10.1257/jep.14.3.115. (http://links.jstor.org/sici?sici=0895-3309(200022)14:3<115:EAOSI>2.0.CO;2-I&size=LARGE&origin=JSTOR-enlargePage) via JSTOR
- Moody, James, and Douglas R. White (2003). "Structural Cohesion and Embeddedness: A Hierarchical Concept of Social Groups." *American Sociological Review* 68(1):103-127. (http://www2.asanet.org/journals/ASRFeb03MoodyWhite.pdf)
- Newman, Mark (2003). "The Structure and Function of Complex Networks". *SIAM Review* **56**: 167–256. doi:10.1137/S003614450342480. pdf (http://www.santafe.edu/files/gems/paleofoodwebs/Newman2003SIAM.pdf)
- Nohria, Nitin and Robert Eccles (1992). *Networks in Organizations.* second ed. Boston: Harvard Business Press.
- Nooy, Wouter d., A. Mrvar and Vladimir Batagelj. (2005). *Exploratory Social Network Analysis with Pajek.* Cambridge: Cambridge University Press.
- Scott, John. (2000). *Social Network Analysis: A Handbook.* 2nd Ed. Newberry Park, CA: Sage.
- Tilly, Charles. (2005). *Identities, Boundaries, and Social Ties.* Boulder, CO: Paradigm press.
- Valente, Thomas. (1995). *Network Models of the Diffusion of Innovation.* Cresskill, NJ: Hampton Press.
- Wasserman, Stanley, & Faust, Katherine. (1994). *Social Networks Analysis: Methods and Applications.* Cambridge: Cambridge University Press.
- Watkins, Susan Cott. (2003). "Social Networks." Pp. 909–910 in *Encyclopedia of Population.* rev. ed. Edited by Paul Demeny and Geoffrey McNicoll. New York: Macmillan Reference.
- Watts, Duncan. (2003). *Small Worlds: The Dynamics of Networks between Order and Randomness.* Princeton: Princeton University Press.
- Watts, Duncan. (2004). *Six Degrees: The Science of a Connected Age.* W. W. Norton & Company.
- Wellman, Barry (1999). *Networks in the Global Village.* Boulder, CO: Westview Press.
- Wellman, Barry. 2001. "Physical Place and Cyber-Place: Changing Portals and the Rise of Networked Individualism." *International Journal for Urban and Regional Research* 25 (2): 227-52.

- Wellman, Barry and Berkowitz, S.D. (1988). *Social Structures: A Network Approach.* Cambridge: Cambridge University Press.
- Weng, M. (2007). *A Multimedia Social-Networking Community for Mobile Devices* Interactive Telecommunications Program, Tisch School of the Arts/ New York University
- White, Harrison, Scott Boorman and Ronald Breiger. 1976. "Social Structure from Multiple Networks: I Blockmodels of Roles and Positions." *American Journal of Sociology* 81: 730-80.

Some Definitions

Betweenness

Within graph theory and network analysis, there are various measures of the **centrality** of a vertex within a graph that determine the relative importance of a vertex within the graph (for example, how important a person is within a social network, or, in the theory of space syntax, how important a room is within a building or how well-used a road is within an urban network).

There are four measures of centrality that are widely used in network analysis: degree centrality, betweenness, closeness, and eigenvector centrality. For a review as well as generalizations to weighted networks, see Opsahl et al. (2010)[1] .

Degree centrality

The first, and simplest, is **degree centrality**. Degree centrality is defined as the number of links incident upon a node (i.e., the number of ties that a node has). Degree is often interpreted in terms of the immediate risk of node for catching whatever is flowing through the network (such as a virus, or some information). If the network is directed (meaning that ties have direction), then we usually define two separate measures of degree centrality, namely indegree and outdegree. Indegree is a count of the number of ties directed to the node, and outdegree is the number of ties that the node directs to others. For positive relations such as friendship or advice, we normally interpret indegree as a form of popularity, and outdegree as gregariousness.

For a graph $G := (V, E)$ with *n* vertices, the degree centrality $C_D(v)$ for vertex v is:

$$C_D(v) = \frac{\deg(v)}{n-1}$$

Calculating degree centrality for all nodes V in a graph takes $\Theta(V^2)$ in a dense adjacency matrix representation of the graph, and for edges E in a graph takes $\Theta(E)$ in a sparse matrix representation.
The definition of centrality can be extended to graphs. Let $v*$ be the node with highest degree centrality in G . Let $X := (Y, Z)$ be the n node connected graph that maximizes the following quantity (with $y*$ being the node with highest degree centrality in X):

$$H = \sum_{j=1}^{|Y|} C_D(y*) - C_D(y_j)$$

Then the degree centrality of the graph G is defined as follows:

$$C_D(G) = \frac{\sum_{i=1}^{|V|} [C_D(v*) - C_D(v_i)]}{H}$$

H is maximized when the graph X contains one node that is connected to all other nodes and all other nodes are connected only to this one central node (a star graph). In this case

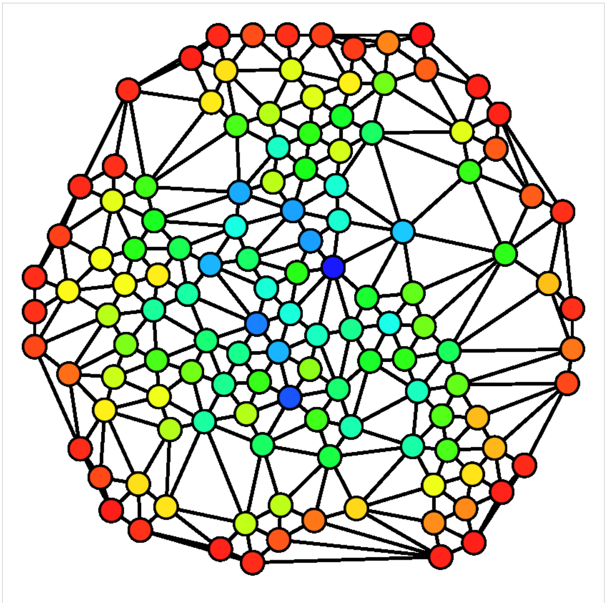
$$H = (n-1)(1 - \frac{1}{n-1}) = n - 2$$

so the degree centrality of G reduces to:

$$C_D(G) = \frac{\sum_{i=1}^{|V|} [C_D(v*) - C_D(v_i)]}{n-2}$$

Betweenness centrality

Betweenness is a centrality measure of a vertex within a graph (there is also edge betweenness, which is not discussed here). Vertices that occur on many shortest paths between other vertices have higher betweenness than those that do not.


Hue (from red=0 to blue=max) shows the node betweenness.

For a graph $G := (V, E)$ with n vertices, the betweenness $C_B(v)$ for vertex v is computed as follows:

1. For each pair of vertices (s,t), compute all shortest paths between them.

2. For each pair of vertices (s,t), determine the fraction of shortest paths that pass through the vertex in question (here, vertex v).

3. Sum this fraction over all pairs of vertices (s,t).

Or, more succinctly:[2]

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the number of shortest paths from *s* to *t*, and $\sigma_{st}(v)$ is the number of shortest paths from *s* to *t* that pass through a vertex *v*. This may be normalised by dividing through the number of pairs of vertices not including *v*, which is $(n-1)(n-2)$ for directed graphs and $(n-1)(n-2)/2$ for undirected graphs. For example, in an undirected star graph, the center vertex (which is contained in every possible shortest path) would have a betweenness of $(n-1)(n-2)/2$ (1, if normalised) while the leaves (which are contained in no shortest paths) would have a betweenness of 0.
Calculating the betweenness and closeness centralities of all the vertices in a graph involves calculating the shortest paths between all pairs of vertices on a graph. This takes $\Theta(V^3)$ time with the Floyd–Warshall algorithm, modified to not only find one but count all shortest paths between two nodes. On a sparse graph, Johnson's algorithm may be more efficient, taking $O(V^2 \log V + VE)$ time. On unweighted graphs, calculating betweenness centrality takes $O(VE)$ time using Brandes' algorithm[3] .
In calculating betweenness and closeness centralities of all vertices in a graph, it is assumed that graphs are undirected and connected with the allowance of loops and multiple edges. When specifically dealing with network graphs, oftentimes graphs are without loops or multiple edges to maintain simple relationships (where edges represent connections between two people or vertices). In this case, using Brandes' algorithm will divide final centrality scores by 2 to account for each shortest path being counted twice[3] .

Closeness centrality

In topology and related areas in mathematics, **closeness** is one of the basic concepts in a topological space. Intuitively we say two sets are close if they are arbitrarily near to each other. The concept can be defined naturally in a metric space where a notion of distance between elements of the space is defined, but it can be generalized to topological spaces where we have no concrete way to measure distances.

In graph theory **closeness** is a centrality measure of a vertex within a graph. Vertices that are 'shallow' to other vertices (that is, those that tend to have short geodesic distances to other vertices with in the graph) have higher closeness. Closeness is preferred in network analysis to mean shortest-path length, as it gives higher values to more central vertices, and so is usually positively associated with other measures such as degree.

In the network theory, **closeness** is a sophisticated measure of centrality. It is defined as the mean geodesic distance (i.e., the shortest path) between a vertex *v* and all other vertices reachable from it:

$$\frac{\sum\limits_{t\in V\setminus v} d_G(v,t)}{n-1}$$

where $n \geq 2$ is the size of the network's 'connectivity component' *V* reachable from *v*. Closeness can be regarded as a measure of how long it will take information to spread from a given vertex to other reachable vertices in the network[4] .

Some define closeness to be the reciprocal of this quantity, but either way the information communicated is the same (this time estimating the speed instead of the timespan). The closeness $C_C(v)$ for a vertex v is the reciprocal of the sum of geodesic distances to all other vertices of *V*[5] :

$$C_C(v) = \frac{1}{\sum_{t\in V\setminus v} d_G(v,t)}.$$

Different methods and algorithms can be introduced to measure closeness, like the *random-walk centrality* introduced by Noh and Rieger (2003) that is a measure of the speed with which randomly walking messages reach a vertex from elsewhere in the network—a sort of random-walk version of closeness centrality[6] .

The *information centrality* of Stephenson and Zelen (1989) is another closeness measure, which bears some similarity to that of Noh and Rieger. In essence it measures the harmonic mean length of paths ending at a vertex **i**, which is smaller if **i** has many short paths connecting it to other vertices[7] .

Dangalchev (2006), in order to measure the network vulnerability, modifies the definition for closeness so it can be used for disconnected graphs and the total closeness is easier to calculate[8] :

$$C_C(v) = \sum_{t\in V\setminus v} 2^{-d_G(v,t)}.$$

An extension to networks with disconnected components has been proposed by Opsahl (2010)[9] .

Eigenvector centrality

Eigenvector centrality is a measure of the importance of a node in a network. It assigns relative scores to all nodes in the network based on the principle that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. Google's PageRank is a variant of the Eigenvector centrality measure.

Using the adjacency matrix to find eigenvector centrality

Let x_i denote the score of the *i*th node. Let $A_{i,j}$ be the adjacency matrix of the network. Hence $A_{i,j} = 1$ if the *i*th node is adjacent to the *j*th node, and $A_{i,j} = 0$ otherwise. More generally, the entries in *A* can be real numbers representing connection strengths, as in a stochastic matrix.
For the i^{th} node, let the centrality score be proportional to the sum of the scores of all nodes which are connected to it. Hence

$$x_i = \frac{1}{\lambda} \sum_{j \in M(i)} x_j = \frac{1}{\lambda} \sum_{j=1}^{N} A_{i,j} x_j$$

where $M(i)$ is the set of nodes that are connected to the i^{th} node, *N* is the total number of nodes and λ is a constant. In vector notation this can be rewritten as

$\mathbf{x} = \frac{1}{\lambda}\mathbf{A}\mathbf{x}$, or as the eigenvector equation $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$

In general, there will be many different eigenvalues λ for which an eigenvector solution exists. However, the additional requirement that all the entries in the eigenvector be positive implies (by the Perron–Frobenius theorem) that only the greatest eigenvalue results in the desired centrality measure.[10] The i^{th} component of the related eigenvector then gives the centrality score of the i^{th} node in the network. Power iteration is one of many eigenvalue algorithms that may be used to find this dominant eigenvector.

Notes and references

[1] Opsahl, Tore; Agneessens, Filip; Skvoretz, John (2010). "Node centrality in weighted networks: Generalizing degree and shortest paths" (http://toreopsahl.com/2010/04/21/article-node-centrality-in-weighted-networks-generalizing-degree-and-shortest-paths/). *Social Networks* **32**: 245. doi:10.1016/j.socnet.2010.03.006. .

[2] Shivaram Narayanan. The Betweenness Centrality Of Biological Networks (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.958&rep=rep1&type=pdf). (Thesis).

[3] Ulrik Brandes (PDF). *A faster algorithm for betweenness centrality* (http://www.cs.ucc.ie/~rb4/resources/Brandes.pdf). .

[4] Newman, MEJ, 2003, Arxiv preprint cond-mat/0309045 (http://arxiv.org/abs/cond-mat/0309045).

[5] Sabidussi, G. (1966) The centrality index of a graph. *Psychometrika* **31**, 581--603.

[6] J. D. Noh and H. Rieger, Phys. Rev. Lett. 92, 118701 (2004).

[7] Stephenson, K. A. and Zelen, M., 1989. Rethinking centrality: Methods and examples. Social Networks 11, 1–37.

[8] Dangalchev Ch., Residual Closeness in Networks, Phisica A **365**, 556 (2006).

[9] Tore Opsahl. *Closeness centrality in networks with disconnected components* (http://toreopsahl.com/2010/03/20/closeness-centrality-in-networks-with-disconnected-components/). .

[10] M. E. J. Newman (PDF). *The mathematics of networks* (http://www-personal.umich.edu/~mejn/papers/palgrave.pdf). . Retrieved 2006-11-09.

Further reading

- Freeman, L. C. (1979). Centrality in social networks: Conceptual clarification. Social Networks, 1(3), 215-239.
- Sabidussi, G. (1966). The centrality index of a graph. Psychometrika, 31 (4), 581-603.
- Freeman, L. C. (1977) A set of measures of centrality based on betweenness. *Sociometry* **40**, 35-41.
- Koschützki, D.; Lehmann, K. A.; Peeters, L.; Richter, S.; Tenfelde-Podehl, D. and Zlotowski, O. (2005) Centrality Indices. In Brandes, U. and Erlebach, T. (Eds.) *Network Analysis: Methodological Foundations*, pp. 16–61, LNCS 3418, Springer-Verlag.
- Bonacich, P.(1987) Power and Centrality: A Family of Measures, *The American Journal of Sociology*, 92 (5), pp 1170–1182

External links

- https://networkx.lanl.gov/trac/attachment/ticket/119/page_rank.py

Centrality

Within graph theory and network analysis, there are various measures of the **centrality** of a vertex within a graph that determine the relative importance of a vertex within the graph (for example, how important a person is within a social network, or, in the theory of space syntax, how important a room is within a building or how well-used a road is within an urban network).

There are four measures of centrality that are widely used in network analysis: degree centrality, betweenness, closeness, and eigenvector centrality. For a review as well as generalizations to weighted networks, see Opsahl et al. (2010)[1] .

Degree centrality

The first, and simplest, is **degree centrality**. Degree centrality is defined as the number of links incident upon a node (i.e., the number of ties that a node has). Degree is often interpreted in terms of the immediate risk of node for catching whatever is flowing through the network (such as a virus, or some information). If the network is directed (meaning that ties have direction), then we usually define two separate measures of degree centrality, namely indegree and outdegree. Indegree is a count of the number of ties directed to the node, and outdegree is the number of ties that the node directs to others. For positive relations such as friendship or advice, we normally interpret indegree as a form of popularity, and outdegree as gregariousness.

For a graph $G := (V, E)$ with *n* vertices, the degree centrality $C_D(v)$ for vertex v is:

$$C_D(v) = \frac{\deg(v)}{n-1}$$

Calculating degree centrality for all nodes V in a graph takes $\Theta(V^2)$ in a dense adjacency matrix representation of the graph, and for edges E in a graph takes $\Theta(E)$ in a sparse matrix representation.
The definition of centrality can be extended to graphs. Let $v*$ be the node with highest degree centrality in G . Let $X := (Y, Z)$ be the n node connected graph that maximizes the following quantity (with $y*$ being the node with highest degree centrality in X):

$$H = \sum_{j=1}^{|Y|} C_D(y*) - C_D(y_j)$$

Then the degree centrality of the graph G is defined as follows:

$$C_D(G) = \frac{\sum_{i=1}^{|V|}[C_D(v*) - C_D(v_i)]}{H}$$

H is maximized when the graph X contains one node that is connected to all other nodes and all other nodes are connected only to this one central node (a star graph). In this case

$$H = (n-1)(1 - \frac{1}{n-1}) = n - 2$$

so the degree centrality of G reduces to:

$$C_D(G) = \frac{\sum_{i=1}^{|V|}[C_D(v*) - C_D(v_i)]}{n-2}$$

Betweenness centrality

Betweenness is a centrality measure of a vertex within a graph (there is also edge betweenness, which is not discussed here). Vertices that occur on many shortest paths between other vertices have higher betweenness than those that do not.

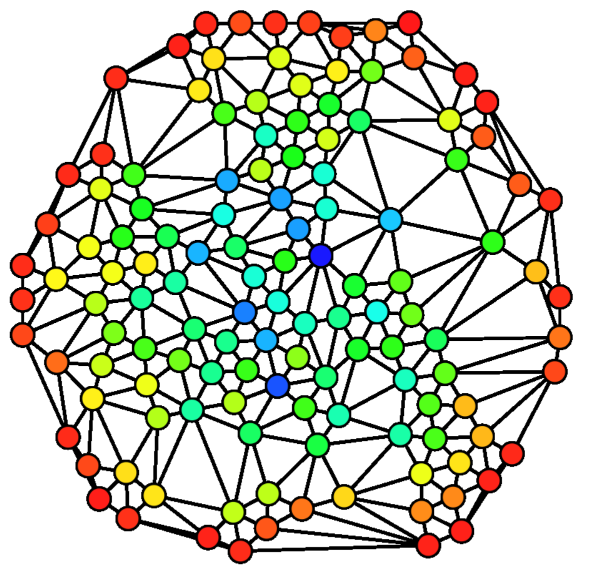
For a graph $G := (V, E)$ with n vertices, the betweenness $C_B(v)$ for vertex v is computed as follows:

1. For each pair of vertices (s,t), compute all shortest paths between them.

2. For each pair of vertices (s,t), determine the fraction of shortest paths that pass through the vertex in question (here, vertex v).

3. Sum this fraction over all pairs of vertices (s,t).

Or, more succinctly:[2]


Hue (from red=0 to blue=max) shows the node betweenness.

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the number of shortest paths from *s* to *t*, and $\sigma_{st}(v)$ is the number of shortest paths from *s* to *t* that pass through a vertex *v*. This may be normalised by dividing through the number of pairs of vertices not including *v*, which is $(n-1)(n-2)$ for directed graphs and $(n-1)(n-2)/2$ for undirected graphs. For example, in an undirected star graph, the center vertex (which is contained in every possible shortest path) would have a betweenness of $(n-1)(n-2)/2$ (1, if normalised) while the leaves (which are contained in no shortest paths) would have a betweenness of 0.

Calculating the betweenness and closeness centralities of all the vertices in a graph involves calculating the shortest paths between all pairs of vertices on a graph. This takes $\Theta(V^3)$ time with the Floyd–Warshall algorithm, modified to not only find one but count all shortest paths between two nodes. On a sparse graph, Johnson's algorithm may be more efficient, taking $O(V^2 \log V + VE)$ time. On unweighted graphs, calculating betweenness centrality takes $O(VE)$ time using Brandes' algorithm[3] .

In calculating betweenness and closeness centralities of all vertices in a graph, it is assumed that graphs are undirected and connected with the allowance of loops and multiple edges. When specifically dealing with network graphs, oftentimes graphs are without loops or multiple edges to maintain simple relationships (where edges represent connections between two people or vertices). In this case, using Brandes' algorithm will divide final centrality scores by 2 to account for each shortest path being counted twice[3] .

Closeness centrality

In topology and related areas in mathematics, **closeness** is one of the basic concepts in a topological space. Intuitively we say two sets are close if they are arbitrarily near to each other. The concept can be defined naturally in a metric space where a notion of distance between elements of the space is defined, but it can be generalized to topological spaces where we have no concrete way to measure distances.

In graph theory **closeness** is a centrality measure of a vertex within a graph. Vertices that are 'shallow' to other vertices (that is, those that tend to have short geodesic distances to other vertices with in the graph) have higher closeness. Closeness is preferred in network analysis to mean shortest-path length, as it gives higher values to more central vertices, and so is usually positively associated with other measures such as degree.

In the network theory, **closeness** is a sophisticated measure of centrality. It is defined as the mean geodesic distance (i.e., the shortest path) between a vertex *v* and all other vertices reachable from it:

$$\frac{\sum\limits_{t\in V\setminus v} d_G(v,t)}{n-1}$$

where $n \geq 2$ is the size of the network's 'connectivity component' *V* reachable from *v*. Closeness can be regarded as a measure of how long it will take information to spread from a given vertex to other reachable vertices in the network[4] .

Some define closeness to be the reciprocal of this quantity, but either way the information communicated is the same (this time estimating the speed instead of the timespan). The closeness $C_C(v)$ for a vertex v is the reciprocal of the sum of geodesic distances to all other vertices of *V*[5] :

$$C_C(v) = \frac{1}{\sum_{t\in V\setminus v} d_G(v,t)}.$$

Different methods and algorithms can be introduced to measure closeness, like the *random-walk centrality* introduced by Noh and Rieger (2003) that is a measure of the speed with which randomly walking messages reach a vertex from elsewhere in the network—a sort of random-walk version of closeness centrality[6] .

The *information centrality* of Stephenson and Zelen (1989) is another closeness measure, which bears some similarity to that of Noh and Rieger. In essence it measures the harmonic mean length of paths ending at a vertex $\mathbf{i}$, which is smaller if $\mathbf{i}$ has many short paths connecting it to other vertices[7] .

Dangalchev (2006), in order to measure the network vulnerability, modifies the definition for closeness so it can be used for disconnected graphs and the total closeness is easier to calculate[8] :

$$C_C(v) = \sum_{t\in V\setminus v} 2^{-d_G(v,t)}.$$

An extension to networks with disconnected components has been proposed by Opsahl (2010)[9] .

Eigenvector centrality

Eigenvector centrality is a measure of the importance of a node in a network. It assigns relative scores to all nodes in the network based on the principle that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. Google's PageRank is a variant of the Eigenvector centrality measure.

Using the adjacency matrix to find eigenvector centrality

Let x_i denote the score of the *i*th node. Let $A_{i,j}$ be the adjacency matrix of the network. Hence $A_{i,j} = 1$ if the *i*th node is adjacent to the *j*th node, and $A_{i,j} = 0$ otherwise. More generally, the entries in *A* can be real numbers representing connection strengths, as in a stochastic matrix.
For the i^{th} node, let the centrality score be proportional to the sum of the scores of all nodes which are connected to it. Hence

$$x_i = \frac{1}{\lambda} \sum_{j \in M(i)} x_j = \frac{1}{\lambda} \sum_{j=1}^{N} A_{i,j} x_j$$

where $M(i)$ is the set of nodes that are connected to the i^{th} node, *N* is the total number of nodes and λ is a constant. In vector notation this can be rewritten as

$\mathbf{x} = \frac{1}{\lambda}\mathbf{A}\mathbf{x}$, or as the eigenvector equation $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$

In general, there will be many different eigenvalues λ for which an eigenvector solution exists. However, the additional requirement that all the entries in the eigenvector be positive implies (by the Perron–Frobenius theorem) that only the greatest eigenvalue results in the desired centrality measure.[10] The i^{th} component of the related eigenvector then gives the centrality score of the i^{th} node in the network. Power iteration is one of many eigenvalue algorithms that may be used to find this dominant eigenvector.

Notes and references

[1] Opsahl, Tore; Agneessens, Filip; Skvoretz, John (2010). "Node centrality in weighted networks: Generalizing degree and shortest paths" (http://toreopsahl.com/2010/04/21/article-node-centrality-in-weighted-networks-generalizing-degree-and-shortest-paths/). *Social Networks* **32**: 245. doi:10.1016/j.socnet.2010.03.006. .

[2] Shivaram Narayanan. The Betweenness Centrality Of Biological Networks (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.958&rep=rep1&type=pdf). (Thesis).

[3] Ulrik Brandes (PDF). *A faster algorithm for betweenness centrality* (http://www.cs.ucc.ie/~rb4/resources/Brandes.pdf). .

[4] Newman, MEJ, 2003, Arxiv preprint cond-mat/0309045 (http://arxiv.org/abs/cond-mat/0309045).

[5] Sabidussi, G. (1966) The centrality index of a graph. *Psychometrika* **31**, 581--603.

[6] J. D. Noh and H. Rieger, Phys. Rev. Lett. 92, 118701 (2004).

[7] Stephenson, K. A. and Zelen, M., 1989. Rethinking centrality: Methods and examples. Social Networks 11, 1–37.

[8] Dangalchev Ch., Residual Closeness in Networks, Phisica A **365**, 556 (2006).

[9] Tore Opsahl. *Closeness centrality in networks with disconnected components* (http://toreopsahl.com/2010/03/20/closeness-centrality-in-networks-with-disconnected-components/). .

[10] M. E. J. Newman (PDF). *The mathematics of networks* (http://www-personal.umich.edu/~mejn/papers/palgrave.pdf). . Retrieved 2006-11-09.

Further reading

- Freeman, L. C. (1979). Centrality in social networks: Conceptual clarification. Social Networks, 1(3), 215-239.
- Sabidussi, G. (1966). The centrality index of a graph. Psychometrika, 31 (4), 581-603.
- Freeman, L. C. (1977) A set of measures of centrality based on betweenness. *Sociometry* **40**, 35-41.
- Koschützki, D.; Lehmann, K. A.; Peeters, L.; Richter, S.; Tenfelde-Podehl, D. and Zlotowski, O. (2005) Centrality Indices. In Brandes, U. and Erlebach, T. (Eds.) *Network Analysis: Methodological Foundations*, pp. 16–61, LNCS 3418, Springer-Verlag.
- Bonacich, P.(1987) Power and Centrality: A Family of Measures, *The American Journal of Sociology*, 92 (5), pp 1170–1182

External links

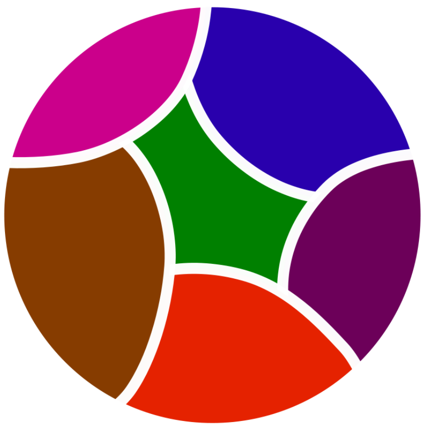
- https://networkx.lanl.gov/trac/attachment/ticket/119/page_rank.py

Equivalence relation

In mathematics, an **equivalence relation** is, loosely, a relation that specifies how to partition a set such that every element of the set is in exactly one of the blocks in the partition, and the union of all the blocks equals the original set. Two elements of the set are considered equivalent (with respect to the equivalence relation) if and only if they are elements of the same block.

Notation

Although various notations are used throughout the literature to denote that two elements a and b of a set are equivalent with respect to an equivalence relation R, the most common are "$a \sim b$" and "$a \equiv b$", which are used when R is the obvious relation being referenced, and variations of "$a \sim_R b$", "$a \equiv_R b$", or "aRb".



An equivalence relation partitions a set into several disjoint subsets, called equivalence classes. All the elements in a given equivalence class are equivalent among themselves, and no element is equivalent with any element from a different class.

Definition

A given binary relation ~ on a set A is said to be an equivalence relation if and only if it is reflexive, symmetric and transitive. Equivalently, for all a, b and c in A:

- $a \sim a$. (Reflexivity)
- if $a \sim b$ then $b \sim a$. (Symmetry)
- if $a \sim b$ and $b \sim c$ then $a \sim c$. (Transitivity)

A together with the relation ~ is called a setoid. The equivalence class of a under ~, denoted $[a]$, is defined as $[a] = \{b \in A | a \sim b\}$.

Reflexivity follows from symmetry and transitivity if for every element $a \in A$, there exists another element $b \in A$ such that $a \sim b$ holds. However, reflexivity does not follow from symmetry and transitivity alone. For example, let A be the set of integers, and let two elements of A be related if they are both even numbers. This relation is clearly symmetric and transitive, but in view of the existence of odd numbers, it is not reflexive.

On the other hand, let *A* be the set of integers, and let two elements of *A* be related if their *difference* is even. This *is* an equivalence relation, which partitions the integers into two equivalence classes, the even and odd integers.

Examples

Equivalence relations

The following are all equivalence relations:

- "Has the same birthday as" on the set of all people.
- "Is similar to" or "congruent to" on the set of all triangles.
- "Is congruent to modulo *n*" on the integers.
- "Has the same image under a function" on the elements of the domain of the function.
- "Is parallel to" on the set of subspaces of an affine space.

Relations that are not equivalences

- The relation "≥" between real numbers is reflexive and transitive, but not symmetric. For example, $7 \geq 5$ does not imply that $5 \geq 7$. It is, however, a partial order.
- The relation "has a common factor greater than 1 with" between natural numbers greater than 1, is reflexive and symmetric, but not transitive. (Example: The natural numbers 2 and 6 have a common factor greater than 1, and 6 and 3 have a common factor greater than 1, but 2 and 3 do not have a common factor greater than 1).
- The empty relation *R* on a non-empty set *X* (i.e. *aRb* is never true) is vacuously symmetric and transitive, but not reflexive. (If *X* is also empty then *R is* reflexive.)
- The relation "is approximately equal to" between real numbers, even if more precisely defined, is not an equivalence relation, because although reflexive and symmetric, it is not transitive, since multiple small changes can accumulate to become a big change. However, if the approximation is defined asymptotically, for example by saying that two functions *f* and *g* are approximately equal near some point if the limit of *f-g* is 0 at that point, then this defines an equivalence relation.
- The relation "is a sibling of" (used to connote pairs of distinct people who have the same parents) on the set of all human beings is not an equivalence relation. Although siblinghood is symmetric (if *A* is a sibling of *B*, then *B* is a sibling of *A*) and transitive on any 3 distinct people (if *A* is a sibling of *B* and *C* is a sibling of *B*, then *A* is a sibling of *C,* provided *A* is not *C*), it is not reflexive (*A* cannot be a sibling of *A*).

Connections to other relations

- A partial order is a relation that is reflexive, *antisymmetric*, and transitive.
- A congruence relation is an equivalence relation whose domain *X* is also the underlying set for an algebraic structure, and which respects the additional structure. In general, congruence relations play the role of kernels of homomorphisms, and the quotient of a structure by a congruence relation can be formed. In many important cases congruence relations have an alternative representation as substructures of the structure on which they are defined. E.g. the congruence relations on groups correspond to the normal subgroups.
- Equality is both an equivalence relation and a partial order. Equality is also the only relation on a set that is reflexive, symmetric and antisymmetric.
- A strict partial order is irreflexive, transitive, and asymmetric.
- A partial equivalence relation is transitive and symmetric. Transitive and symmetric imply reflexive if and only if for all $a \in X$ exists $b \in X$ such that $a \sim b$.
- A dependency relation is reflexive and symmetric.
- A preorder is reflexive and transitive.
- A compatibility relation is reflexive and symmetric.

Well-definedness under an equivalence relation

If ~ is an equivalence relation on X, and $P(x)$ is a property of elements of X, such that whenever $x \sim y$, $P(x)$ is true if $P(y)$ is true, then the property P is said to be well-defined or a *class invariant* under the relation ~.

A frequent particular case occurs when f is a function from X to another set Y; if $x_1 \sim x_2$ implies $f(x_1) = f(x_2)$ then f is said to be a *morphism* for ~, a *class invariant under* ~, or simply *invariant under* ~. This occurs, e.g. in the character theory of finite groups. The latter case with the function f can be expressed by a commutative triangle. See also invariant. Some authors use "compatible with ~" or just "respects ~" instead of "invariant under ~".

More generally, a function may map equivalent arguments (under an equivalence relation $\sim_A$) to equivalent values (under an equivalence relation $\sim_B$). Such a function is known as a morphism from $\sim_A$ to $\sim_B$.

Equivalence class, quotient set, partition

Let X be a nonempty set, and let $a, b \in X$. Some definitions:

Equivalence class

The set of all a and b for which $a \sim b$ holds make up an **equivalence class** of X by ~. Let $[a] := \{x \in X | x \sim a\}$ denote the equivalence class to which a belongs. Then all elements of X equivalent to each other are also elements of the same equivalence class.

Quotient set

The set of all possible equivalence classes of X by ~, denoted $X/\sim := \{[x] | x \in X\}$, is the **quotient set** of X by ~. If X is a topological space, there is a natural way of transforming $X/\sim$ into a topological space; see quotient space for the details.

Projection

The **projection** of ~ is the function $\pi : X \to X/\sim$ defined by $\pi(x) = [x]$ which maps elements of X into their respective equivalence classes by ~.

Theorem on projections:[1] Let the function f: $X \to B$ be such that $a \sim b \to f(a) = f(b)$. Then there is a unique function $g : X/\sim \to B$, such that $f = g\pi$. If f is a surjection and $a \sim b \leftrightarrow f(a) = f(b)$, then g is a bijection.

Equivalence kernel

The **equivalence kernel** of a function f is the equivalence relation ~ defined by $x \sim y \iff f(x) = f(y)$. The equivalence kernel of an injection is the identity relation.

Partition

A **partition** of X is a set P of subsets of X, such that every element of X is an element of a single element of P. Each element of P is a *cell* of the partition. Moreover, the elements of P are pairwise disjoint and their union is X.

Counting possible partitions

Let X be a finite set with n elements. Since every equivalence relation over X corresponds to a partition of X, and vice versa, the number of possible equivalence relations on X equals the number of distinct partitions of X, which is the *nth* Bell number B_n:

$$B_n = \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!},$$

where the above is one of the ways to write the nth Bell number.

Fundamental Theorem of Equivalence Relations

A key result links equivalence relations and partitions:[2] [3]

- An equivalence relation ~ on a set *X* partitions *X*.
- Conversely, corresponding to any partition of *X*, there exists an equivalence relation ~ on *X*.

In both cases, the cells of the partition of *X* are the equivalence classes of *X* by ~. Since each element of *X* belongs to a unique cell of any partition of *X*, and since each cell of the partition is identical to an equivalence class of *X* by ~, each element of *X* belongs to a unique equivalence class of *X* by ~. Thus there is a natural bijection from the set of all possible equivalence relations on *X* and the set of all partitions of *X*.

Comparing equivalence relations

If ~ and ≈ are two equivalence relations on the same set *S*, and $a{\sim}b$ implies $a{\approx}b$ for all $a,b \in S$, then ≈ is said to be a **coarser** relation than ~, and ~ is a **finer** relation than ≈. Equivalently,

- ~ is finer than ≈ if every equivalence class of ~ is a subset of an equivalence class of ≈, and thus every equivalence class of ≈ is a union of equivalence classes of ~.
- ~ is finer than ≈ if the partition created by ~ is a refinement of the partition created by ≈.

The equality equivalence relation is the finest equivalence relation on any set, while the trivial relation that makes all pairs of elements related is the coarsest.

The relation "~ is finer than ≈" on the collection of all equivalence relations on a fixed set is itself a partial order relation.

Generating equivalence relations

- Given any set *X*, there is an equivalence relation over the set $[X\to X]$ of all possible functions $X\to X$. Two such functions are deemed equivalent when their respective sets of fixpoints have the same cardinality, corresponding to cycles of length one in a permutation. Functions equivalent in this manner form an equivalence class on $[X\to X]$, and these equivalence classes partition $[X\to X]$.
- An equivalence relation ~ on *X* is the equivalence kernel of its surjective projection $\pi : X \to X/{\sim}$.[4] Conversely, any surjection between sets determines a partition on its domain, the set of preimages of singletons in the codomain. Thus an equivalence relation over *X*, a partition of *X*, and a projection whose domain is *X*, are three equivalent ways of specifying the same thing.
- The intersection of any collection of equivalence relations over *X* (viewed as a subset of $X \times X$) is also an equivalence relation. This yields a convenient way of generating an equivalence relation: given any binary relation *R* on *X*, the equivalence relation *generated by R* is the smallest equivalence relation containing *R*. Concretely, *R* generates the equivalence relation $a \sim b$ if and only if there exist elements $x_1, x_2, ..., x_n$ in *X* such that $a = x_1$, $b = x_n$, and $(x_i,x_{i+1})\in R$ or $(x_{i+1},x_i)\in R$, $i = 1, ..., n-1$.

  Note that the equivalence relation generated in this manner can be trivial. For instance, the equivalence relation ~ generated by:

  - Any total order on *X* has exactly one equivalence class, *X* itself, because $x \sim y$ for all *x* and *y*;
  - Any subset of the identity relation on *X* has equivalence classes that are the singletons of *X*.
- Equivalence relations can construct new spaces by "gluing things together." Let *X* be the unit Cartesian square $[0,1] \times [0,1]$, and let ~ be the equivalence relation on *X* defined by $\forall a, b \in [0,1]\ ((a, 0) \sim (a, 1) \wedge (0, b) \sim (1, b))$. Then the quotient space $X/{\sim}$ can be naturally identified with a torus: take a square piece of paper, bend and glue together the upper and lower edge to form a cylinder, then bend the resulting cylinder so as to glue together its two open ends, resulting in a torus.

Algebraic structure

Much of mathematics is grounded in the study of equivalences, and order relations. It is very well known that lattice theory captures the mathematical structure of order relations. Even though equivalence relations are as ubiquitous in mathematics as order relations, the algebraic structure of equivalences is not as well known as that of orders. The former structure draws primarily on group theory and, to a lesser extent, on the theory of lattices, categories, and groupoids.

Group theory

Just as order relations are grounded in ordered sets, sets closed under pairwise supremum and infimum, equivalence relations are grounded in partitioned sets, which are sets closed under bijections and preserve partition structure. Since all such bijections map an equivalence class onto itself, such bijections are also known as permutations. Hence permutation groups (also known as transformation groups) and the related notion of orbit shed light on the mathematical structure of equivalence relations.

Let '~' denote an equivalence relation over some nonempty set A, called the universe or underlying set. Let G denote the set of bijective functions over A that preserve the partition structure of A: $\forall x \in A\ \forall g \in G\ (g(x) \in [x])$. Then the following three connected theorems hold:[5]

- ~ partitions A into equivalence classes. (This is the *Fundamental Theorem of Equivalence Relations,* mentioned above);
- Given a partition of A, G is a transformation group under composition, whose orbits are the cells of the partition‡;
- Given a transformation group G over A, there exists an equivalence relation ~ over A, whose equivalence classes are the orbits of G.[6] [7]

In sum, given an equivalence relation ~ over A, there exists a transformation group G over A whose orbits are the equivalence classes of A under ~.

This transformation group characterisation of equivalence relations differs fundamentally from the way lattices characterize order relations. The arguments of the lattice theory operations meet and join are elements of some universe A. Meanwhile, the arguments of the transformation group operations composition and inverse are elements of a set of bijections, $A \to A$.

Moving to groups in general, let H be a subgroup of some group G. Let ~ be an equivalence relation on G, such that $a \sim b \leftrightarrow (ab^{-1} \in H)$. The equivalence classes of ~—also called the orbits of the action of H on G—are the right **cosets** of H in G. Interchanging a and b yields the left cosets.

‡*Proof.*[8] Let function composition interpret group multiplication, and function inverse interpret group inverse. Then G is a group under composition, meaning that $\forall x \in A\ \forall g \in G\ ([g(x)] = [x])$, because G satisfies the following four conditions:

- *G is closed under composition*. The composition of any two elements of G exists, because the domain and codomain of any element of G is A. Moreover, the composition of bijections is bijective;[9]
- *Existence of identity element*. The identity function, $I(x)=x$, is an obvious element of G;
- *Existence of inverse function*. Every bijective function g has an inverse g^{-1}, such that $gg^{-1} = I$;
- *Composition associates*. $f(gh) = (fg)h$. This holds for all functions over all domains.[10]

Let f and g be any two elements of G. By virtue of the definition of G, $[g(f(x))] = [f(x)]$ and $[f(x)] = [x]$, so that $[g(f(x))] = [x]$. Hence G is also a transformation group (and an automorphism group) because function composition preserves the partitioning of A. $\square$

Related thinking can be found in Rosen (2008: chpt. 10).

Categories and groupoids

The composition of morphisms central to category theory, denoted here by concatenation, generalizes the composition of functions central to transformation groups. The axioms of category theory assert that the composition of morphisms associates, and that the left and right identity morphisms exist for any morphism.

If a morphism f has an inverse, f is an isomorphism, i.e., there exists a morphism g such that the compositions fg and gf equal the appropriate identity morphisms. Hence the category-theoretic concept nearest to an equivalence relation is a (small) category whose morphisms are all isomorphisms. Groupoid is another name for a small category of this nature.

Let G be a set and let "~" denote an equivalence relation over G. Then we can form a groupoid representing this equivalence relation as follows. The objects are the elements of G, and for any two elements x and y of G, there exists a unique morphism from x to y if and only if $x \sim y$. The elements x and y are "equivalent" if there is an element g of the groupoid from x to y. There may be many such g, each of which can be regarded as a distinct "proof" that x and y are equivalent.

The advantages of regarding an equivalence relation as a special case of a groupoid include:

- Whereas the notion of "free equivalence relation" does not exist, that of a free groupoid on a directed graph does. Thus it is meaningful to speak of a "presentation of an equivalence relation," i.e., a presentation of the corresponding groupoid;
- Bundles of groups, group actions, sets, and equivalence relations can be regarded as special cases of the notion of groupoid, a point of view that suggests a number of analogies;
- In many contexts "quotienting," and hence the appropriate equivalence relations often called congruences, are important. This leads to the notion of an internal groupoid in a category.[11]

Lattices

The possible equivalence relations on any set X, when ordered by set inclusion, form a complete lattice, called **Con** X by convention. The canonical map **ker**: X^$X \rightarrow$ **Con** X, relates the monoid X^X of all functions on X and **Con** X. **ker** is surjective but not injective. Less formally, the equivalence relation **ker** on X, takes each function f: $X \rightarrow X$ to its kernel **ker** f. Likewise, **ker(ker)** is an equivalence relation on X^X.

Equivalence relations and mathematical logic

Equivalence relations are a ready source of examples or counterexamples. For example, an equivalence relation with exactly two infinite equivalence classes is an easy example of a theory which is ω-categorical, but not categorical for any larger cardinal number.

An implication of model theory is that the properties defining a relation can be proved independent of each other (and hence necessary parts of the definition) if and only if, for each property, examples can be found of relations not satisfying the given property while satisfying all the other properties. Hence the three defining properties of equivalence relations can be proved mutually independent by the following three examples:

- *Reflexive and transitive*: The relation ≤ on **N**. Or any preorder;
- *Symmetric and transitive*: The relation R on **N**, defined as $aRb \leftrightarrow ab \neq 0$. Or any partial equivalence relation;
- *Reflexive and symmetric*: The relation R on **Z**, defined as $aRb \leftrightarrow$ "$a - b$ is divisible by at least one of 2 or 3." Or any dependency relation.

Properties definable in first-order logic that an equivalence relation may or may not possess include:

- The number of equivalence classes is finite or infinite;
- The number of equivalence classes equals the (finite) natural number n;
- All equivalence classes have infinite cardinality;
- The number of elements in each equivalence class is the natural number n.

Euclidean relations

Euclid's *The Elements* includes the following "Common Notion 1":

> Things which equal the same thing also equal one another.

Nowadays, the property described by Common Notion 1 is called Euclidean (replacing "equal" by "are in relation with"). The following theorem connects Euclidean relations and equivalence relations:

Theorem. If a relation is Euclidean and reflexive, it is also symmetric and transitive.

Proof:

- $(aRc \wedge bRc) \to aRb$ [a/c] = $(aRa \wedge bRa) \to aRb$ [*reflexive*; erase $\mathbf{T}\wedge$] = $bRa \to aRb$. Hence R is symmetric.
- $(aRc \wedge bRc) \to aRb$ [*symmetry*] = $(aRc \wedge cRb) \to aRb$. Hence R is transitive. □

Hence an equivalence relation is a relation that is *Euclidean* and *reflexive*. *The Elements* mentions neither symmetry nor reflexivity, and Euclid probably would have deemed the reflexivity of equality too obvious to warrant explicit mention.

Notes

[1] Garrett Birkhoff and Saunders Mac Lane, 1999 (1967). *Algebra*, 3rd ed. p. 35, Th. 19. Chelsea.
[2] Wallace, D. A. R., 1998. *Groups, Rings and Fields*. p. 31, Th. 8. Springer-Verlag.
[3] Dummit, D. S., and Foote, R. M., 2004. *Abstract Algebra*, 3rd ed. p. 3, Prop. 2. John Wiley & Sons.
[4] Garrett Birkhoff and Saunders Mac Lane, 1999 (1967). *Algebra*, 3rd ed. p. 33, Th. 18. Chelsea.
[5] Rosen (2008), pp. 243-45. Less clear is §10.3 of Bas van Fraassen, 1989. *Laws and Symmetry*. Oxford Univ. Press.
[6] Wallace, D. A. R., 1998. *Groups, Rings and Fields*. Springer-Verlag: 202, Th. 6.
[7] Dummit, D. S., and Foote, R. M., 2004. *Abstract Algebra*, 3rd ed. John Wiley & Sons: 114, Prop. 2.
[8] Bas van Fraassen, 1989. *Laws and Symmetry*. Oxford Univ. Press: 246.
[9] Wallace, D. A. R., 1998. *Groups, Rings and Fields*. Springer-Verlag: 22, Th. 6.
[10] Wallace, D. A. R., 1998. *Groups, Rings and Fields*. Springer-Verlag: 24, Th. 7.
[11] Borceux, F. and Janelidze, G., 2001. *Galois theories*, Cambridge University Press, ISBN 0-521-80309-8

References

- Brown, Ronald, 2006. *Topology and Groupoids.* (http://www.bangor.ac.uk/r.brown/topgpds.html) Booksurge LLC. ISBN 1419627228.
- Castellani, E., 2003, "Symmetry and equivalence" in Brading, Katherine, and E. Castellani, eds., *Symmetries in Physics: Philosophical Reflections*. Cambridge Univ. Press: 422-433.
- Robert Dilworth and Crawley, Peter, 1973. *Algebraic Theory of Lattices*. Prentice Hall. Chpt. 12 discusses how equivalence relations arise in lattice theory.
- Higgins, P.J., 1971. *Categories and groupoids.* (http://www.emis.de/journals/TAC/reprints/articles/7/tr7abs.html) Van Nostrand. Downloadable since 2005 as a TAC Reprint.
- John Randolph Lucas, 1973. *A Treatise on Time and Space*. London: Methuen. Section 31.
- Rosen, Joseph (2008) *Symmetry Rules: How Science and Nature are Founded on Symmetry*. Springer-Verlag. Mostly chpts. 9,10.

External links

- Bogomolny, A., " Equivalence Relationship (http://www.cut-the-knot.org/blue/equi.shtml)" cut-the-knot. Accessed 1 September 2009
- Equivalence relation (http://planetmath.org/encyclopedia/EquivalenceClass2.html) at PlanetMath

Centralization

Centralisation, or **centralization** (see spelling differences), is the process by which the activities of an organisation, particularly those regarding planning decision-making, become concentrated within a particular location and/or group.

In political science, this refers to the concentration of a government's power - both geographically and politically, into a centralised government.

In neuroscience, centralization refers to the evolutionary trend of the nervous system to be partitioned into a central nervous system and peripheral nervous system.

In business studies centralisation and decentralisation is about where decisions are taken in the chain of command.

Clustering coefficient

WARNING: Article could not be rendered - ouputting plain text.
Potential causes of the problem are: (a) a bug in the pdf-writer software (b) problematic Mediawiki markup (c) table is too wide

In graph theory, a clustering coefficient is a measure of degree to which nodes in a graph tend to cluster together. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterised by a relatively high density of ties (Holland and Leinhardt, 1971P. W. Holland and S. Leinhardt (1998). "Transitivity in structural models of small groups". Comparative Group Studies 2: 107–124.; Watts and Strogatz, 1998D. J. Watts and Steven Strogatz (June 1998). "Collective dynamics of 'small-world' networks". Nature (journal)Nature 393 (6684): 440–442. doi:10.1038/30918. PMID 9623998. .). In real-world networks, this likelihood tends to be greater than the average probability of a tie randomly established between two nodes (Holland and Leinhardt, 1971; Watts and Strogatz, 1998).Two versions of this measure exist: the global and the local. The global version was designed to give an overall indication of the clustering in the network, whereas the local gives an indication of the embeddedness of single nodes. Global clustering coefficient The global clustering coefficient is based on triplets of nodes. A triplet is three nodes that are connected by either two (open triplet) or three (closed triplet) undirected ties. A triangle consists of three closed triplets, one centred on each of the nodes. The global clustering coefficient is the number of closed triplets (or 3 x triangles) over the total number of triplets (both open and closed). The first attempt to measure it was made by Luce and Perry (1949)R. D. Luce and A. D. Perry (1949). "A method of matrix analysis of group structure". Psychometrika 14 (1): 95–116. doi:10.1007/BF02289146. PMID 18152948.. This measure gives an indication of the clustering in the whole network (global), and can be applied to both undirected and directed networks (often called transitivity, see Wasserman and Faust, 1994, page 243Stanley Wasserman, Kathrine Faust, 1994. Social Network Analysis: Methods and Applications. Cambridge: Cambridge University Press.). Formally, it has been defined as: C = \frac{3 \times

\mbox{number of triangles}}{\mbox{number of connected triples of vertices}} = \frac{\mbox{number of closed triplets}}{\mbox{number of connected triples of vertices}}.A generalisation to weighted networks was proposed by Opsahl and Panzarasa (2009)Tore Opsahl and Pietro Panzarasa (2009). "Clustering in Weighted Networks". Social Networks 31 (2): 155–163. doi:10.1016/j.socnet.2009.02.002. ., and a redefinition to two-mode networks (both binary and weighted) by Opsahl (2009)Tore Opsahl (2009). "Clustering in Two-mode Networks". Conference and Workshop on Two-Mode Social Analysis (Sept 30-Oct 2, 2009). .. Local clustering coefficient Example local clustering coefficient on an undirected graph. The local clustering coefficient of the light blue node is computed as the proportion of connections among its neighbors which are actually realized compared with the number of all possible connections. In the figure, the light blue node has three neighbours, which can have a maximum of 3 connections among them. In the top part of the figure all three possible connections are realised (thick black segments), giving a local clustering coefficient of 1. In the middle part of the figure only one connection is realised (thick black line) and 2 connections are missing (dotted red lines), giving a local cluster coefficient of 1/3. Finally, none of the possible connections among the neighbours of the light blue node are realised, producing a local clustering coefficient value of 0. The local clustering coefficient of a vertex (graph theory)vertex in a Graph (mathematics)graph quantifies how close its Neighbourhood (graph theory)neighbors are to being a Clique (graph theory)clique (complete graph). Duncan J. Watts and Steven Strogatz introduced the measure in 1998 to determine whether a graph is a small-world network.A graph G=(V,E) formally consists of a set of vertices V and a set of edges E between them. An edge e_{ij} connects vertex i with vertex j. The Neighbourhood (graph theory)neighbourhood N for a vertex v_i is defined as its immediately connected neighbours as follows:N_i = \{v_j : e_{ij} \in E \and e_{ji} \in E\}.The degree (mathematics)degree k_i of a vertex is defined as the number of vertices, |N_i|, in its neighbourhood N_i. The local clustering coefficient C_i for a vertex v_i is then given by the proportion of links between the vertices within its neighbourhood divided by the number of links that could possibly exist between them. For a directed graph, e_{ij} is distinct from e_{ji}, and therefore for each neighbourhood N_i there are k_i(k_i-1) links that could exist among the vertices within the neighbourhood (k_i is the total (in + out) degree of the vertex). Thus, the local clustering coefficient for directed graphs is given asC_i = \frac{|\{e_{jk}\}|}{k_i(k_i-1)} : v_j,v_k \in N_i, e_{jk} \in E.An undirected graph has the property that e_{ij} and e_{ji} are considered identical. Therefore, if a vertex v_i has k_i neighbours, \frac{k_i(k_i-1)}{2} edges could exist among the vertices within the neighbourhood. Thus, the local clustering coefficient for undirected graphs can be defined asC_i = \frac{2|\{e_{jk}\}|}{k_i(k_i-1)} : v_j,v_k \in N_i, e_{jk} \in E.Let \lambda_G(v) be the number of triangles on v \in V(G) for undirected graph G. That is, \lambda_G(v) is the number of subgraphs of G with 3 edges and 3 vertices, one of which is v. Let \tau_G(v) be the number of triples on v \in G. That is, \tau_G(v) is the number of subgraphs (not necessarily induced) with 2 edges and 3 vertices, one of which is v and such that v is incident to both edges. Then we can also define the clustering coefficient as C_i = \frac{\lambda_G(v)}{\tau_G(v)}.It is simple to show that the two preceding definitions are the same, since \tau_G(v) = C({k_i},2) = \frac{1}{2}k_i(k_i-1).These measures are 1 if every neighbour connected to v_i is also connected to every other vertex within the neighbourhood, and 0 if no vertex that is connected to v_i connects to any other vertex that is connected to v_i. Network average clustering coefficient The clustering coefficient for the whole network is given by Watts and Strogatz as the average of the local clustering coefficients of all the vertices n : \bar{C} = \frac{1}{n}\sum_{i=1}^{n} C_i.A graph is considered Small-world networksmall-world, if its average clustering coefficient \bar{C} is significantly higher than a random graph constructed on the same vertex set, and if the graph has approximately the same distance (graph theory)mean-shortest path length as its corresponding random graph. A generalisation to weighted networks was proposed by Barrat et al. (2004)A. Barrat and M. Barthelemy and R. Pastor-Satorras and A. Vespignani (2004). "The architecture of complex weighted networks". Proceedings of the National Academy of Sciences 101 (11): 3747–3752. doi:10.1073/pnas.0400087101. PMID 15007165. PMC 374315., and a redefinition to bipartite graphs (also called two-mode networks) by Latapy et al. (2008)M. Latapy and C. Magnien and N. Del Vecchio (2008). "Basic Notions for the Analysis of Large Two-mode Networks". Social Networks 30 (1): 31–48. doi:10.1016/j.socnet.2007.04.006. and Opsahl (2009)Tore Opsahl (2009). "Clustering in Two-mode Networks".

Conference and Workshop on Two-Mode Social Analysis (Sept 30-Oct 2, 2009). .. This formula is not, by default, defined for graphs with isolated vertices; see Kaiser, (2008)Marcus kaiser (2008). "Mean clustering coefficients: the role of isolated nodes and leafs on clustering measures for small-world networks". New Journal of Physics 10 (8): 083042. doi:10.1088/1367-2630/10/8/083042. . and Barmpoutis et al. D.Barmpoutis and R.M. Murray (2010). "Networks with the Smallest Average Distance and the Largest Average Clustering". ArXiv Digital Library. . The networks with the largest possible average clustering coefficient are found to have a modular structure, and at the same time, they have the smallest possible average distance among the different nodes.References

Structural cohesion

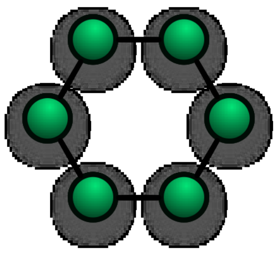
Structural cohesion is the sociological and graph theory conception [1] [2] and measurement of cohesion for maximal social group or graphical boundaries where related elements cannot be disconnected except by removal of a certain minimal number of other nodes. The solution to the boundary problem for structural cohesion is found by the vertex-cut version of Menger's theorem. The boundaries of structural endogamy are a special case of structural cohesion. It is also useful to know that k-cohesive graphs (or k-components) are always a subgraph of a k-core, although a k-core is not always k-cohesive. A k-core is simply a subgraph in which all nodes have at least k neighbors but it need not even be connected.

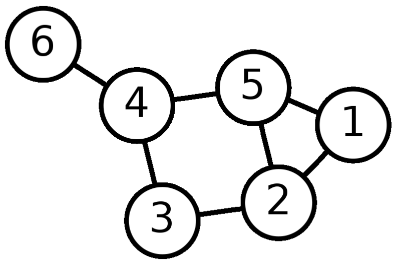
Software

Cohesive.blocking [3] is the R program for computing structural cohesion according to the Moody-White (2003) algorithm. This wiki site provides numerous examples and a tutorial for use with R.

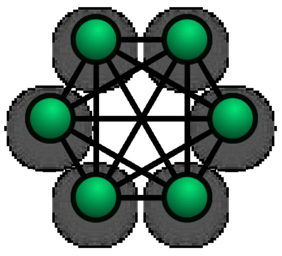
Examples

Some illustrative examples are presented in the gallery below:


The 6-node ring in the graph has connectivity-2 or a level 2 of structural cohesion because the removal of two nodes is needed to disconnect it.



The 6-node component (1-connected) has an embedded 2-component, nodes 1-5


A 6-node clique is a 5-component, structural cohesion 5

Perceived cohesion

Perceived Cohesion Scale (PCS) is a six item scale that is used to measure structural cohesion in groups. In 1990, Bollen and Hoyle used the PCS and applied it to a study of large groups which were used to assess the psychometric qualities of their scale.[3]

See also

- Community cohesion
- Social cohesion
- Social network
- Social-circles network model
- Perceived cohesion

References

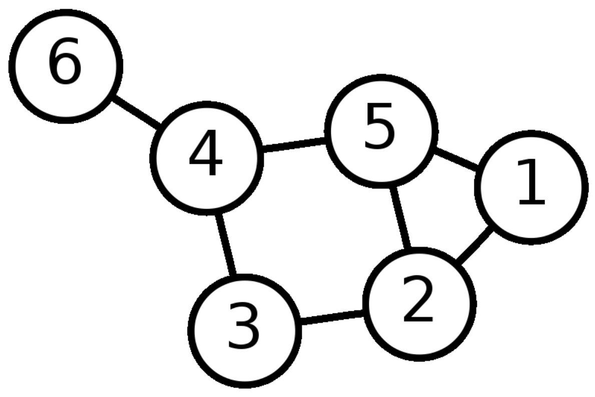
[1] Moody, James; White, Douglas (2003). "Structural Cohesion and Embeddedness: A Hierarchical Concept of Social Groups." (http://www2.asanet.org/journals/ASRFeb03MoodyWhite.pdf) (PDF). *American Sociological Review* **68** (1): 1–25. . Retrieved 2006-08-19.

[2] White, Douglas; Frank Harary (2001). "The Cohesiveness of Blocks in Social Networks: Node Connectivity and Conditional Density." (http://www.ingentaconnect.com/content/bpl/some/2001/00000031/00000001/art00098;jsessionid=1f96pkuejdhry.victoria) (book). *Sociological Methodology 2001* (Blackwell Publishers, Inc., Boston, USA and Oxford, UK.) **31** (1): 305–359. doi:10.1111/0081-1750.00098. . Retrieved 2006-08-19.

[3] Chin, Wynne W., et al. Perceived Cohesion: A Conceptual and Empirical Examination: Adapting and Testing the Perceived Cohesion Scale in a Small-Group Setting. (http://sgr.sagepub.com/cgi/content/abstract/30/6/751) 1999. Small Group Research 30(6):751-766.

Mathematics of Graphs

Graph (mathematics)



A drawing of a labeled graph on 6 vertices and 7 edges.

In mathematics, a **graph** is an abstract representation of a set of objects where some pairs of the objects are connected by links. The interconnected objects are represented by mathematical abstractions called *vertices*, and the links that connect some pairs of vertices are called *edges*. Typically, a graph is depicted in diagrammatic form as a set of dots for the vertices, joined by lines or curves for the edges. Graphs are one of the objects of study in discrete mathematics.

The edges may be directed (asymmetric) or undirected (symmetric). For example, if the vertices represent people at a party, and there is an edge between two people if they shake hands, then this is an undirected graph, because if person A shook hands with person B, then person B also shook hands with person A. On the other hand, if the vertices represent people at a party, and there is an edge from person A to person B when person A knows of person B, then this graph is directed, because knowing of someone is not necessarily a symmetric relation (that is, one person knowing of another person does not necessarily imply the reverse; for example, many fans may know of a celebrity, but the celebrity is unlikely to know of all their fans). This latter type of graph is called a *directed* graph and the edges are called *directed edges* or *arcs*; in contrast, a graph where the edges are not directed is called *undirected*.

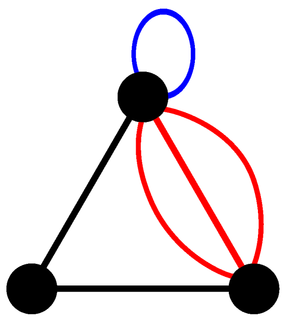
Vertices are also called *nodes* or *points*, and edges are also called *lines*. Graphs are the basic subject studied by graph theory. The word "graph" was first used in this sense by James Joseph Sylvester in 1878.[1]

Definitions

Definitions in graph theory vary. The following are some of the more basic ways of defining graphs and related mathematical structures.

Graph

In the most common sense of the term,[2] a **graph** is an ordered pair $G = (V, E)$ comprising a set V of **vertices** or **nodes** together with a set E of **edges** or **lines**, which are 2-element subsets of V (i.e., an edge is related with two vertices, and the relation is represented as unordered pair of the vertices with respect to the particular edge). To avoid ambiguity, this type of graph may be described precisely as undirected and simple**.**


A general example of a graph (actually, a pseudograph) with three vertices and six edges.

Other senses of *graph* stem from different conceptions of the edge set. In one more generalized notion,[3] E is a set together with a relation of **incidence** that associates with each edge two vertices. In another generalized notion, E is a multiset of unordered pairs of (not necessarily distinct) vertices. Many authors call this type of object a multigraph or pseudograph.

All of these variants and others are described more fully below.

The vertices belonging to an edge are called the **ends**, **endpoints**, or **end vertices** of the edge. A vertex may exist in a graph and not belong to an edge.

V and E are usually taken to be finite, and many of the well-known results are not true (or are rather different) for **infinite graphs** because many of the arguments fail in the infinite case. The **order** of a graph is $|V|$ (the number of vertices). A graph's **size** is $|E|$, the number of edges. The **degree** of a vertex is the number of edges that connect to it, where an edge that connects to the vertex at both ends (a loop) is counted twice.
For an edge $\{u, v\}$, graph theorists usually use the somewhat shorter notation uv.

Adjacency relation

The edges E of an undirected graph G induce a symmetric binary relation ~ on V that is called the **adjacency** relation of G. Specifically, for each edge $\{u, v\}$ the vertices u and v are said to be **adjacent** to one another, which is denoted $u \sim v$.

Types of graphs

Distinction in terms of the main definition

As stated above, in different contexts it may be useful to define the term *graph* with different degrees of generality. Whenever it is necessary to draw a strict distinction, the following terms are used. Most commonly, in modern texts in graph theory, unless stated otherwise, *graph* means "undirected simple finite graph" (see the definitions below).

Undirected graph

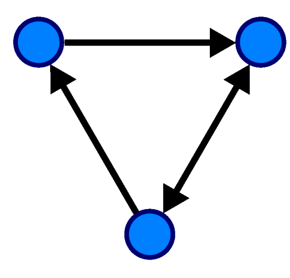
A graph in which edges have no orientation, i.e., they are not ordered pairs, but sets $\{u, v\}$ (or 2-multisets) of vertices.

Directed graph

A **directed graph** or **digraph** is an ordered pair $D = (V, A)$ with

- V a set whose elements are called **vertices** or **nodes**, and
- A a set of ordered pairs of vertices, called **arcs**, **directed edges**, or **arrows**.

An arc $a = (x, y)$ is considered to be directed **from** x **to** y; y is called the **head** and x is called the **tail** of the arc; y is said to be a **direct successor** of x, and x is said to be a **direct predecessor** of y. If a path leads from x to y, then y is said to be a **successor** of x and **reachable** from x, and x is said to be a **predecessor** of y. The arc (y, x) is called the arc (x, y) **inverted**.


A directed graph.

A directed graph D is called **symmetric** if, for every arc in D, the corresponding inverted arc also belongs to D. A symmetric loopless directed graph $D = (V, A)$ is equivalent to a simple undirected graph $G = (V, E)$, where the pairs of inverse arcs in A correspond 1-to-1 with the edges in E; thus the edges in G number $|E| = |A|/2$, or half the number of arcs in D.

A variation on this definition is the **oriented graph**, in which not more than one of (x, y) and (y, x) may be arcs.

Mixed graph

A **mixed graph** G is a graph in which some edges may be directed and some may be undirected. It is written as an ordered triple $G = (V, E, A)$ with V, E, and A defined as above. Directed and undirected graphs are special cases.

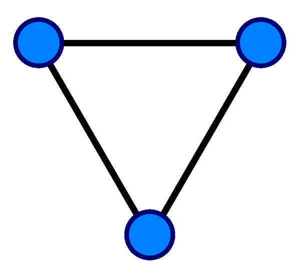
Multigraph

A loop is an edge (directed or undirected) which starts and ends on the same vertex; these may be permitted or not permitted according to the application. In this context, an edge with two different ends is called a **link**.

The term "multigraph" is generally understood to mean that multiple edges (and sometimes loops) are allowed. Where graphs are defined so as to *allow* loops and multiple edges, a multigraph is often defined to mean a graph *without* loops,[4] however, where graphs are defined so as to *disallow* loops and multiple edges, the term is often defined to mean a "graph" which can have both multiple edges *and* loops,[5] although many use the term "pseudograph" for this meaning.[6]

Simple graph

As opposed to a multigraph, a simple graph is an undirected graph that has no loops and no more than one edge between any two different vertices. In a simple graph the edges of the graph form a set (rather than a multiset) and each edge is a pair of *distinct* vertices. In a simple graph with n vertices every vertex has a degree that is less than n (the converse, however, is not true - there exist non-simple graphs with n vertices in which every vertex has a degree smaller than n).


A simple graph with three vertices and three edges. Each vertex has degree two, so this is also a regular graph.

Weighted graph

A graph is a weighted graph if a number (weight) is assigned to each edge. Such weights might represent, for example, costs, lengths or capacities, etc. depending on the problem.

The weight of the graph is the sum of the weights given to all edges.

Half-edges, loose edges

In exceptional situations it is even necessary to have edges with only one end, called **half-edges**, or no ends (**loose edges**); see for example signed graphs and biased graphs.

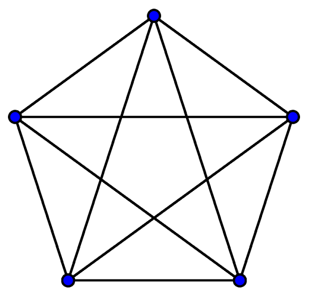
Important graph classes

Regular graph

A regular graph is a graph where each vertex has the same number of neighbors, i.e., every vertex has the same degree or valency. A regular graph with vertices of degree *k* is called a *k*-regular graph or regular graph of degree *k*.

Complete graph

Complete graphs have the feature that each pair of vertices has an edge connecting them.


A complete graph with 5 vertices. Each vertex has an edge to every other vertex.

Finite and infinite graphs

A finite graph is a graph $G = (V, E)$ such that V and E are finite sets. An infinite graph is one with an infinite set of vertices or edges or both.

Most commonly in graph theory it is implied that the graphs discussed are finite. If the graphs are infinite, that is usually specifically stated.

Graph classes in terms of connectivity

In an undirected graph *G*, two vertices *u* and *v* are called **connected** if *G* contains a path from *u* to *v*. Otherwise, they are called **disconnected**. A graph is called **connected** if every pair of distinct vertices in the graph is connected; otherwise, it is called **disconnected**.

A graph is called ***k*-vertex-connected** or ***k*-edge-connected** if no set of *k-1* vertices (respectively, edges) exists that disconnects the graph. A *k*-vertex-connected graph is often called simply ***k*-connected**.

A directed graph is called **weakly connected** if replacing all of its directed edges with undirected edges produces a connected (undirected) graph. It is **strongly connected** or **strong** if it contains a directed path from *u* to *v* and a directed path from *v* to *u* for every pair of vertices *u*, *v*.

Properties of graphs

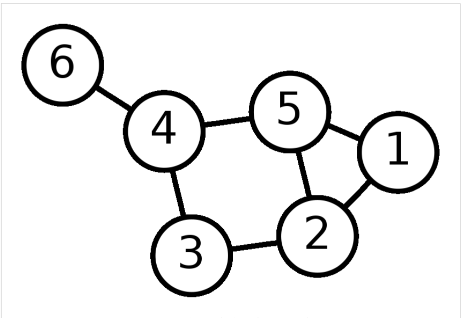
Two edges of a graph are called **adjacent** (sometimes **coincident**) if they share a common vertex. Two arrows of a directed graph are called **consecutive** if the head of the first one is at the nock (notch end) of the second one. Similarly, two vertices are called **adjacent** if they share a common edge (**consecutive** if they are at the notch and at the head of an arrow), in which case the common edge is said to **join** the two vertices. An edge and a vertex on that edge are called **incident**.

The graph with only one vertex and no edges is called the **trivial graph**. A graph with only vertices and no edges is known as an **edgeless graph**. The graph with no vertices and no edges is sometimes called the **null graph** or **empty graph**, but the terminology is not consistent and not all mathematicians allow this object.

In a **weighted** graph or digraph, each edge is associated with some value, variously called its *cost*, *weight*, *length* or other term depending on the application; such graphs arise in many contexts, for example in optimal routing problems such as the traveling salesman problem.

Normally, the vertices of a graph, by their nature as elements of a set, are distinguishable. This kind of graph may be called **vertex-labeled**. However, for many questions it is better to treat vertices as indistinguishable; then the graph may be called **unlabeled**. (Of course, the vertices may be still distinguishable by the properties of the graph itself, e.g., by the numbers of incident edges). The same remarks apply to edges, so that graphs which have labeled edges are called **edge-labeled** graphs. Graphs with labels attached to edges or vertices are more generally designated as **labeled**. Consequently, graphs in which vertices are indistinguishable and edges are indistinguishable are called *unlabeled*. (Note that in the literature the term *labeled* may apply to other kinds of labeling, besides that which serves only to distinguish different vertices or edges.)

Examples



A graph with six nodes.

- The diagram at right is a graphic representation of the following graph:

  $V = \{1, 2, 3, 4, 5, 6\}$

  $E =$.
- In category theory a small category can be represented by a directed multigraph in which the objects of the category represented as vertices and the morphisms as directed edges. Then, the functors between categories induce some, but not necessarily all, of the digraph morphisms of the graph.
- In computer science, directed graphs are used to represent knowledge (e.g., Conceptual graph), finite state machines, and many other discrete structures.
- A binary relation R on a set X defines a directed graph. An element x of X is a direct predecessor of an element y of X iff xRy.

Important graphs

Basic examples are:

- In a complete graph, each pair of vertices is joined by an edge; that is, the graph contains all possible edges.
- In a bipartite graph, the vertex set can be partitioned into two sets, W and X, so that no two vertices in W are adjacent and no two vertices in X are adjacent. Alternatively, it is a graph with a chromatic number of 2.
- In a complete bipartite graph, the vertex set is the union of two disjoint sets, W and X, so that every vertex in W is adjacent to every vertex in X but there are no edges within W or X.
- In a *linear graph* or path graph of length n, the vertices can be listed in order, $v_0, v_1, ..., v_n$, so that the edges are $v_{i-1}v_i$ for each $i = 1, 2, ..., n$. If a linear graph occurs as a subgraph of another graph, it is a path in that graph.
- In a cycle graph of length $n \geq 3$, vertices can be named $v_1, ..., v_n$ so that the edges are $v_{i-1}v_i$ for each $i = 2,...,n$ in addition to $v_n v_1$. Cycle graphs can be characterized as connected 2-regular graphs. If a cycle graph occurs as a subgraph of another graph, it is a *cycle* or *circuit* in that graph.
- A planar graph is a graph whose vertices and edges can be drawn in a plane such that no two of the edges intersect (i.e., *embedded* in a plane).
- A tree is a connected graph with no cycles.
- A *forest* is a graph with no cycles (i.e. the disjoint union of one or more *trees*).

More advanced kinds of graphs are:

- The Petersen graph and its generalizations
- Perfect graphs
- Cographs
- Other graphs with large automorphism groups: vertex-transitive, arc-transitive, and distance-transitive graphs.
- Strongly regular graphs and their generalization distance-regular graphs.

Operations on graphs

There are several operations that produce new graphs from old ones, which might be classified into the following categories:

- Elementary operations, sometimes called "editing operations" on graphs, which create a new graph from the original one by a simple, local change, such as addition or deletion of a vertex or an edge, merging and splitting of vertices, etc.
- Graph rewrite operations replacing the occurrence of some pattern graph within the host graph by an instance of the corresponding replacement graph.
- Unary operations, which create a significantly new graph from the old one. Examples:
  - Line graph
  - Dual graph
  - Complement graph
- Binary operations, which create new graph from two initial graphs. Examples:
  - Disjoint union of graphs
  - Cartesian product of graphs
  - Tensor product of graphs
  - Strong product of graphs
  - Lexicographic product of graphs

Generalizations

In a hypergraph, an edge can join more than two vertices.

An undirected graph can be seen as a simplicial complex consisting of 1-simplices (the edges) and 0-simplices (the vertices). As such, complexes are generalizations of graphs since they allow for higher-dimensional simplices.

Every graph gives rise to a matroid.

In model theory, a graph is just a structure. But in that case, there is no limitation on the number of edges: it can be any cardinal number, see continuous graph.

In computational biology, power graph analysis introduces power graphs as an alternative representation of undirected graphs.

Notes

[1] Gross, Jonathan L.; Yellen, Jay (2004). *Handbook of graph theory* (http://books.google.com/?id=mKkIGIea_BkC). CRC Press. p. 35 (http://books.google.com/books?id=mKkIGIea_BkC&pg=PA35&lpg=PA35). ISBN 9781584880905.

[2] See, for instance, Iyanaga and Kawada, **69 J**, p. 234 or Biggs, p. 4.

[3] See, for instance, Graham et al., p. 5.

[4] For example, see Balakrishnan, p. 1, Gross (2003), p. 4, and Zwillinger, p. 220.

[5] For example, see. Bollobas, p. 7 and Diestel, p. 25.

[6] Gross (1998), p. 3, Gross (2003), p. 205, Harary, p.10, and Zwillinger, p. 220.

References

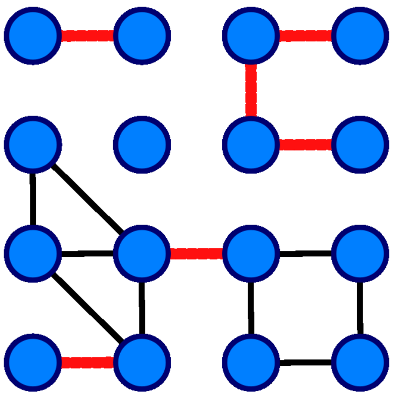
- Balakrishnan, V. K. (1997-02-01). *Graph Theory* (1st ed.). McGraw-Hill. ISBN 0-07-005489-4.
- Berge, Claude (1958) (in French). *Théorie des graphes et ses applications*. Dunod, Paris: Collection Universitaire de Mathématiques, II. pp. viii+277. Translation: . Dover, New York: Wiley. 2001 [1962].
- Biggs, Norman (1993). *Algebraic Graph Theory* (2nd ed.). Cambridge University Press. ISBN 0-521-45897-8.
- Bollobas, Bela (2002-08-12). *Modern Graph Theory* (1st ed.). Springer. ISBN 0-387-98488-7.
- Bang-Jensen, J.; Gutin, G. (2000). *Digraphs: Theory, Algorithms and Applications* (http://www.cs.rhul.ac.uk/books/dbook/). Springer.
- Diestel, Reinhard (2005). *Graph Theory* (http://diestel-graph-theory.com/GrTh.html) (3rd ed.). Berlin, New York: Springer-Verlag. ISBN 978-3-540-26183-4.
- Graham, R.L., Grötschel, M., and Lovász, L, ed (1995). *Handbook of Combinatorics*. MIT Press. ISBN 0-262-07169-X.
- Gross, Jonathan L.; Yellen, Jay (1998-12-30). *Graph Theory and Its Applications*. CRC Press. ISBN 0-8493-3982-0.
- Gross, Jonathan L., & Yellen, Jay, ed (2003-12-29). *Handbook of Graph Theory*. CRC. ISBN 1-58488-090-2.
- Harary, Frank (January 1995). *Graph Theory*. Addison Wesley Publishing Company. ISBN 0-201-41033-8.
- Iyanaga, Shôkichi; Kawada, Yukiyosi (1977). *Encyclopedic Dictionary of Mathematics*. MIT Press. ISBN 0-262-09016-3.
- Zwillinger, Daniel (2002-11-27). *CRC Standard Mathematical Tables and Formulae* (31st ed.). Chapman & Hall/CRC. ISBN 1-58488-291-3.

External links

- Graph theory tutorial (http://www.utm.edu/departments/math/graph/)
- Image gallery : Some real-life graphs (http://www.aisee.com/graphs/)
- A searchable database of small connected graphs (http://www.gfredericks.com/main/sandbox/graphs)
- VisualComplexity.com (http://www.visualcomplexity.com) — A visual exploration on mapping complex networks
- Weisstein, Eric W., " Graph (http://mathworld.wolfram.com/Graph.html)" from MathWorld.
- Intelligent Graph Visualizer (https://sourceforge.net/projects/igv-intelligent/) — IGV create and edit graph, automatically places graph, search shortest path (+coloring vertices), center, degree, eccentricity, etc.

Bridge (graph theory)

In graph theory, a **bridge** (also known as a cut-edge or cut arc or an isthmus) is an edge whose deletion increases the number of connected components. Equivalently, an edge is a bridge if and only if it is not contained in any cycle.

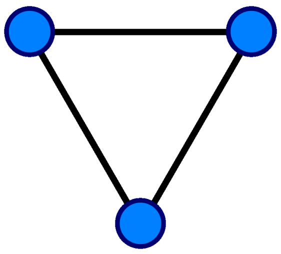

A graph with 6 bridges (highlighted in red)

A graph is said to be **bridgeless** if it contains no bridges. It is easy to see that this is equivalent to 2-edge-connectivity of each nontrivial component.

Cycle double cover conjecture

An important open problem involving bridges is the cycle double cover conjecture, due to Seymour and Szekeres (1978 and 1979, independently), which states that every bridgeless graph admits a set of cycles which contains each edge exactly twice.[1]

Bridge-Finding Algorithm


An undirected connected graph with no cut edges

An $O(|V|+|E|)$ algorithm for finding bridges in a connected graph was found by Tarjan in 1974.[2]

Definitions: A non-tree edge between v and w is denoted by $v--w$. An in-tree edge with v as the parent is denoted by $v \longrightarrow w$.

$ND(v) = 1 + \sum_{v \longrightarrow w} ND(w)$ where v is the parent node of w.

$L(v) = \min(\{v - ND(v) + 1\} \cup \{L(w) \mid v \rightarrow w\} \cup \{w \mid v--w\})$

L detects connections to nodes further left or further down in the tree.

$H(v) = \max(\{v\} \cup \{H(w) \mid v \rightarrow w\} \cup \{w \mid v--w\})$

H detects connections to nodes further right or further up in the tree.

Algorithm:

1. Find a spanning tree of G
2. Create a rooted tree T from the spanning tree
3. Traverse the tree T in postorder and number the nodes. Parent nodes in the tree now have higher numbers than child nodes.
4. for each node from 1 to v_1 (the root node of the tree) do:
    1. Compute the number of descendants $ND(v)$ for this node.
    2. Compute $L(v)$ and $H(v)$
    3. for each w such that $v \longrightarrow w$: if $H(w) \leq w$ and $L(w) > w - ND(w)$ then (v, w) is a bridge.

Cut arc in trees

An edge or arc e = uv of a tree G is a cut arc of G if and only if the degree of the vertices u and v are greater than 1. Cut arcs are also defined for directed graphs [3]

Notes

[1] http://www.cems.uvm.edu/%7Earchdeac/problems/cyclecov.htm

[2] "A note on finding the bridges of a graph", Robert Endre Tarjan, Information Processing Letters, April 1974 pp160-161.

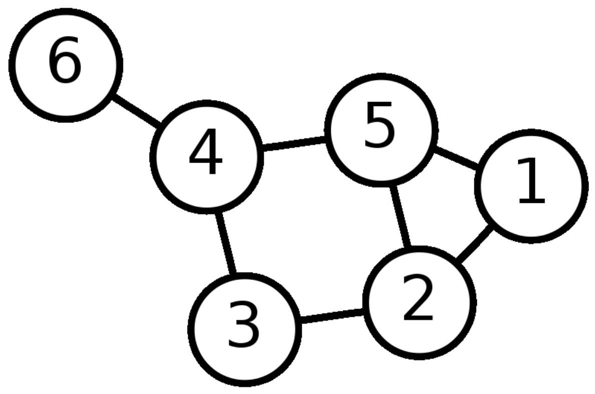
[3] Rao, S.B.; Ramachandra Rao, A. The number of cut vertices and cut arcs in a strong directed graph. (English) Acta Math. Acad. Sci. Hung. 22, 411-421 (1972).

References

- Béla Bollobás, *Modern graph theory*, GTM **184**, Springer Verlag, 1998. Page 6.
- Tarjan, Robert Endre. "A note on finding the bridges of a graph". *Information Processing Letters* **2** (6): 160–161. doi:10.1016/0020-0190(74)90003-9.

Graph theory

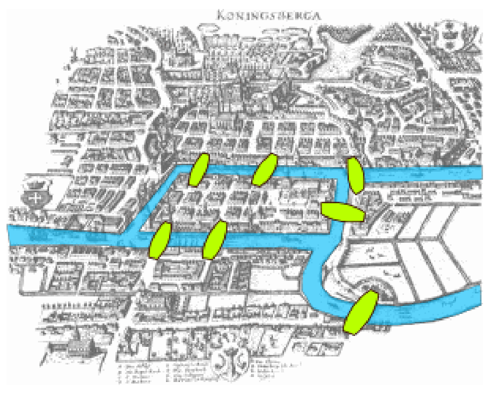
In mathematics and computer science, **graph theory** is the study of *graphs*: mathematical structures used to model pairwise relations between objects from a certain collection. A "graph" in this context refers to a collection of vertices or 'nodes' and a collection of *edges* that connect pairs of vertices. A graph may be *undirected*, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be *directed* from one vertex to another; see graph (mathematics) for more detailed definitions and for other variations in the types of graphs that are commonly considered. The graphs studied in graph theory should not be confused with "graphs of functions" and other kinds of graphs.



A drawing of a graph

Graphs are one of the prime objects of study in Discrete Mathematics. Refer to Glossary of graph theory for basic definitions in graph theory.

History



The Königsberg Bridge problem

The paper written by Leonhard Euler on the *Seven Bridges of Königsberg* and published in 1736 is regarded as the first paper in the history of graph theory.[1] This paper, as well as the one written by Vandermonde on the *knight problem,* carried on with the *analysis situs* initiated by Leibniz. Euler's formula relating the number of edges, vertices, and faces of a convex polyhedron was studied and generalized by Cauchy[2] and L'Huillier,[3] and is at the origin of topology.

More than one century after Euler's paper on the bridges of Königsberg and while Listing introduced topology, Cayley was led by the study of particular analytical forms arising from differential calculus to study a particular class of graphs, the *trees*. This study had many implications in theoretical chemistry. The involved techniques mainly concerned the enumeration of graphs having particular properties. Enumerative graph theory then rose from the results of Cayley and the fundamental results published by Pólya between 1935 and 1937 and the generalization of these by De Bruijn in 1959. Cayley linked his results on trees with the contemporary studies of chemical composition.[4] The fusion of the ideas coming from mathematics with those coming from chemistry is at the origin of a part of the standard terminology of graph theory.

In particular, the term "graph" was introduced by Sylvester in a paper published in 1878 in *Nature*, where he draws an analogy between "quantic invariants" and "co-variants" of algebra and molecular diagrams:[5]

> "[...] Every invariant and co-variant thus becomes expressible by a *graph* precisely identical with a Kekuléan diagram or chemicograph. [...] I give a rule for the geometrical multiplication of graphs, *i.e.* for constructing a *graph* to the product of in- or co-variants whose separate graphs are given. [...]" (italics as in the original).

One of the most famous and productive problems of graph theory is the four color problem: "Is it true that any map drawn in the plane may have its regions colored with four colors, in such a way that any two regions having a common border have different colors?" This problem was first posed by Francis Guthrie in 1852 and its first written record is in a letter of De Morgan addressed to Hamilton the same year. Many incorrect proofs have been proposed, including those by Cayley, Kempe, and others. The study and the generalization of this problem by Tait, Heawood, Ramsey and Hadwiger led to the study of the colorings of the graphs embedded on surfaces with arbitrary genus. Tait's reformulation generated a new class of problems, the *factorization problems*, particularly studied by Petersen and Kőnig. The works of Ramsey on colorations and more specially the results obtained by Turán in 1941 was at the origin of another branch of graph theory, *extremal graph theory*.

The four color problem remained unsolved for more than a century. In 1969 Heinrich Heesch published a method for solving the problem using computers.[6] A computer-aided proof produced in 1976 by Kenneth Appel and Wolfgang Haken makes fundamental use of the notion of "discharging" developed by Heesch.[7] [8] The proof involved checking the properties of 1,936 configurations by computer, and was not fully accepted at the time due to its complexity. A simpler proof considering only 633 configurations was given twenty years later by Robertson, Seymour, Sanders and Thomas.[9]

The autonomous development of topology from 1860 and 1930 fertilized graph theory back through the works of Jordan, Kuratowski and Whitney. Another important factor of common development of graph theory and topology came from the use of the techniques of modern algebra. The first example of such a use comes from the work of the physicist Gustav Kirchhoff, who published in 1845 his Kirchhoff's circuit laws for calculating the voltage and current in electric circuits.

The introduction of probabilistic methods in graph theory, especially in the study of Erdős and Rényi of the asymptotic probability of graph connectivity, gave rise to yet another branch, known as *random graph theory*, which has been a fruitful source of graph-theoretic results.

Drawing graphs

Graphs are represented graphically by drawing a dot for every vertex, and drawing an arc between two vertices if they are connected by an edge. If the graph is directed, the direction is indicated by drawing an arrow.

A graph drawing should not be confused with the graph itself (the abstract, non-visual structure) as there are several ways to structure the graph drawing. All that matters is which vertices are connected to which others by how many edges and not the exact layout. In practice it is often difficult to decide if two drawings represent the same graph. Depending on the problem domain some layouts may be better suited and easier to understand than others.

Graph-theoretic data structures

There are different ways to store graphs in a computer system. The data structure used depends on both the graph structure and the algorithm used for manipulating the graph. Theoretically one can distinguish between list and matrix structures but in concrete applications the best structure is often a combination of both. List structures are often preferred for sparse graphs as they have smaller memory requirements. Matrix structures on the other hand provide faster access for some applications but can consume huge amounts of memory.

List structures

Incidence list

The edges are represented by an array containing pairs (tuples if directed) of vertices (that the edge connects) and possibly weight and other data. Vertices connected by an edge are said to be *adjacent*.

Adjacency list

Much like the incidence list, each vertex has a list of which vertices it is adjacent to. This causes redundancy in an undirected graph: for example, if vertices A and B are adjacent, A's adjacency list contains B, while B's list contains A. Adjacency queries are faster, at the cost of extra storage space.

Matrix structures

Incidence matrix

The graph is represented by a matrix of size $|V|$ (number of vertices) by $|E|$ (number of edges) where the entry [vertex, edge] contains the edge's endpoint data (simplest case: 1 - incident, 0 - not incident).

Adjacency matrix

This is an n by n matrix A, where n is the number of vertices in the graph. If there is an edge from a vertex x to a vertex y, then the element $a_{x,y}$ is 1 (or in general the number of xy edges), otherwise it is 0. In computing, this matrix makes it easy to find subgraphs, and to reverse a directed graph.

Laplacian matrix or Kirchhoff matrix or Admittance matrix

This is defined as $D - A$, where D is the diagonal degree matrix. It explicitly contains both adjacency information and degree information. (However, there are other, similar matrices that are also called "Laplacian matrices" of a graph.)

Distance matrix

A symmetric n by n matrix D whose element $d_{x,y}$ is the length of a shortest path between x and y; if there is no such path $d_{x,y}$ = infinity. It can be derived from powers of A

$$d_{x,y} = \min\{n \mid A^n[x,y] \neq 0\}.$$

Problems in graph theory

Enumeration

There is a large literature on graphical enumeration: the problem of counting graphs meeting specified conditions. Some of this work is found in Harary and Palmer (1973).

Subgraphs, induced subgraphs, and minors

A common problem, called the subgraph isomorphism problem, is finding a fixed graph as a subgraph in a given graph. One reason to be interested in such a question is that many graph properties are *hereditary* for subgraphs, which means that a graph has the property if and only if all subgraphs have it too. Unfortunately, finding maximal subgraphs of a certain kind is often an NP-complete problem.

- Finding the largest complete graph is called the clique problem (NP-complete).

A similar problem is finding induced subgraphs in a given graph. Again, some important graph properties are hereditary with respect to induced subgraphs, which means that a graph has a property if and only if all induced subgraphs also have it. Finding maximal induced subgraphs of a certain kind is also often NP-complete. For example,

- Finding the largest edgeless induced subgraph, or independent set, called the independent set problem (NP-complete).

Still another such problem, the *minor containment problem*, is to find a fixed graph as a minor of a given graph. A minor or **subcontraction** of a graph is any graph obtained by taking a subgraph and contracting some (or no) edges. Many graph properties are hereditary for minors, which means that a graph has a property if and only if all minors have it too. A famous example:

- A graph is planar if it contains as a minor neither the complete bipartite graph $K_{3,3}$(See the Three-cottage problem) nor the complete graph K_5.

Another class of problems has to do with the extent to which various species and generalizations of graphs are determined by their *point-deleted subgraphs*, for example:

- The reconstruction conjecture

Graph coloring

Many problems have to do with various ways of coloring graphs, for example:

- The four-color theorem
- The strong perfect graph theorem
- The Erdős–Faber–Lovász conjecture (unsolved)
- The total coloring conjecture (unsolved)
- The list coloring conjecture (unsolved)
- The Hadwiger conjecture (graph theory) (unsolved)

Route problems

- Hamiltonian path and cycle problems
- Minimum spanning tree
- Route inspection problem (also called the "Chinese Postman Problem")
- Seven Bridges of Königsberg
- Shortest path problem
- Steiner tree
- Three-cottage problem
- Traveling salesman problem (NP-complete)

Network flow

There are numerous problems arising especially from applications that have to do with various notions of flows in networks, for example:

- Max flow min cut theorem

Visibility graph problems

- Museum guard problem

Covering problems

Covering problems are specific instances of subgraph-finding problems, and they tend to be closely related to the clique problem or the independent set problem.

- Set cover problem
- Vertex cover problem

Graph classes

Many problems involve characterizing the members of various classes of graphs. Overlapping significantly with other types in this list, this type of problem includes, for instance:

- Enumerating the members of a class
- Characterizing a class in terms of forbidden substructures
- Ascertaining relationships among classes (e.g., does one property of graphs imply another)
- Finding efficient algorithms to decide membership in a class
- Finding representations for members of a class

Applications

Graphs are among the most ubiquitous models of both natural and human-made structures. They can be used to model many types of relations and process dynamics in physical, biological and social systems. Many problems of practical interest can be represented by graphs.

In computer science, graphs are used to represent networks of communication, data organization, computational devices, the flow of computation, etc. One practical example: The link structure of a website could be represented by a directed graph. The vertices are the web pages available at the website and a directed edge from page *A* to page *B* exists if and only if *A* contains a link to *B*. A similar approach can be taken to problems in travel, biology, computer chip design, and many other fields. The development of algorithms to handle graphs is therefore of major interest in computer science. There, the transformation of graphs is often formalized and represented by graph rewrite systems. They are either directly used or properties of the rewrite systems (e.g. confluence) are studied. Complementary to graph transformation systems focussing on rule-based in-memory manipulation of graphs are graph databases geared towards transaction-safe, persistent storing and querying of graph-structured data.

Graph-theoretic methods, in various forms, have proven particularly useful in linguistics, since natural language often lends itself well to discrete structure. Traditionally, syntax and compositional semantics follow tree-based structures, whose expressive power lies in the Principle of Compositionality, modeled in a hierarchical graph. Within lexical semantics, especially as applied to computers, modeling word meaning is easier when a given word is understood in terms of related words; semantic networks are therefore important in computational linguistics. Still other methods in phonology (e.g. Optimality Theory, which uses lattice graphs) and morphology (e.g. finite-state morphology, using finite-state transducers) are common in the analysis of language as a graph. Indeed, the usefulness of this area of mathematics to linguistics has borne organizations such as TextGraphs [10], as well as various 'Net' projects, such as WordNet, VerbNet, and others.

Graph theory is also used to study molecules in chemistry and physics. In condensed matter physics, the three dimensional structure of complicated simulated atomic structures can be studied quantitatively by gathering statistics on graph-theoretic properties related to the topology of the atoms. For example, Franzblau's shortest-path (SP) rings. In chemistry a graph makes a natural model for a molecule, where vertices represent atoms and edges bonds. This approach is especially used in computer processing of molecular structures, ranging from chemical editors to database searching. In statistical physics, graphs can represent local connections between interacting parts of a system, as well as the dynamics of a physical process on such systems.

Graph theory is also widely used in sociology as a way, for example, to measure actors' prestige or to explore diffusion mechanisms, notably through the use of social network analysis software.

Likewise, graph theory is useful in biology and conservation efforts where a vertex can represent regions where certain species exist (or habitats) and the edges represent migration paths, or movement between the regions. This information is important when looking at breeding patterns or tracking the spread of disease, parasites or how changes to the movement can affect other species.

In mathematics, graphs are useful in geometry and certain parts of topology, e.g. Knot Theory. Algebraic graph theory has close links with group theory.

A graph structure can be extended by assigning a weight to each edge of the graph. Graphs with weights, or weighted graphs, are used to represent structures in which pairwise connections have some numerical values. For example if a graph represents a road network, the weights could represent the length of each road.

A digraph with weighted edges in the context of graph theory is called a network. Network analysis have many practical applications, for example, to model and analyze traffic networks. Applications of network analysis split broadly into three categories:

1. First, analysis to determine structural properties of a network, such as the distribution of vertex degrees and the diameter of the graph. A vast number of graph measures exist, and the production of useful ones for various domains remains an active area of research.
2. Second, analysis to find a measurable quantity within the network, for example, for a transportation network, the level of vehicular flow within any portion of it.
3. Third, analysis of dynamical properties of networks.

Notes

[1] Biggs, N.; Lloyd, E. and Wilson, R. (1986), *Graph Theory, 1736-1936*, Oxford University Press

[2] Cauchy, A.L. (1813), "Recherche sur les polyèdres - premier mémoire", *Journal de l'Ecole Polytechnique* **9 (Cahier 16)**: 66–86.

[3] L'Huillier, S.-A.-J. (1861), "Mémoire sur la polyèdrométrie", *Annales de Mathématiques* **3**: 169–189.

[4] Cayley, A. (1875), "Ueber die Analytischen Figuren, welche in der Mathematik Bäume genannt werden und ihre Anwendung auf die Theorie chemischer Verbindungen", *Berichte der deutschen Chemischen Gesellschaft* **8**: 1056–1059, doi:10.1002/cber.18750080252.

[5] John Joseph Sylvester (1878), *Chemistry and Algebra*. Nature, volume 17, page 284. doi:10.1038/017284a0. Online version (http://www.archive.org/stream/nature15unkngoog#page/n312/mode/1up) accessed on 2009-12-30.

[6] Heinrich Heesch: Untersuchungen zum Vierfarbenproblem. Mannheim: Bibliographisches Institut 1969.

[7] Appel, K. and Haken, W. (1977), "Every planar map is four colorable. Part I. Discharging", *Illinois J. Math.* **21**: 429–490.

[8] Appel, K. and Haken, W. (1977), "Every planar map is four colorable. Part II. Reducibility", *Illinois J. Math.* **21**: 491–567.

[9] Robertson, N.; Sanders, D.; Seymour, P. and Thomas, R. (1997), "The four color theorem", *Journal of Combinatorial Theory Series B* **70**: 2–44, doi:10.1006/jctb.1997.1750.

References

- Berge, Claude (1958), *Théorie des graphes et ses applications*, Collection Universitaire de Mathématiques, **II**, Paris: Dunod. English edition, Wiley 1961; Methuen & Co, New York 1962; Russian, Moscow 1961; Spanish, Mexico 1962; Roumanian, Bucharest 1969; Chinese, Shanghai 1963; Second printing of the 1962 first English edition, Dover, New York 2001.
- Biggs, N.; Lloyd, E.; Wilson, R. (1986), *Graph Theory, 1736–1936*, Oxford University Press.
- Bondy, J.A.; Murty, U.S.R. (2008), *Graph Theory*, Springer, ISBN 978-1-84628-969-9.
- Chartrand, Gary (1985), *Introductory Graph Theory*, Dover, ISBN 0-486-24775-9.
- Gibbons, Alan (1985), *Algorithmic Graph Theory*, Cambridge University Press.
- Golumbic, Martin (1980), *Algorithmic Graph Theory and Perfect Graphs*, Academic Press.
- Harary, Frank (1969), *Graph Theory*, Reading, MA: Addison-Wesley.
- Harary, Frank; Palmer, Edgar M. (1973), *Graphical Enumeration*, New York, NY: Academic Press.
- Mahadev, N.V.R.; Peled, Uri N. (1995), *Threshold Graphs and Related Topics*, North-Holland.

External links

Online textbooks

- Graph Theory with Applications (http://www.ecp6.jussieu.fr/pageperso/bondy/books/gtwa/gtwa.html) (1976) by Bondy and Murty
- Phase Transitions in Combinatorial Optimization Problems, Section 3: Introduction to Graphs (http://arxiv.org/pdf/cond-mat/0602129) (2006) by Hartmann and Weigt
- Digraphs: Theory Algorithms and Applications (http://www.cs.rhul.ac.uk/books/dbook/) 2007 by Jorgen Bang-Jensen and Gregory Gutin
- Graph Theory, by Reinhard Diestel (http://diestel-graph-theory.com/index.html)

Other resources

- Graph theory tutorial (http://www.utm.edu/departments/math/graph/)
- A searchable database of small connected graphs (http://www.gfredericks.com/main/sandbox/graphs)
- Image gallery: graphs (http://web.archive.org/web/20060206155001/http://www.nd.edu/~networks/gallery.htm)
- Concise, annotated list of graph theory resources for researchers (http://www.babelgraph.org/links.html)

Network theory

For the sociological theory, see Social network

Network theory is an area of computer science and network science and part of graph theory. It has application in many disciplines including particle physics, computer science, biology, economics, operations research, and sociology. Network theory concerns itself with the study of graphs as a representation of either symmetric relations or, more generally, of asymmetric relations between discrete objects. Applications of network theory include logistical networks, the World Wide Web, gene regulatory networks, metabolic networks, social networks, epistemological networks, etc. See list of network theory topics for more examples.

Network optimization

Network problems that involve finding an optimal way of doing something are studied under the name of combinatorial optimization. Examples include network flow, shortest path problem, transport problem, transshipment problem, location problem, matching problem, assignment problem, packing problem, routing problem, Critical Path Analysis and PERT (Program Evaluation & Review Technique).

Network analysis

Social network analysis

Social network analysis maps relationships between individuals in social networks.[1] Such individuals are often persons, but may be groups (including cliques and cohesive blocks [3]), organizations, nation states, web sites, or citations between scholarly publications (scientometrics).

Network analysis, and its close cousin traffic analysis, has significant use in intelligence. By monitoring the communication patterns between the network nodes, its structure can be established. This can be used for uncovering insurgent networks of both hierarchical and leaderless nature.

Biological network analysis

With the recent explosion of publicly available high throughput biological data, the analysis of molecular networks has gained significant interest. The type of analysis in this content are closely related to social network analysis, but often focusing on local patterns in the network. For example network motifs are small subgraphs that are over-represented in the network. Activity motifs are similar over-represented patterns in the attributes of nodes and edges in the network that are over represented given the network structure.

Link analysis

Link analysis is a subset of network analysis, exploring associations between objects. An example may be examining the addresses of suspects and victims, the telephone numbers they have dialed and financial transactions that they have partaken in during a given timeframe, and the familial relationships between these subjects as a part of police investigation. Link analysis here provides the crucial relationships and associations between very many objects of different types that are not apparent from isolated pieces of information. Computer-assisted or fully automatic computer-based link analysis is increasingly employed by banks and insurance agencies in fraud detection, by telecommunication operators in telecommunication network analysis, by medical sector in epidemiology and pharmacology, in law enforcement investigations, by search engines for relevance rating (and conversely by the spammers for spamdexing and by business owners for search engine optimization), and everywhere else where relationships between many objects have to be analyzed.

Web link analysis

Several Web search ranking algorithms use link-based centrality metrics, including (in order of appearance) Marchiori's Hyper Search, Google's PageRank, Kleinberg's HITS algorithm, and the TrustRank algorithm. Link analysis is also conducted in information science and communication science in order to understand and extract information from the structure of collections of web pages. For example the analysis might be of the interlinking between politicians' web sites or blogs.

Centrality measures

Information about the relative importance of nodes and edges in a graph can be obtained through centrality measures, widely used in disciplines like sociology. For example, eigenvector centrality uses the eigenvectors of the adjacency matrix to determine nodes that tend to be frequently visited.

Spread of content in networks

Content in a complex network can spread via two major methods: conserved spread and non-conserved spread.[2] In conserved spread, the total amount of content that enters a complex network remains constant as it passes through. The model of conserved spread can best be represented by a pitcher containing a fixed amount of water being poured into a series of funnels connected by tubes . Here, the pitcher represents the original source and the water is the content being spread. The funnels and connecting tubing represent the nodes and the connections between nodes, respectively. As the water passes from one funnel into another, the water disappears instantly from the funnel that was previously exposed to the water. In non-conserved spread, the amount of content changes as it enters and passes through a complex network. The model of non-conserved spread can best be represented by a continuously running faucet running through a series of funnels connected by tubes . Here, the amount of water from the original source is infinite. Also, any funnels that have been exposed to the water continue to experience the water even as it passes into successive funnels. The non-conserved model is the most suitable for explaining the transmission of most infectious diseases.

Implementations

- Orange, a free data mining software suite, module orngNetwork [3]
- Pajek [73], program for (large) network analysis and visualization
- Tulip, a free data mining and visualization software dedicated to the analysis and visualization of relational data. [4]

Notes

[1] Wasserman, Stanley and Katherine Faust. 1994. *Social Network Analysis: Methods and Applications.* Cambridge: Cambridge University Press.

[2] Newman, M., Barabási, A.-L., Watts, D.J. [eds.] (2006) The Structure and Dynamics of Networks. Princeton, N.J.: Princeton University Press.

External links

- netwiki (http://netwiki.amath.unc.edu/) Scientific wiki dedicated to network theory
- New Network Theory (http://www.networkcultures.org/networktheory/) International Conference on 'New Network Theory'
- Network Workbench (http://nwb.slis.indiana.edu/): A Large-Scale Network Analysis, Modeling and Visualization Toolkit
- Network analysis of computer networks (http://www.orgnet.com/SocialLifeOfRouters.pdf)
- Network analysis of organizational networks (http://www.orgnet.com/orgnetmap.pdf)
- Network analysis of terrorist networks (http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/941/863)
- Network analysis of a disease outbreak (http://www.orgnet.com/AJPH2007.pdf)
- Link Analysis: An Information Science Approach (http://linkanalysis.wlv.ac.uk/) (book)
- Connected: The Power of Six Degrees (http://gephi.org/2008/how-kevin-bacon-cured-cancer/) (documentary)
- Constructal law (http://www.constructal.org)

Closeness (graph theory)

Within graph theory and network analysis, there are various measures of the **centrality** of a vertex within a graph that determine the relative importance of a vertex within the graph (for example, how important a person is within a social network, or, in the theory of space syntax, how important a room is within a building or how well-used a road is within an urban network).

There are four measures of centrality that are widely used in network analysis: degree centrality, betweenness, closeness, and eigenvector centrality. For a review as well as generalizations to weighted networks, see Opsahl et al. (2010)[1] .

Degree centrality

The first, and simplest, is **degree centrality**. Degree centrality is defined as the number of links incident upon a node (i.e., the number of ties that a node has). Degree is often interpreted in terms of the immediate risk of node for catching whatever is flowing through the network (such as a virus, or some information). If the network is directed (meaning that ties have direction), then we usually define two separate measures of degree centrality, namely indegree and outdegree. Indegree is a count of the number of ties directed to the node, and outdegree is the number of ties that the node directs to others. For positive relations such as friendship or advice, we normally interpret indegree as a form of popularity, and outdegree as gregariousness.

For a graph $G := (V, E)$ with *n* vertices, the degree centrality $C_D(v)$ for vertex v is:

$$C_D(v) = \frac{\deg(v)}{n-1}$$

Calculating degree centrality for all nodes V in a graph takes $\Theta(V^2)$ in a dense adjacency matrix representation of the graph, and for edges E in a graph takes $\Theta(E)$ in a sparse matrix representation.
The definition of centrality can be extended to graphs. Let $v*$ be the node with highest degree centrality in G . Let $X := (Y, Z)$ be the n node connected graph that maximizes the following quantity (with $y*$ being the node with highest degree centrality in X):

$$H = \sum_{j=1}^{|Y|} C_D(y*) - C_D(y_j)$$

Then the degree centrality of the graph G is defined as follows:

$$C_D(G) = \frac{\sum_{i=1}^{|V|} [C_D(v*) - C_D(v_i)]}{H}$$

H is maximized when the graph X contains one node that is connected to all other nodes and all other nodes are connected only to this one central node (a star graph). In this case

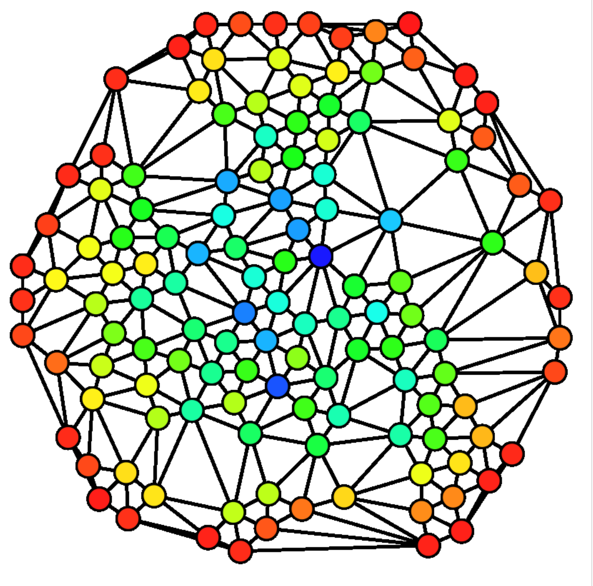
$$H = (n-1)(1 - \frac{1}{n-1}) = n - 2$$

so the degree centrality of G reduces to:

$$C_D(G) = \frac{\sum_{i=1}^{|V|} [C_D(v*) - C_D(v_i)]}{n-2}$$

Betweenness centrality

Betweenness is a centrality measure of a vertex within a graph (there is also edge betweenness, which is not discussed here). Vertices that occur on many shortest paths between other vertices have higher betweenness than those that do not.


Hue (from red=0 to blue=max) shows the node betweenness.

For a graph $G := (V, E)$ with *n* vertices, the betweenness $C_B(v)$ for vertex v is computed as follows:

1. For each pair of vertices (s,t), compute all shortest paths between them.

2. For each pair of vertices (s,t), determine the fraction of shortest paths that pass through the vertex in question (here, vertex v).

3. Sum this fraction over all pairs of vertices (s,t).

Or, more succinctly:[2]

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the number of shortest paths from *s* to *t*, and $\sigma_{st}(v)$ is the number of shortest paths from *s* to *t* that pass through a vertex *v*. This may be normalised by dividing through the number of pairs of vertices not including *v*, which is $(n-1)(n-2)$ for directed graphs and $(n-1)(n-2)/2$ for undirected graphs. For example, in an undirected star graph, the center vertex (which is contained in every possible shortest path) would have a betweenness of $(n-1)(n-2)/2$ (1, if normalised) while the leaves (which are contained in no shortest paths) would have a betweenness of 0.

Calculating the betweenness and closeness centralities of all the vertices in a graph involves calculating the shortest paths between all pairs of vertices on a graph. This takes $\Theta(V^3)$ time with the Floyd–Warshall algorithm, modified to not only find one but count all shortest paths between two nodes. On a sparse graph, Johnson's algorithm may be more efficient, taking $O(V^2 \log V + VE)$ time. On unweighted graphs, calculating betweenness centrality takes $O(VE)$ time using Brandes' algorithm[3] .

In calculating betweenness and closeness centralities of all vertices in a graph, it is assumed that graphs are undirected and connected with the allowance of loops and multiple edges. When specifically dealing with network graphs, oftentimes graphs are without loops or multiple edges to maintain simple relationships (where edges represent connections between two people or vertices). In this case, using Brandes' algorithm will divide final centrality scores by 2 to account for each shortest path being counted twice[3] .

Closeness centrality

In topology and related areas in mathematics, **closeness** is one of the basic concepts in a topological space. Intuitively we say two sets are close if they are arbitrarily near to each other. The concept can be defined naturally in a metric space where a notion of distance between elements of the space is defined, but it can be generalized to topological spaces where we have no concrete way to measure distances.

In graph theory **closeness** is a centrality measure of a vertex within a graph. Vertices that are 'shallow' to other vertices (that is, those that tend to have short geodesic distances to other vertices with in the graph) have higher closeness. Closeness is preferred in network analysis to mean shortest-path length, as it gives higher values to more central vertices, and so is usually positively associated with other measures such as degree.

In the network theory, **closeness** is a sophisticated measure of centrality. It is defined as the mean geodesic distance (i.e., the shortest path) between a vertex *v* and all other vertices reachable from it:

$$\frac{\sum\limits_{t \in V \setminus v} d_G(v,t)}{n-1}$$

where $n \geq 2$ is the size of the network's 'connectivity component' *V* reachable from *v*. Closeness can be regarded as a measure of how long it will take information to spread from a given vertex to other reachable vertices in the network[4] .

Some define closeness to be the reciprocal of this quantity, but either way the information communicated is the same (this time estimating the speed instead of the timespan). The closeness $C_C(v)$ for a vertex v is the reciprocal of the sum of geodesic distances to all other vertices of *V*[5] :

$$C_C(v) = \frac{1}{\sum_{t \in V \setminus v} d_G(v,t)}.$$

Different methods and algorithms can be introduced to measure closeness, like the *random-walk centrality* introduced by Noh and Rieger (2003) that is a measure of the speed with which randomly walking messages reach a vertex from elsewhere in the network—a sort of random-walk version of closeness centrality[6] .

The *information centrality* of Stephenson and Zelen (1989) is another closeness measure, which bears some similarity to that of Noh and Rieger. In essence it measures the harmonic mean length of paths ending at a vertex $\mathbf{i}$, which is smaller if $\mathbf{i}$ has many short paths connecting it to other vertices[7] .

Dangalchev (2006), in order to measure the network vulnerability, modifies the definition for closeness so it can be used for disconnected graphs and the total closeness is easier to calculate[8] :

$$C_C(v) = \sum_{t \in V \setminus v} 2^{-d_G(v,t)}.$$

An extension to networks with disconnected components has been proposed by Opsahl (2010)[9] .

Eigenvector centrality

Eigenvector centrality is a measure of the importance of a node in a network. It assigns relative scores to all nodes in the network based on the principle that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. Google's PageRank is a variant of the Eigenvector centrality measure.

Using the adjacency matrix to find eigenvector centrality

Let x_i denote the score of the *i*th node. Let $A_{i,j}$ be the adjacency matrix of the network. Hence $A_{i,j} = 1$ if the *i*th node is adjacent to the *j*th node, and $A_{i,j} = 0$ otherwise. More generally, the entries in *A* can be real numbers representing connection strengths, as in a stochastic matrix.
For the i^{th} node, let the centrality score be proportional to the sum of the scores of all nodes which are connected to it. Hence

$$x_i = \frac{1}{\lambda} \sum_{j \in M(i)} x_j = \frac{1}{\lambda} \sum_{j=1}^{N} A_{i,j} x_j$$

where $M(i)$ is the set of nodes that are connected to the i^{th} node, *N* is the total number of nodes and λ is a constant. In vector notation this can be rewritten as

$\mathbf{x} = \frac{1}{\lambda}\mathbf{A}\mathbf{x}$, or as the eigenvector equation $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$

In general, there will be many different eigenvalues λ for which an eigenvector solution exists. However, the additional requirement that all the entries in the eigenvector be positive implies (by the Perron–Frobenius theorem) that only the greatest eigenvalue results in the desired centrality measure.[10] The i^{th} component of the related eigenvector then gives the centrality score of the i^{th} node in the network. Power iteration is one of many eigenvalue algorithms that may be used to find this dominant eigenvector.

Notes and references

[1] Opsahl, Tore; Agneessens, Filip; Skvoretz, John (2010). "Node centrality in weighted networks: Generalizing degree and shortest paths" (http://toreopsahl.com/2010/04/21/article-node-centrality-in-weighted-networks-generalizing-degree-and-shortest-paths/). *Social Networks* **32**: 245. doi:10.1016/j.socnet.2010.03.006. .

[2] Shivaram Narayanan. The Betweenness Centrality Of Biological Networks (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.958&rep=rep1&type=pdf). (Thesis).

[3] Ulrik Brandes (PDF). *A faster algorithm for betweenness centrality* (http://www.cs.ucc.ie/~rb4/resources/Brandes.pdf). .

[4] Newman, MEJ, 2003, Arxiv preprint cond-mat/0309045 (http://arxiv.org/abs/cond-mat/0309045).

[5] Sabidussi, G. (1966) The centrality index of a graph. *Psychometrika* **31**, 581--603.

[6] J. D. Noh and H. Rieger, Phys. Rev. Lett. 92, 118701 (2004).

[7] Stephenson, K. A. and Zelen, M., 1989. Rethinking centrality: Methods and examples. Social Networks 11, 1–37.

[8] Dangalchev Ch., Residual Closeness in Networks, Phisica A **365**, 556 (2006).

[9] Tore Opsahl. *Closeness centrality in networks with disconnected components* (http://toreopsahl.com/2010/03/20/closeness-centrality-in-networks-with-disconnected-components/). .

[10] M. E. J. Newman (PDF). *The mathematics of networks* (http://www-personal.umich.edu/~mejn/papers/palgrave.pdf). . Retrieved 2006-11-09.

Further reading

- Freeman, L. C. (1979). Centrality in social networks: Conceptual clarification. Social Networks, 1(3), 215-239.
- Sabidussi, G. (1966). The centrality index of a graph. Psychometrika, 31 (4), 581-603.
- Freeman, L. C. (1977) A set of measures of centrality based on betweenness. *Sociometry* **40**, 35-41.
- Koschützki, D.; Lehmann, K. A.; Peeters, L.; Richter, S.; Tenfelde-Podehl, D. and Zlotowski, O. (2005) Centrality Indices. In Brandes, U. and Erlebach, T. (Eds.) *Network Analysis: Methodological Foundations*, pp. 16–61, LNCS 3418, Springer-Verlag.
- Bonacich, P.(1987) Power and Centrality: A Family of Measures, *The American Journal of Sociology*, 92 (5), pp 1170–1182

External links

- https://networkx.lanl.gov/trac/attachment/ticket/119/page_rank.py

Dense graph

In mathematics, a **dense graph** is a graph in which the number of edges is close to the maximal number of edges. The opposite, a graph with only a few edges, is a **sparse graph**.

The distinction between sparse and dense graphs is rather vague. One possibility is to choose a number k with $1 < k < 2$ and to define a *sparse graph* to be a graph with $|E| = O(|V|^k)$, where $|E|$ denotes the number of edges, $|V|$ the number of vertices, and the letter O refers to the Big O notation (Preiss 1998, p. 534).

For undirected simple graphs, the **graph density** is defined as:

$$D = \frac{2|E|}{|V|\,(|V|-1)}$$

The maximum number of edges is ½ $|V|$ $(|V|-1)$, so the maximal density is 1 (for complete graphs) and the minimal density is 0 (Coleman & Moré 1983).

Upper density

Upper density is an extension of the concept of graph density defined above from finite graphs to infinite graphs. Intuitively, an infinite graph has arbitrarily large finite subgraphs with any density less than its upper density, and does not have arbitrarily large finite subgraphs with density greater than its upper density. Formally, the upper density of a graph G is the infimum of the values α such that the finite subgraphs of G with density α have a bounded number of vertices. It can be shown using the Erdős-Stone theorem that the upper density can only be 1 or one of the superparticular ratios 0, 1/2, 2/3, 3/4, 4/5, ... $n/(n + 1)$, ... (see, e.g., Diestel, p. 189).

Sparse and tight graphs

Streinu & Theran (2008) define a graph as being (k,l)-sparse if every nonempty subgraph with n vertices has at most $kn - l$ edges, and (k,l)-tight if it is (k,l)-sparse and has exactly $kn - l$ edges. Thus trees are exactly the (1,1)-tight graphs, forests are exactly the (1,1)-sparse graphs, and graphs with arboricity k are exactly the (k,k)-sparse graphs. Pseudoforests are exactly the (1,0)-sparse graphs, and the Laman graphs arising in rigidity theory are exactly the (2,3)-tight graphs.

Other graph families not characterized by their sparsity can also be described in this way. For instance the facts that any planar graph with n vertices has at most $3n - 6$ edges, and that any subgraph of a planar graph is planar, together imply that the planar graphs are (3,6)-sparse. However, not every (3,6)-sparse graph is planar. Similarly, outerplanar graphs are (2,3)-sparse and planar bipartite graphs are (2,4)-sparse.

Streinu and Theran show that testing (*k*,*l*)-sparsity may be performed in polynomial time.

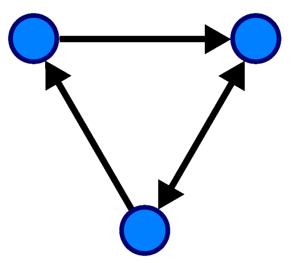
References

- Paul E. Black, Sparse graph [1], from *Dictionary of Algorithms and Data Structures,* Paul E. Black (ed.), NIST. Retrieved on 29 September 2005.
- Coleman, Thomas F.; Moré, Jorge J. (1983), "Estimation of sparse Jacobian matrices and graph coloring Problems", *SIAM Journal on Numerical Analysis* **20** (1): 187–209, doi:10.1137/0720013.
- Diestel, Reinhard (2005), *Graph Theory*, Graduate Texts in Mathematics, Springer-Verlag, ISBN 3540261834, OCLC 181535575.
- Preiss, Bruno (1998), *Data Structures and Algorithms with Object-Oriented Design Patterns in C++*, John Wiley & Sons, ISBN 0-471-24134-2.
- Streinu, I.; Theran, L. (2008), "Sparse hypergraphs and pebble game algorithms", *European Journal of Combinatorics (special issue for Oriented Matroids '05)*, arXiv:math/0703921.

Directed graph

A **directed graph** or **digraph** is a pair $G=(V,A)$ (sometimes $G=(V,E)$) of:[1]

- a set *V*, whose elements are called **vertices** or **nodes,**
- a set *A* of ordered pairs of vertices, called **arcs**, **directed edges**, or **arrows** (and sometimes simply **edges** with the corresponding set named *E* instead of *A*).


A directed graph.

It differs from an ordinary or undirected graph, in that the latter is defined in terms of unordered pairs of vertices, which are usually called edges.

Sometimes a digraph is called a **simple digraph** to distinguish it from a **directed multigraph**, in which the arcs constitute a multiset, rather than a set, of ordered pairs of vertices. Also, in a simple digraph loops are disallowed. (A loop is an arc that pairs a vertex to itself.) On the other hand, some texts allow loops, multiple arcs, or both in a digraph.

Basic terminology

An arc $e=(x,y)$ is considered to be directed **from** x **to** y; y is called the **head** and x is called the **tail** of the arc; y is said to be a **direct successor** of x, and x is said to be a **direct predecessor** of y. If a path made up of one or more successive arcs leads from x to y, then y is said to be a **successor** of x, and x is said to be a **predecessor** of y. The arc (y,x) is called the arc (x,y) **inverted**.
A directed graph *G* is called **symmetric** if, for every arc that belongs to *G*, the corresponding inverted arc also belongs to *G*. A symmetric loopless directed graph is equivalent to an undirected graph with the pairs of inverted arcs replaced with edges; thus the number of edges is equal to the number of arcs halved.

The **orientation** of a simple undirected graph is obtained by assigning a direction to each edge. Any directed graph constructed this way is called an **oriented graph**. A distinction between a simple directed graph and an oriented graph is that if x and y are vertices, a simple directed graph allows both (x,y) and (y,x) as edges, while only one is permitted in an oriented graph.[2] [3] [4]
A **weighted digraph** is a digraph with weights assigned for its arcs, similarly to the weighted graph.

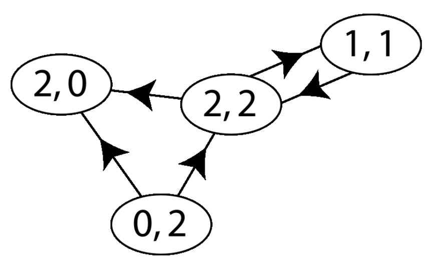
The adjacency matrix of a digraph (with loops and multiple arcs) is the integer-valued matrix with rows and columns corresponding to the digraph nodes, where a nondiagonal entry a_{ij} is the number of arcs from node *i* to node *j*, and the diagonal entry a_{ii} is the number of loops at node *i*. The adjacency matrix for a digraph is unique up to the permutations of rows and columns.

Another matrix representation for a digraph is its incidence matrix.

See Glossary of graph theory#Direction for more definitions.

Indegree and outdegree

For a node, the number of head endpoints adjacent to a node is called the **indegree** of the node and the number of tail endpoints is its **outdegree**.



A digraph with vertices labeled (indegree, outdegree)

The indegree is denoted $\deg^-(v)$ and the outdegree as $\deg^+(v)$. A vertex with $\deg^-(v) = 0$ is called a **source**, as it is the origin of each of its incident edges. Similarly, a vertex with $\deg^+(v) = 0$ is called a **sink**.
The **degree sum formula** states that, for a directed graph

$$\sum_{v\in V} \deg^+(v) = \sum_{v\in V} \deg^-(v) = |A|\,.$$

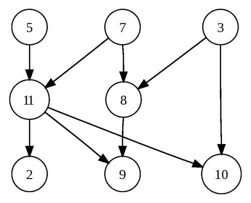
If for every node, *v* ∈ *V*, $\deg^+(v) = \deg^-(v)$, the graph is called a **balanced digraph**.

Digraph connectivity

A digraph G is called **weakly connected** (or just **connected**[5]) if the undirected **underlying graph** obtained by replacing all directed edges of G with undirected edges is a connected graph. A digraph is **strongly connected** or **strong** if it contains a directed path from *u* to *v* and a directed path from *v* to *u* for every pair of vertices *u*,*v*. The **strong components** are the maximal strongly connected subgraphs.

Classes of digraphs

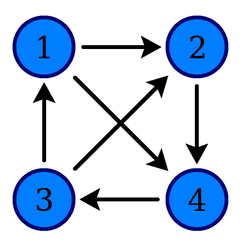
An acyclic digraph (occasionally called a **dag** or **DAG** for "directed acyclic graph", although it is not the same as an orientation of an acyclic graph) is a directed graph with no directed cycles.



A simple directed acyclic graph

A rooted tree naturally defines an acyclic digraph, if all edges of the underlying tree are directed away from the root.

A tournament is an oriented graph obtained by choosing a direction for each edge in an undirected complete graph.



A tournament on 4 vertices

In the theory of Lie groups, a quiver Q is a directed graph serving as the domain of, and thus characterizing the shape of, a **representation** V defined as a functor, specifically an object of the functor category $\mathbf{FinVct}_K^{F(Q)}$ where $F(Q)$ is the free category on Q consisting of paths in Q and $\mathbf{FinVct}_K$ is the category of finite dimensional vector spaces over a field K. Representations of a quiver label its vertices with vector spaces and its edges (and hence paths) compatibly with linear transformations between them, and transform via natural transformations.

Notes

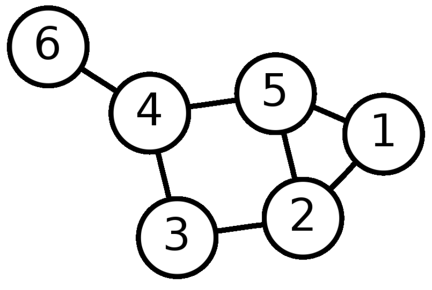
[1] Bang-Jensen & Gutin (2000). Diestel (2005), Section 1.10. Bondy & Murty (1976), Section 10.
[2] Diestel (2005), Section 1.10.
[3] Weisstein, Eric W., " Oriented Graph (http://mathworld.wolfram.com/OrientedGraph.html)" from MathWorld.
[4] Weisstein, Eric W., " Graph Orientation (http://mathworld.wolfram.com/GraphOrientation.html)" from MathWorld.
[5] Bang-Jensen & Gutin (2000) p. 19 in the 2007 edition; p. 20 in the 2nd edition (2009).

References

- Bang-Jensen, Jørgen; Gutin, Gregory (2000), *Digraphs: Theory, Algorithms and Applications* (http://www.cs.rhul.ac.uk/books/dbook/), Springer, ISBN 1-85233-268-9
  (the corrected 1st edition of 2007 is now freely available on the authors' site; the 2nd edition appeared in 2009 ISBN 1848009976).
- Bondy, John Adrian; Murty, U. S. R. (1976), *Graph Theory with Applications* (http://www.ecp6.jussieu.fr/pageperso/bondy/books/gtwa/gtwa.html), North-Holland, ISBN 0-444-19451-7.
- Diestel, Reinhard (2005), *Graph Theory* (http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/) (3rd ed.), Springer, ISBN 3-540-26182-6 (the electronic 3rd edition is freely available on author's site).
- Harary, Frank; Norman, Robert Z.; Cartwright, Dorwin (1965), *Structural Models: An Introduction to the Theory of Directed Graphs*, New York: Wiley.
- *Number of directed graphs (or digraphs) with n nodes.* (http://oeis.org/A000273)

Vertex (graph theory)

In graph theory, a **vertex** (plural **vertices**) or **node** is the fundamental unit out of which graphs are formed: an undirected graph consists of a set of vertices and a set of edges (unordered pairs of vertices), while a directed graph consists of a set of vertices and a set of arcs (ordered pairs of vertices). From the point of view of graph theory, vertices are treated as featureless and indivisible objects, although they may have additional structure depending on the application from which the graph arises; for instance, a semantic network is a graph in which the vertices represent concepts or classes of objects.



A graph with 6 vertices and 7 edges where the vertex no 6 on the far-left is a **leaf vertex** or a **pendant vertex**.

The two vertices forming an edge are said to be its endpoints, and the edge is said to be incident to the vertices. A vertex *w* is said to be adjacent to another vertex *v* if the graph contains an edge (*v*,*w*). The neighborhood of a vertex *v* is an induced subgraph of the graph, formed by all vertices adjacent to *v*.

The degree of a vertex in a graph is the number of edges incident to it. An **isolated vertex** is a vertex with degree zero; that is, a vertex that is not an endpoint of any edge. A **leaf vertex** (also **pendant vertex**) is a vertex with degree one. In a directed graph, one can distinguish the outdegree (number of outgoing edges) from the indegree (number of incoming edges); a **source vertex** is a vertex with indegree zero, while a **sink vertex** is a vertex with outdegree zero.

A cut vertex is a vertex the removal of which would disconnect the remaining graph; a vertex separator is a collection of vertices the removal of which would disconnect the remaining graph into small pieces. A k-vertex-connected graph is a graph in which removing fewer than *k* vertices always leaves the remaining graph connected. An independent set is a set of vertices no two of which are adjacent, and a vertex cover is a set of vertices that includes the endpoint of each edge in the graph. The vertex space of a graph is a vector space having a set of basis vectors corresponding with the graph's vertices.

A graph is vertex-transitive if it has symmetries that map any vertex to any other vertex. In the context of graph enumeration and graph isomorphism it is important to distinguish between **labeled vertices** and **unlabeled vertices**. A labeled vertex is a vertex that is associated with extra information that enables it to be distinguished from other labeled vertices; two graphs can be considered isomorphic only if the correspondence between their vertices pairs up vertices with equal labels. An unlabeled vertex is one that can be substituted for any other vertex based only on its adjacencies in the graph and not based on any additional information.

Vertices in graphs are analogous to, but not the same as, vertices of polyhedra: the skeleton of a polyhedron forms a graph, the vertices of which are the vertices of the polyhedron, but polyhedron vertices have additional structure (their geometric location) that is not assumed to be present in graph theory. The vertex figure of a vertex in a polyhedron is analogous to the neighborhood of a vertex in a graph.

In a directed graph, the forward star of a node u is defined as its outgoing edges. In a Graph G with the set of vertices V and the set of edges E, the forward star of u can be described as

$$\{(u,v) \in E\}.$$ [1]

Notes

[1] (Gallo 1988, p. 4)

References

- Gallo, Giorgio; Pallotino, Stefano (December 1988). "Shortest Path Algorithms" (http://www.springerlink.com/content/awn535w405321948/) (PDF). *Annals of Operations Research* (Netherlands: Springer) **13** (1): 1–79. doi:10.1007/BF02288320. Retrieved 2008-06-18.
- Berge, Claude, *Théorie des graphes et ses applications*. Collection Universitaire de Mathématiques, II Dunod, Paris 1958, viii+277 pp. (English edition, Wiley 1961; Methuen & Co, New York 1962; Russian, Moscow 1961; Spanish, Mexico 1962; Roumanian, Bucharest 1969; Chinese, Shanghai 1963; Second printing of the 1962 first English edition. Dover, New York 2001)
- Chartrand, Gary (1985). *Introductory graph theory*. New York: Dover. ISBN 0-486-24775-9.
- Biggs, Norman; Lloyd, E. H.; Wilson, Robin J. (1986). *Graph theory, 1736-1936*. Oxford [Oxfordshire]: Clarendon Press. ISBN 0-19-853916-9.
- Harary, Frank (1969). *Graph theory*. Reading, Mass.: Addison-Wesley Publishing. ISBN 0-201-41033-8.
- Harary, Frank; Palmer, Edgar M. (1973). *Graphical enumeration*. New York, Academic Press. ISBN 0-12-324245-2.

External links

- Weisstein, Eric W., " Graph Vertex (http://mathworld.wolfram.com/GraphVertex.html)" from MathWorld.

Flow network

In graph theory, a **flow network** is a directed graph where each edge has a **capacity** and each edge receives a flow. The amount of flow on an edge cannot exceed the capacity of the edge. Often in Operations Research, a directed graph is called a **network**, the vertices are called **nodes** and the edges are called **arcs**. A flow must satisfy the restriction that the amount of flow into a node equals the amount of flow out of it, except when it is a **source**, which has more outgoing flow, or **sink**, which has more incoming flow. A network can be used to model traffic in a road system, fluids in pipes, currents in an electrical circuit, or anything similar in which something travels through a network of nodes.

Definition

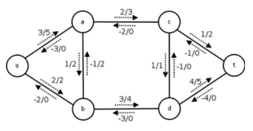
$G(V,E)$is a finite directed graph in which every edge $(u,v) \in E$ has a non-negative, real-valued capacity $c(u,v)$. If $(u,v) \notin E$, we assume that $c(u,v) = 0$. We distinguish two vertices: a source s and a sink t . A flow network is a real function $f : V \times V \rightarrow \mathbb{R}$ with the following three properties for all nodes u and v :

Capacity constraints:	$f(u,v) \leq c(u,v)$. The flow along an edge cannot exceed its capacity.
Skew symmetry:	$f(u,v) = -f(v,u)$. The net flow from u to v must be the opposite of the net flow from v to u (see example).
Flow conservation:	$\sum_{w \in V} f(u,w) = 0$, unless $u = s$ or $w = t$. The net flow to a node is zero, except for the source, which "produces" flow, and the sink, which "consumes" flow.

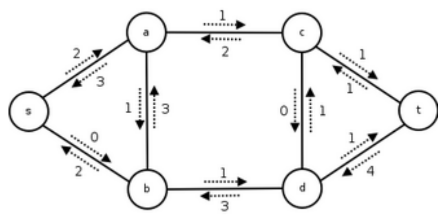
Notice that $f(u,v)$ is the *net* flow from u to v. If the graph represents a physical network, and if there is a real flow of, for example, 4 units from u to v, and a real flow of 3 units from v to u, we have $f(u,v) = 1$ and $f(v,u) = -1$.
The **residual capacity** of an edge is $c_f(u,v) = c(u,v) - f(u,v)$. This defines a **residual network** denoted $G_f(V, E_f)$, giving the amount of *available* capacity. See that there can be an edge from u to v in the residual network, even though there is no edge from u to v in the original network. Since flows in opposite directions cancel out, *decreasing* the flow from v to u is the same as *increasing* the flow from u to v. An **augmenting path** is a path $(u_1, u_2, \dots, u_k)$ in the residual network, where $u_1 = s$, $u_k = t$, and $c_f(u_i, u_{i+1}) > 0$. A network is at maximum flow if and only if there is no augmenting path in the residual network.

Example

To the right you see a flow network with source labeled s, sink t, and four additional nodes. The flow and capacity is denoted f/c. Notice how the network upholds skew symmetry, capacity constraints and flow conservation. The total amount of flow from s to t is 5, which can be easily seen from the fact that the total outgoing flow from s is 5, which is also the incoming flow to t. We know that no flow appears or disappears in any of the other nodes.



A flow network showing flow and capacity



Residual network for the above flow network, showing residual capacities

Below you see the residual network for the given flow. Notice how there is positive residual capacity on some edges where the original capacity is zero, for example for the edge (d,c). This flow is not a maximum flow. There is available capacity along the paths (s,a,c,t), (s,a,b,d,t) and (s,a,b,d,c,t), which are then the augmenting paths. The residual capacity of the first path is $\min(c(s,a) - f(s,a), c(a,c) - f(a,c), c(c,t) - f(c,t))$ $= \min(5-3, 3-2, 2-1) = \min(2,1,1) = 1$. Notice that augmenting path (s,a,b,d,c,t) does not exist in the original network, but you can send flow along it, and still get a legal flow.
If this is a real network, there might actually be a flow of 2 from a to b, and a flow of 1 from b to a, but we only maintain the **net** flow.

Applications

Picture a series of water pipes, fitting into a network. Each pipe is of a certain diameter, so it can only maintain a flow of a certain amount of water. Anywhere that pipes meet, the total amount of water coming into that junction must be equal to the amount going out, otherwise we would quickly run out of water, or we would have a build up of water. We have a water inlet, which is the source, and an outlet, the sink. A flow would then be one possible way for water to get from source to sink so that the total amount of water coming out of the outlet is consistent. Intuitively, the total flow of a network is the rate at which water comes out of the outlet.

Flows can pertain to people or material over transportation networks, or to electricity over electrical distribution systems. For any such physical network, the flow coming into any intermediate node needs to equal the flow going out of that node. Bollobás characterizes this constraint in terms of Kirchhoff's current law, while later authors (i.e.: Chartrand) mention its generalization to some conservation equation.

Flow networks also find applications in ecology: flow networks arise naturally when considering the flow of nutrients and energy between different organizations in a food web. The mathematical problems associated with such networks are quite different from those that arise in networks of fluid or traffic flow. The field of ecosystem network analysis, developed by Robert Ulanowicz and others, involves using concepts from information theory and thermodynamics to study the evolution of these networks over time.

The simplest and most common problem using flow networks is to find what is called the maximum flow, which provides the largest possible total flow from the source to the sink in a given graph. There are many other problems which can be solved using max flow algorithms, if they are appropriately modeled as flow networks, such as bipartite matching, the assignment problem and the transportation problem.

In a multi-commodity flow problem, you have multiple sources and sinks, and various "commodities" which are to flow from a given source to a given sink. This could be for example various goods that are produced at various factories, and are to be delivered to various given customers through the *same* transportation network.

In a minimum cost flow problem, each edge u, v has a given cost $k(u, v)$, and the cost of sending the flow $f(u, v)$ across the edge is $f(u, v) \cdot k(u, v)$. The objective is to send a given amount of flow from the source to the sink, at the lowest possible price.
In a circulation problem, you have a lower bound $l(u, v)$ on the edges, in addition to the upper bound $c(u, v)$. Each edge also has a cost. Often, flow conservation holds for *all* nodes in a circulation problem, and there is a connection from the sink back to the source. In this way, you can dictate the total flow with $l(t, s)$ and $c(t, s)$. The flow *circulates* through the network, hence the name of the problem.
In a **network with gains** or **generalized network** each edge has a **gain**, a real number (not zero) such that, if the edge has gain *g*, and an amount *x* flows into the edge at its tail, then an amount *gx* flows out at the head.

Further reading

- George T. Heineman, Gary Pollice, and Stanley Selkow (2008). "Chapter 8:Network Flow Algorithms". *Algorithms in a Nutshell*. Oreilly Media. pp. 226–250. ISBN 978-0-596-51624-6.
- Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice Hall. ISBN 0-13-617549-X.
- Bollobás, Béla (1979). *Graph Theory: An Introductory Course*. Heidelberg: Springer-Verlag. ISBN 3-540-90399-2.
- Chartrand, Gary & Oellermann, Ortrud R. (1993). *Applied and Algorithmic Graph Theory*. New York: McGraw-Hill. ISBN 0-07-557101-3.
- Even, Shimon (1979). *Graph Algorithms*. Rockville, Maryland: Computer Science Press. ISBN 0-914894-21-8.
- Gibbons, Alan (1985). *Algorithmic Graph Theory*. Cambridge: Cambridge University Press. ISBN 0-521-28881-9 ISBN 0-521-24659-8.

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (2001) [1990]. "26". *Introduction to Algorithms* (2nd edition ed.). MIT Press and McGraw-Hill. pp. 696–697. ISBN 0-262-03293-7.

External links

- Maximum Flow Problem [1]
- Maximum Flow [2]
- Real graph instances [3]
- Software, papers, test graphs, etc. [4]
- Solutions for network flow problems [5]
- Software and papers for network flow problems [6]
- Lemon C++ library with several maximum flow and minimum cost circulation algorithms [7]
- QuickGraph [8], graph data structures and algorithms for .Net

Cycle (graph theory)

In graph theory, the term **cycle** may refer to several closely related objects.

- A closed walk, with repeated vertices allowed. See path (graph theory). (This usage is common in computer science. In graph theory it is more often called a **closed walk**.)
- A closed (simple) path, with no other repeated vertices or edges other than the starting and ending vertices. (This usage is common in graph theory, see "Cycle graph") This may also be called a **simple cycle**, **circuit**, **circle**, or **polygon**.
- A closed directed walk, with repeated vertices allowed. (This usage is common in computer science. In graph theory it is more often called a **closed directed walk**.)
- A closed directed (simple) path, with no repeated vertices other than the starting and ending vertices. (This usage is common in graph theory.) This may also be called a **simple (directed) cycle**.
- The edge set of an undirected closed path without repeated vertices or edges. This may also be called a **circuit**, **circle**, or **polygon**.
- An element of the binary or integral (or real, complex, etc.) cycle space of a graph. (This is the usage closest to that in the rest of mathematics, in particular algebraic topology.) Such a cycle may be called a **binary cycle**, **integral cycle**, etc.
- An edge set which has even degree at every vertex; also called an **even edge set** or, when taken together with its vertices, an **even subgraph**. This is equivalent to a binary cycle, since a binary cycle is the indicator function of an edge set of this type.

Chordless cycles in a graph are sometimes called **graph holes**. A **graph antihole** is the complement of a graph hole.

See also

- Euler cycle
- Hamiltonian cycle
- Chordal graph

References

Adjacency matrix

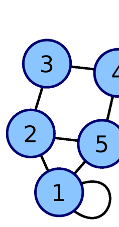
In mathematics and computer science, an **adjacency matrix** is a means of representing which vertices of a graph are adjacent to which other vertices. Another matrix representation for a graph is the incidence matrix.

Specifically, the adjacency matrix of a finite graph **G** on *n* vertices is the *n* × *n* matrix where the non-diagonal entry a_{ij} is the number of edges from vertex *i* to vertex *j*, and the diagonal entry a_{ii}, depending on the convention, is either once or twice the number of edges (loops) from vertex *i* to itself. Undirected graphs often use the former convention of counting loops twice, whereas directed graphs typically use the latter convention. There exists a unique adjacency matrix for each isomorphism class of graphs (up to permuting rows and columns), and it is not the adjacency matrix of any other isomorphism class of graphs. In the special case of a finite simple graph, the adjacency matrix is a (0,1)-matrix with zeros on its diagonal. If the graph is undirected, the adjacency matrix is symmetric.

The relationship between a graph and the eigenvalues and eigenvectors of its adjacency matrix is studied in spectral graph theory.

Examples

- Here is an example of a labeled graph and its adjacency matrix. The convention followed here is that an adjacent edge counts 1 in the matrix for an undirected graph. (X,Y coordinates are 1-6)

Labeled graph	Adjacency matrix
	$\begin{pmatrix}1&1&0&0&1&0\\1&0&1&0&1&0\\0&1&0&1&0&0\\0&0&1&0&1&1\\1&1&0&1&0&0\\0&0&0&1&0&0\end{pmatrix}$

- The adjacency matrix of a complete graph is all 1's except for 0's on the diagonal.
- The adjacency matrix of an empty graph is a zero matrix.

Adjacency matrix of a bipartite graph

The adjacency matrix *A* of a bipartite graph whose parts have *r* and *s* vertices has the form

$$A = \begin{pmatrix} O & B \\ B^T & O \end{pmatrix},$$

where *B* is an *r* × *s* matrix and *O* is an all-zero matrix. Clearly, the matrix *B* uniquely represents the bipartite graphs, and it is commonly called its biadjacency matrix.

Formally, let *G* = (*U*, *V*, *E*) be a bipartite graph or bigraph with parts $U = u_1, \ldots, u_r$ and $V = v_1, \ldots, v_s$. An *r* x *s* 0-1 matrix *B* is called the **biadjacency matrix** if $B_{i,j} = 1$ iff $(u_i, v_j) \in E$.
If *G* is a bipartite multigraph or weighted graph then the elements $B_{i,j}$ are taken to be the number of edges between or the weight of (u_i, v_j) respectively.

Properties

The adjacency matrix of an undirected simple graph is symmetric, and therefore has a complete set of real eigenvalues and an orthogonal eigenvector basis. The set of eigenvalues of a graph is the **spectrum** of the graph.

Suppose two directed or undirected graphs G_1 and G_2 with adjacency matrices A_1 and A_2 are given. G_1 and G_2 are isomorphic if and only if there exists a permutation matrix P such that

$$PA_1P^{-1} = A_2.$$

In particular, A_1 and A_2 are similar and therefore have the same minimal polynomial, characteristic polynomial, eigenvalues, determinant and trace. These can therefore serve as isomorphism invariants of graphs. However, two graphs may possess the same set of eigenvalues but not be isomorphic.

If *A* is the adjacency matrix of the directed or undirected graph *G*, then the matrix A^n (i.e., the matrix product of *n* copies of *A*) has an interesting interpretation: the entry in row *i* and column *j* gives the number of (directed or undirected) walks of length *n* from vertex *i* to vertex *j*. This implies, for example, that the number of triangles in an undirected graph *G* is exactly the trace of A^3 divided by 6.

The main diagonal of every adjacency matrix corresponding to a graph without loops has all zero entries.

For (d)-regular graphs, d is also an eigenvalue of A for the vector $v = (1, \dots, 1)$, and G is connected if and only if the multiplicity of d is 1. It can be shown that $-d$ is also an eigenvalue of A if G is a connected bipartite graph. The above are results of Perron–Frobenius theorem.

Variations

The Seidel adjacency matrix or **(0,−1,1)-adjacency matrix** of a simple graph has zero on the diagonal and entry $a_{ij} = -1$ if *ij* is an edge and +1 if it is not. This matrix is used in studying strongly regular graphs and two-graphs.

A distance matrix is like a higher-level adjacency matrix. Instead of only providing information about whether or not two vertices are connected, also tells the distances between them. This assumes the length of every edge is 1. A variation is where the length of an edge is not necessarily 1.

Data structures

When used as a data structure, the main alternative for the adjacency matrix is the adjacency list. Because each entry in the adjacency matrix requires only one bit, they can be represented in a very compact way, occupying only $n^2/8$ bytes of contiguous space, where n is the number of vertices. Besides just avoiding wasted space, this compactness encourages locality of reference.

On the other hand, for a sparse graph, adjacency lists win out, because they do not use any space to represent edges which are *not* present. Using a naïve array implementation on a 32-bit computer, an adjacency list for an undirected graph requires about $8e$ bytes of storage, where e is the number of edges.

Noting that a simple graph can have at most n^2 edges, allowing loops, we can let $d = e/n^2$ denote the *density* of the graph. Then, $8e > n^2/8$, or the adjacency list representation occupies more space, precisely when $d > 1/64$. Thus a graph must be sparse indeed to justify an adjacency list representation.

Besides the space tradeoff, the different data structures also facilitate different operations. Finding all vertices adjacent to a given vertex in an adjacency list is as simple as reading the list. With an adjacency matrix, an entire row must instead be scanned, which takes O(n) time. Whether there is an edge between two given vertices can be determined at once with an adjacency matrix, while requiring time proportional to the minimum degree of the two vertices with the adjacency list.

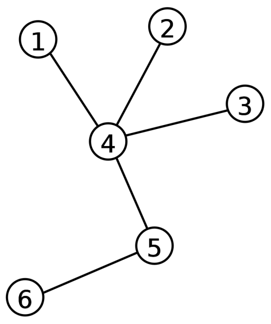
References

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (2001), *Introduction to Algorithms*, second edition. MIT Press and McGraw-Hill. ISBN 0-262-03293-7. Section 22.1: Representations of graphs, pp. 527–531.
- Chris Godsil and Gordon Royle (2001), Algebraic Graph Theory. New York: Springer-Verlag. ISBN 0-387-95241-1

External links

- Fluffschack (http://www.x2d.org/java/projects/fluffschack.jnlp) — an educational Java web start game demonstrating the relationship between adjacency matrices and graphs.

Tree (graph theory)

Trees	
 A labeled tree with 6 vertices and 5 edges	
Vertices	v
Edges	v - 1
Chromatic number	2 if $v > 1$

In mathematics, more specifically graph theory, a **tree** is an undirected graph in which any two vertices are connected by *exactly one* simple path. In other words, any connected graph without cycles is a tree. A **forest** is a disjoint union of trees.

The various kinds of data structures referred to as trees in computer science are similar to trees in graph theory, except that computer science trees have directed edges. Although they do not meet the definition given here, these graphs are referred to in graph theory as ordered directed trees (see below).

Definitions

A **tree** is an undirected simple graph G that satisfies any of the following equivalent conditions:

- G is connected and has no cycles.
- G has no cycles, and a simple cycle is formed if any edge is added to G.
- G is connected, and it is not connected anymore if any edge is removed from G.
- G is connected and the 3-vertex complete graph K_3 is not a minor of G.
- Any two vertices in G can be connected by a unique simple path.

If G has finitely many vertices, say n of them, then the above statements are also equivalent to any of the following conditions:

- G is connected and has $n - 1$ edges.
- G has no simple cycles and has $n - 1$ edges.

An **irreducible** (or *series-reduced*) tree is a tree in which there is no vertex of degree 2.

A **forest** is an undirected graph, all of whose connected components are trees; in other words, the graph consists of a disjoint union of trees. Equivalently, a forest is an undirected cycle-free graph. As special cases, an empty graph, a single tree, and the discrete graph on a set of vertices (that is, the graph with these vertices that has no edges), all are examples of forests.

The term **hedge** sometimes refers to an ordered sequence of trees.

A **polytree** or **oriented tree** is a directed graph with at most one undirected path between any two vertices. In other words, a polytree is a directed acyclic graph for which there are no undirected cycles either.

A **directed tree** is a directed graph which would be a tree if the directions on the edges were ignored. Some authors restrict the phrase to the case where the edges are all directed towards a particular vertex, or all directed away from a particular vertex (see arborescence).

A tree is called a **rooted tree** if one vertex has been designated the **root**, in which case the edges have a natural orientation, *towards* or *away* from the root. The *tree-order* is the partial ordering on the vertices of a tree with $u \leq v$ if and only if the unique path from the root to v passes through u. A rooted tree which is a subgraph of some graph G is a **normal tree** if the ends of every edge in G are comparable in this tree-order whenever those ends are vertices of the tree (Diestel 2005, p. 15). Rooted trees, often with additional structure such as ordering of the neighbors at each vertex, are a key data structure in computer science; see tree data structure. In a context where trees are supposed to have a root, a tree without any designated root is called a **free tree**.

In a rooted tree, the **parent** of a vertex is the vertex connected to it on the path to the root; every vertex except the root has a unique parent. A **child** of a vertex v is a vertex of which v is the parent. A **leaf** is a vertex without children.

A **labeled tree** is a tree in which each vertex is given a unique label. The vertices of a labeled tree on n vertices are typically given the labels 1, 2, …, n. A **recursive tree** is a labeled rooted tree where the vertex labels respect the tree order (*i.e.*, if $u < v$ for two vertices u and v, then the label of u is smaller than the label of v).

An **ordered tree** is a rooted tree for which an ordering is specified for the children of each vertex.

An **n-ary tree** is a rooted tree for which each vertex which is not a leaf has at most n children. *2-ary trees* are sometimes called binary trees, while *3-ary trees* are sometimes called ternary trees.

A **terminal vertex** of a tree is a vertex of degree 1. In a rooted tree, the leaves are all terminal vertices; additionally, the root, if not a leaf itself, is a terminal vertex if it has precisely one child.

Example

The example tree shown to the right has 6 vertices and $6 - 1 = 5$ edges. The unique simple path connecting the vertices 2 and 6 is 2-4-5-6.

Facts

- Every tree is a bipartite graph and a median graph. Every tree with only countably many vertices is a planar graph.
- Every connected graph G admits a spanning tree, which is a tree that contains every vertex of G and whose edges are edges of G.
- Every connected graph with only countably many vertices admits a normal spanning tree (Diestel 2005, Prop. 8.2.4).
- There exist connected graphs with uncountably many vertices which do not admit a normal spanning tree (Diestel 2005, Prop. 8.5.2).
- Every finite tree with n vertices, with $n > 1$, has at least two terminal vertices. This minimal number of terminal vertices is characteristic of path graphs; the maximal number, $n - 1$, is attained by star graphs.
- For any three vertices in a tree, the three paths between them have exactly one vertex in common.

Enumeration

Given n labeled vertices, there are n^{n-2} different ways to connect them to make a tree. This result is called Cayley's formula. It can be proven by first showing that the number of trees with n vertices of degree $d_1, d_2, \ldots, d_n$ is the multinomial coefficient

$$\binom{n-2}{d_1-1, d_2-1, \ldots, d_n-1}.$$

An alternative proof uses Prüfer sequences. This is the special case for complete graphs of a more general problem, counting the number of spanning trees in an undirected graph, which can be achieved by computing a determinant according to the matrix tree theorem. The similar problem of counting all the subtrees regardless of size has been shown to be #P-complete in the general case (Jerrum (1994)).

Counting the number of unlabeled free trees is a harder problem. No closed formula for the number $t(n)$ of trees with n vertices up to graph isomorphism is known. The first few values of $t(n)$ are 1, 1, 1, 1, 2, 3, 6, 11, 23, 47, 106, 235, 551, 1301, 3159, ... (sequence A000055 [1] in OEIS). Otter (1948) proved that

$$t(n) \sim C\alpha^n n^{-5/2} \quad \text{as } n \to \infty,$$

with $C = 0.534949606\ldots$ and $\alpha = 2.99557658565\ldots$ (Here, $f \sim g$ means that $\lim_{n\to\infty} f/g = 1$, where $f = t(n)$ and $g = C\alpha^n n^{-5/2}$). Similarly, he showed that for the number $r(n)$ of unlabeled rooted trees with n vertices holds

$$r(n) \sim D\alpha^n n^{-5/2} \quad \text{as } n \to \infty,$$

with $D = 0.43992401257\ldots$ and α the same as above (cf. Knuth (1997), Chap. 2.3.4.4 and Flajolet & Sedgewick (2009), Chap. VII.5).

Types of trees

A star is a tree in which there is only one internal node and $n - 1$ leaves; that is, a star is a tree with as many leaves as possible. A tree with two leaves, the fewest possible, is a path graph. If all nodes in a tree are within distance one of a central path, then the tree is a caterpillar tree. If all nodes are within distance two of a central path, then the tree is a lobster.

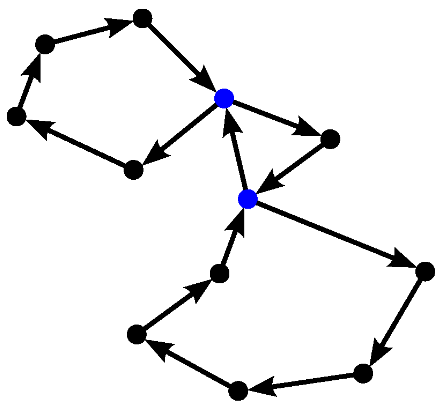
References

- Diestel, Reinhard (2005), *Graph Theory* [2] (3rd ed.), Berlin, New York: Springer-Verlag, ISBN 978-3-540-26183-4.
- Flajolet, Philippe; Sedgewick, Robert (2009), *Analytic Combinatorics*, Cambridge University Press, ISBN 9780521898065
- Donald E. Knuth. *The Art of Computer Programming Volume 1: Fundamental Algorithms*. Addison-Wesley Professional; 3rd edition (November 14, 1997).
- Jerrum, Mark (1994), "Counting trees in a graph is #P-complete", *Information Processing Letters* **51** (3): 111–116, doi:10.1016/0020-0190(94)00085-9, ISSN 0020-0190.
- Otter, Richard (1948), "The Number of Trees" [3], *Annals of Mathematics, Second Series* **49** (3): 583–599, doi:10.2307/1969046.

Path (graph theory)

In graph theory, a **path** in a graph is a sequence of vertices such that from each of its vertices there is an edge to the next vertex in the sequence. A path may be infinite, but a finite path always has a first vertex, called its *start vertex*, and a last vertex, called its *end vertex*. Both of them are called *end or terminal vertices* of the path. The other vertices in the path are *internal vertices*. A **cycle** is a path such that the start vertex and end vertex are the same. Note that the choice of the start vertex in a cycle is arbitrary.

Paths and cycles are fundamental concepts of graph theory, described in the introductory sections of most graph theory texts. See e.g. Bondy and Murty (1976), Gibbons (1985), or Diestel (2005). Korte et al. (1990) cover more advanced algorithmic topics concerning paths in graphs.


A directed cycle. Without the arrows, it is just a cycle. This is not a simple cycle, since the blue vertices are used twice.

Different types of path

The same concepts apply both to undirected graphs and directed graphs, with the edges being directed from each vertex to the following one. Often the terms *directed path* and *directed cycle* are used in the directed case.

A path with no repeated vertices is called a **simple path**, and a cycle with no repeated vertices or edges aside from the necessary repetition of the start and end vertex is a **simple cycle**. In modern graph theory, most often "simple" is implied; i.e., "cycle" means "simple cycle" and "path" means "simple path", but this convention is not always observed, especially in applied graph theory. Some authors (e.g. Bondy and Murty 1976) use the term "walk" for a path in which vertices or edges may be repeated, and reserve the term "path" for what is here called a simple path.

A path such that no graph edges connect two nonconsecutive path vertices is called an induced path.

A simple cycle that includes every vertex, without repetition, of the graph is known as a Hamiltonian cycle.

A cycle with just one edge removed in the corresponding spanning tree of the original graph is known as a Fundamental cycle.

Two paths are *independent* (alternatively, *internally vertex-disjoint*) if they do not have any internal vertex in common.

The *length* of a path is the number of edges that the path uses, counting multiple edges multiple times. The length can be zero for the case of a single vertex.

A weighted graph associates a value (*weight*) with every edge in the graph. The *weight of a path* in a weighted graph is the sum of the weights of the traversed edges. Sometimes the words *cost* or *length* are used instead of weight.

See also

- Glossary of graph theory
- Shortest path problem
- Traveling salesman problem
- Cycle space

References

- Bondy, J. A.; Murty, U. S. R. (1976). *Graph Theory with Applications* [1]. North Holland. pp. 12–21. ISBN 0-444-19451-7.
- Diestel, Reinhard (2005). *Graph Theory* [2] (3rd ed. ed.). Graduate Texts in Mathematics, vol. 173, Springer-Verlag. pp. 6–9. ISBN 3-540-26182-6.
- Gibbons, A. (1985). *Algorithmic Graph Theory*. Cambridge University Press. pp. 5–6. ISBN 0-521-28881-9.
- Korte, Bernhard; Lovász, László; Prömel, Hans Jürgen; Schrijver, Alexander (Eds.) (1990). *Paths, Flows, and VLSI-Layout*. Algorithms and Combinatorics 9, Springer-Verlag. ISBN 0-387-52685-4.

Glossary of graph theory

Graph theory is a growing area in mathematical research, and has a large specialized vocabulary. Some authors use the same word with different meanings. Some authors use different words to mean the same thing. This page attempts to keep up with current usage.

Basics

A **graph** *G* consists of two types of elements, namely *vertices* and *edges*. Every edge has two *endpoints* in the set of vertices, and is said to **connect** or **join** the two endpoints. An edge can thus be defined as a set of two vertices (or an ordered pair, in the case of a **directed graph** - see Section Direction).

Alternative models of graphs exist; e.g., a graph may be thought of as a Boolean binary function over the set of vertices or as a square (0,1)-matrix.

A **vertex** is simply drawn as a *node* or a *dot*. The **vertex set** of *G* is usually denoted by $V(G)$, or V when there is no danger of confusion. The **order** of a graph is the number of its vertices, i.e. $|V(G)|$.

An **edge** (a set of two elements) is drawn as a *line* connecting two vertices, called **endpoint** or (less often) **endvertices**. An edge with endvertices x and y is denoted by xy (without any symbol in between). The **edge set** of *G* is usually denoted by $E(G)$, or E when there is no danger of confusion.

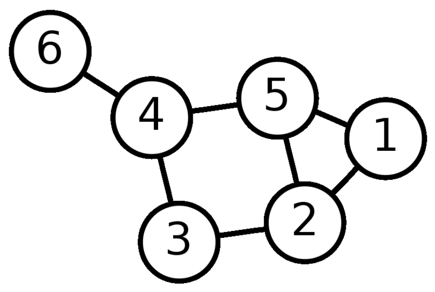
The **size** of a graph is the number of its edges, i.e. $|E(G)|$.

A **loop** is an edge whose endpoints are the same vertex. A **link** has two distinct endvertices. An edge is **multiple** if there is another edge with the same endvertices; otherwise it is **simple**. The **multiplicity of an edge** is the number of multiple edges sharing the same endvertices; the **multiplicity of a graph**, the maximum multiplicity of its edges. A graph is a **simple graph** if it has no multiple edges or loops, a **multigraph** if it has multiple edges, but no loops, and a **multigraph** or **pseudograph** if it contains both multiple edges and loops (the literature is highly inconsistent). When stated without any qualification, a graph is almost always assumed to be simple—one has to judge from the context.

Graphs whose edges or vertices have names or labels are known as **labeled**, those without as **unlabeled**. Graphs with labeled vertices only are **vertex-labeled**, those with labeled edges only are **edge-labeled**. The difference between a labeled and an unlabeled graph is that the latter has no specific set of vertices or edges; it is regarded as another way to look upon an isomorphism type of graphs. (Thus, this usage distinguishes between graphs with identifiable vertex or edge sets on the one hand, and isomorphism types or classes of graphs on the other.)

(**Graph labeling** usually refers to the assignment of labels (usually natural numbers, usually distinct) to the edges and vertices of a graph, subject to certain rules depending on the situation. This should not be confused with a graph's merely having distinct labels or names on the vertices.)

A **hyperedge** is an edge that is allowed to take on any number of vertices, possibly more than 2. A graph that allows any hyperedge is called a **hypergraph**. A simple graph can be considered a special case of the hypergraph, namely the 2-uniform hypergraph. However, when stated without any qualification, an edge is always assumed to consist of at most 2 vertices, and a graph is never confused with a hypergraph.



A labeled simple graph with vertex set *V* = {1, 2, 3, 4, 5, 6} and edge set *E* = .

A **non-edge** (or **anti-edge**) is an edge that is not present in the graph. More formally, for two vertices u and v, $\{u, v\}$ is a non-edge in a graph G whenever $\{u, v\}$ is not an edge in G. This means that there is either no edge between the two vertices or (for directed graphs) at most one of (u, v) and (v, u) from v is an arc in G.
Occasionally the term **cotriangle** or **anti-triangle** is used for a set of three vertices none of which are connected.

The **complement** $\bar{G}$ of a graph *G* is a graph with the same vertex set as *G* but with an edge set such that *xy* is an edge in $\bar{G}$ if and only if *xy* is not an edge in *G*.

An **edgeless graph** or **empty graph** or **null graph** is a graph with zero or more vertices, but no edges. The **empty graph** or **null graph** may also be the graph with no vertices and no edges. If it is a graph with no edges and any number n of vertices, it may be called the **null graph on n vertices**. (There is no consistency at all in the literature.)

A graph is **infinite** if it has infinitely many vertices or edges or both; otherwise the graph is **finite**. An infinite graph where every vertex has finite degree is called **locally finite**. When stated without any qualification, a graph is usually assumed to be finite. See also continuous graph.

Two graphs *G* and *H* are said to be **isomorphic**, denoted by *G* ~ *H*, if there is a one-to-one correspondence, called an **isomorphism**, between the vertices of the graph such that two vertices are adjacent in *G* if and only if their corresponding vertices are adjacent in *H*. Likewise, a graph *G* is said to be **homomorphic** to a graph *H* if there is a mapping, called a **homomorphism**, from *V*(*G*) to *V*(*H*) such that if two vertices are adjacent in *G* then their corresponding vertices are adjacent in *H*.

Subgraphs

A **subgraph** of a graph *G* is a graph whose vertex set is a subset of that of *G*, and whose adjacency relation is a subset of that of *G* restricted to this subset. In the other direction, a **supergraph** of a graph *G* is a graph of which *G* is a subgraph. We say a graph *G* **contains** another graph *H* if some subgraph of *G* is *H* or is isomorphic to *H*.

A subgraph *H* is a **spanning subgraph**, or **factor**, of a graph *G* if it has the same vertex set as *G*. We say *H* spans *G*.

A subgraph *H* of a graph *G* is said to be **induced** if, for any pair of vertices *x* and *y* of H, *xy* is an edge of *H* if and only if *xy* is an edge of G. In other words, *H* is an induced subgraph of *G* if it has exactly the edges that appear in *G* over the same vertex set. If the vertex set of *H* is the subset *S* of *V(G)*, then *H* can be written as *G*[*S*] and is said to be **induced by *S***.

A graph that does *not* contain *H* as an induced subgraph is said to be ***H*-free**.

A **universal graph** in a class ***K*** of graphs is a simple graph in which every element in ***K*** can be embedded as a subgraph.

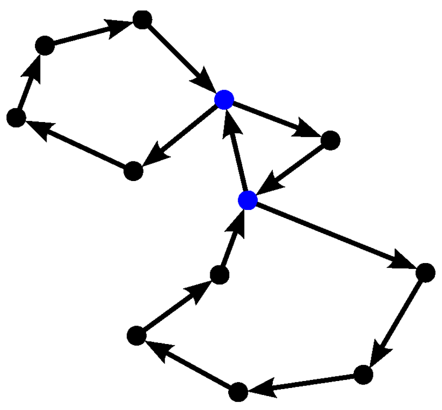
Walks

A **walk** is an alternating sequence of vertices and edges, beginning and ending with a vertex, where each vertex is incident to both the edge that precedes it and the edge that follows it in the sequence, and where the vertices that precede and follow an edge are the end vertices of that edge. A walk is **closed** if its first and last vertices are the same, and **open** if they are different.

The **length** l of a walk is the number of edges that it uses. For an open walk, $l = n-1$, where n is the number of vertices visited (a vertex is counted each time it is visited). For a closed walk, $l = n$ (the start/end vertex is listed twice, but is not counted twice). In the example graph, (1, 2, 5, 1, 2, 3) is an open walk with length 5, and (4, 5, 2, 1, 5, 4) is a closed walk of length 5.

A **trail** is a walk in which all the edges are distinct. A closed trail has been called a **tour** or **circuit**, but these are not universal, and the latter is often reserved for a regular subgraph of degree two.

Traditionally, a **path** referred to what is now usually known as an *open walk*. Nowadays, when stated without any qualification, a path is usually understood to be **simple**, meaning that no vertices (and thus no edges) are repeated. (The term **chain** has also been used to refer to a walk in which all vertices and edges are distinct.) In the example graph, (5, 2, 1) is a path of length 2. The closed equivalent to this type of walk, a walk that starts and ends at the same vertex but otherwise has no repeated vertices or edges, is called a **cycle**. Like *path*, this term traditionally referred to any closed walk, but now is usually understood to be simple by definition. In the example graph, (1, 5, 2, 1) is a cycle of length 3. (A cycle, unlike a path, is not allowed to have length 0.) Paths and cycles of n vertices are often denoted by P_n and C_n, respectively. (Some authors use the length instead of the number of vertices, however.)


A directed cycle. Without the arrows, it is just a cycle. This is not a simple cycle, since the blue vertices are used twice.

C_1 is a **loop**, C_2 is a **digon** (a pair of parallel undirected edges in a multigraph, or a pair of antiparallel edges in a directed graph), and C_3 is called a **triangle**.

A cycle that has odd length is an **odd cycle**; otherwise it is an **even cycle**. One theorem is that a graph is bipartite if and only if it contains no odd cycles. (See complete bipartite graph.)

A graph is **acyclic** if it contains no cycles; **unicyclic** if it contains exactly one cycle; and **pancyclic** if it contains cycles of every possible length (from 3 to the order of the graph).

The **girth** of a graph is the length of a shortest (simple) cycle in the graph; and the **circumference**, the length of a longest (simple) cycle. The girth and circumference of an acyclic graph are defined to be infinity ∞.

A path or cycle is **Hamiltonian** (or *spanning*) if it uses all vertices exactly once. A graph that contains a Hamiltonian path is **traceable**; and one that contains a Hamiltonian path for any given pair of (distinct) end vertices is a **Hamiltonian connected graph**. A graph that contains a Hamiltonian cycle is a **Hamiltonian graph**.

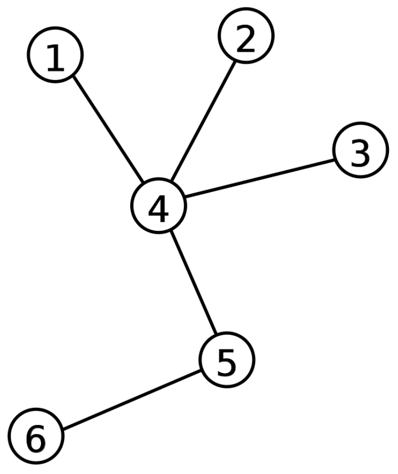
A trail or circuit (or cycle) is **Eulerian** if it uses all edges precisely once. A graph that contains an Eulerian trail is **traversable**. A graph that contains an Eulerian circuit is an **Eulerian graph**.

Two paths are **internally disjoint** (some people call it *independent*) if they do not have any vertex in common, except the first and last ones.

A **theta graph** is the union of three internally disjoint (simple) paths that have the same two distinct end vertices. A **theta$_0$ graph** has seven vertices which can be arranged as the vertices of a regular hexagon plus an additional vertex in the center. The eight edges are the perimeter of the hexagon plus one diameter.

Trees

A **tree** is a connected acyclic simple graph. A vertex of degree 1 is called a **leaf**, or *pendant vertex*. An edge incident to a leaf is a **leaf edge**, or *pendant edge*. (Some people define a leaf edge as a *leaf* and then define a *leaf vertex* on top of it. These two sets of definitions are often used interchangeably.) A non-leaf vertex is an **internal vertex**. Sometimes, one vertex of the tree is distinguished, and called the **root**; in this case, the tree is called **rooted**. Rooted trees are often treated as **directed acyclic graphs** with the edges pointing away from the root.



A labeled tree with 6 vertices and 5 edges.

A **subtree** of the tree *T* is a connected subgraph of *T*.

A **forest** is an acyclic simple graph.

A **subforest** of the forest *F* is a subgraph of *F*.

A **spanning tree** is a spanning subgraph that is a tree. Every graph has a spanning forest. But only a connected graph has a spanning tree.

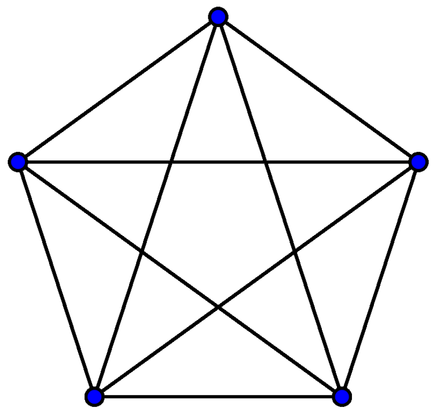
A special kind of tree called a **star** is $K_{1,k}$. An induced star with 3 edges is a **claw**.

A **caterpillar** is a tree in which all non-leaf nodes form a single path.

A ***k*-ary** tree is a rooted tree in which every internal vertex has *k children*. A 1-ary tree is just a path. A 2-ary tree is also called a **binary tree**.

Cliques

The **complete graph** K_n of order *n* is a simple graph with *n* vertices in which every vertex is adjacent to every other. The example graph to the right is complete. The complete graph on *n* vertices is often denoted by K_n. It has $n(n-1)/2$ edges (corresponding to all possible choices of pairs of vertices).


K_5, a complete graph. If a subgraph looks like this, the vertices in that subgraph form a clique of size 5.

A **clique** in a graph is a set of pairwise adjacent vertices. Since any subgraph induced by a clique is a complete subgraph, the two terms and their notations are usually used interchangeably. A ***k*-clique** is a clique of order *k*. In the example graph above, vertices 1, 2 and 5 form a 3-clique, or a *triangle*. A maximal clique is a clique that is not a subset of any other clique (some authors reserve the term clique for maximal cliques).

The **clique number** ω(*G*) of a graph *G* is the order of a largest clique in *G*.

Strongly connected component

A related but weaker concept is that of a *strongly connected component*. Informally, a strongly connected component of a directed graph is a subgraph where all nodes in the subgraph are reachable by all other nodes in the subgraph. Reachability between nodes is established by the existence of a *path* between the nodes.

A directed graph can be decomposed into strongly connected components by running the depth-first search (DFS) algorithm twice: first, on the graph itself and next on the *transpose* of the graph in decreasing order of the finishing times of the first DFS. Given a directed graph *G*, the transpose G_T is the graph *G* with all the edge directions reversed.

Knots

A **knot** in a directed graph is a collection of vertices and edges with the property that every vertex in the knot has outgoing edges, and all outgoing edges from vertices in the knot terminate at other vertices in the knot. Thus it is impossible to leave the knot while following the directions of the edges.

Minors

A *minor* $G_2 = (V_2, E_2)$ of $G_1 = (V_1, E_1)$ is an injection from V_2 to V_1 such that every edge in E_2 corresponds to a path (disjoint from all other such paths) in G_1 such that every vertex in V_1 is in one or more paths, or is part of the injection from V_1 to V_2. This can alternatively be phrased in terms of *contractions*, which are operations which collapse a path and all vertices on it into a single edge (see Minor (graph theory)).

Embedding

An *embedding* $G_2 = (V_2, E_2)$ of $G_1 = (V_1, E_1)$ is an injection from V_2 to V_1 such that every edge in E_2 corresponds to a path (disjoint from all other such paths) in G_1.

Adjacency and degree

In graph theory, degree, especially that of a vertex, is usually a *measure of immediate adjacency*.

An edge connects two vertices; these two vertices are said to be **incident** to that edge, or, equivalently, that edge incident to those two vertices. All degree-related concepts have to do with adjacency or incidence.

The **degree**, or *valency*, $d_G(v)$ of a vertex *v* in a graph *G* is the number of edges incident to *v*, with loops being counted twice. A vertex of degree 0 is an **isolated vertex**. A vertex of degree 1 is a leaf. In the labelled simple graph example, vertices 1 and 3 have a degree of 2, vertices 2, 4 and 5 have a degree of 3, and vertex 6 has a degree of 1. If *E* is finite, then the total sum of vertex degrees is equal to twice the number of edges.

The **total degree** of a graph is equal to two times the number of edges, loops included. This means that for a graph with 3 vertices with each vertex having a degree of two (i.e. a triangle) the total degree would be six (e.g. 3 x 2 = 6). The general formula for this is **total degree = 2n** where **n = number of edges**.

A **degree sequence** is a list of degrees of a graph in non-increasing order (e.g. $d_1 \geq d_2 \geq \ldots \geq d_n$). A sequence of non-increasing integers is **realizable** if it is a degree sequence of some graph.

Two vertices *u* and *v* are called **adjacent** if an edge exists between them. We denote this by $u \sim v$ or $u \downarrow v$. In the above graph, vertices 1 and 2 are adjacent, but vertices 2 and 4 are not. The set of **neighbors** of *v*, that is, vertices adjacent to *v* not including *v* itself, forms an induced subgraph called the **(open) neighborhood** of *v* and denoted $N_G(v)$. When *v* is also included, it is called a **closed neighborhood** and denoted by $N_G[v]$. When stated without any qualification, a neighborhood is assumed to be open. The subscript *G* is usually dropped when there is no danger of confusion; the same neighborhood notation may also be used to refer to sets of adjacent vertices rather than the corresponding induced subgraphs. In the example graph, vertex 1 has two neighbors: vertices 2 and 5. For a simple graph, the number of neighbors that a vertex has coincides with its degree.

A **dominating set** of a graph is a vertex subset whose closed neighborhood includes all vertices of the graph. A vertex *v* **dominates** another vertex *u* if there is an edge from *v* to *u*. A vertex subset *V* **dominates** another vertex subset *U* if every vertex in *U* is adjacent to some vertex in *V*. The minimum size of a dominating set is the **domination number** $\gamma(G)$.

In computers, a finite, directed or undirected graph (with *n* vertices, say) is often represented by its **adjacency matrix**: an *n*-by-*n* matrix whose entry in row *i* and column *j* gives the number of edges from the *i*-th to the *j*-th vertex.

Spectral graph theory studies relationships between the properties of the graph and its adjacency matrix.

The **maximum degree** $\Delta(G)$ of a graph *G* is the largest degree over all vertices; the **minimum degree** $\delta(G)$, the smallest.

A graph in which every vertex has the same degree is **regular**. It is ***k*-regular** if every vertex has degree *k*. A 0-regular graph is an independent set. A 1-regular graph is a matching. A 2-regular graph is a vertex disjoint union of cycles. A 3-regular graph is said to be **cubic**, or *trivalent*.

A ***k*-factor** is a *k*-regular spanning subgraph. A 1-factor is a **perfect matching**. A partition of edges of a graph into *k*-factors is called a ***k*-factorization**. A ***k*-factorable graph** is a graph that admits a *k*-factorization.

A graph is **biregular** if it has unequal maximum and minimum degrees and every vertex has one of those two degrees.

A **strongly regular graph** is a regular graph such that any adjacent vertices have the same number of common neighbors as other adjacent pairs and that any nonadjacent vertices have the same number of common neighbors as other nonadjacent pairs.

Independence

In graph theory, the word *independent* usually carries the connotation of *pairwise disjoint* or *mutually nonadjacent*. In this sense, independence is a form of *immediate nonadjacency*. An **isolated vertex** is a vertex not incident to any edges. An **independent set**, or *coclique*, or *stable set* or *staset*, is a set of vertices of which no pair is adjacent. Since the graph induced by any independent set is an empty graph, the two terms are usually used interchangeably. In the example above, vertices 1, 3, and 6 form an independent set; and 3, 5, and 6 form another one.

Two subgraphs are **edge disjoint** if they have no edges in common. Similarly, two subgraphs are **vertex disjoint** if they have no vertices (and thus, also no edges) in common. Unless specified otherwise, a set of **disjoint subgraphs** are assumed to be pairwise vertex disjoint.

The **independence number** $\alpha(G)$ of a graph *G* is the size of the largest independent set of *G*.

A graph can be **decomposed** into independent sets in the sense that the entire vertex set of the graph can be partitioned into pairwise disjoint independent subsets. Such independent subsets are called **partite sets**, or simply *parts*.

A graph that can be decomposed into two partite sets but not fewer is **bipartite**; three sets but not fewer, **tripartite**; *k* sets but not fewer, ***k*-partite**; and an unknown number of sets, **multipartite**. An 1-partite graph is the same as an independent set, or an empty graph. A 2-partite graph is the same as a bipartite graph. A graph that can be decomposed into *k* partite sets is also said to be ***k*-colourable**.

A **complete multipartite** graph is a graph in which vertices are adjacent if and only if they belong to different partite sets. A *complete bipartite graph* is also referred to as a **biclique**; if its partite sets contain *n* and *m* vertices, respectively, then the graph is denoted $K_{n,m}$.

A *k*-partite graph is **semiregular** if each of its partite sets has a uniform degree; **equipartite** if each partite set has the same size; and **balanced *k*-partite** if each partite set differs in size by at most 1 with any other.

The **matching number** $\alpha'(G)$ of a graph G is the size of a largest **matching**, or pairwise vertex disjoint edges, of G.
A *spanning matching*, also called a **perfect matching** is a matching that covers all vertices of a graph.

Connectivity

Connectivity extends the concept of adjacency and is essentially a form (and measure) of *concatenated adjacency*.

If it is possible to establish a path from any vertex to any other vertex of a graph, the graph is said to be **connected**; otherwise, the graph is **disconnected**. A graph is **totally disconnected** if there is no path connecting any pair of vertices. This is just another name to describe an empty graph or independent set.

A **cut vertex**, or *articulation point*, is a vertex whose removal disconnects the remaining subgraph. A **cut set**, or *vertex cut* or *separating set*, is a set of vertices whose removal disconnects the remaining subgraph. A *bridge* is an analogous edge (see below).

If it is always possible to establish a path from any vertex to every other even after removing any k - 1 vertices, then the graph is said to be ***k*-vertex-connected** or ***k*-connected**. Note that a graph is k-connected if and only if it contains k internally disjoint paths between any two vertices. The example graph above is connected (and therefore 1-connected), but not 2-connected. The **vertex connectivity** or **connectivity** $\kappa(G)$ of a graph G is the minimum number of vertices that need to be removed to disconnect G. The complete graph K_n has connectivity n - 1 for $n > 1$; and a disconnected graph has connectivity 0.

In network theory, a **giant component** is a connected subgraph that contains a majority of the entire graph's nodes.

A **bridge**, or *cut edge* or *isthmus*, is an edge whose removal disconnects a graph. (For example, all the edges in a tree are bridges.) A **disconnecting set** is a set of edges whose removal increases the number of components. An **edge cut** is the set of all edges which have one vertex in some proper vertex subset S and the other vertex in $V(G)\backslash S$. Edges of K_3 form a disconnecting set but not an edge cut. Any two edges of K_3 form a minimal disconnecting set as well as an edge cut. An edge cut is necessarily a disconnecting set; and a minimal disconnecting set of a nonempty graph is necessarily an edge cut. A **bond** is a minimal (but not necessarily minimum), nonempty set of edges whose removal disconnects a graph. A *cut vertex* is an analogous vertex (see above).

A graph is ***k*-edge-connected** if any subgraph formed by removing any k - 1 edges is still connected. The **edge connectivity** $\kappa'(G)$ of a graph G is the minimum number of edges needed to disconnect G. One well-known result is that $\kappa(G) \leq \kappa'(G) \leq \delta(G)$.
A **component** is a maximally connected subgraph. A **block** is either a maximally 2-connected subgraph, a bridge (together with its vertices), or an isolated vertex. A **biconnected component** is a 2-connected component.

An **articulation point** (also known as a *separating vertex*) of a graph is a vertex whose removal from the graph increases its number of connected components. A biconnected component can be defined as a subgraph induced by a maximal set of nodes that has no separating vertex.

Distance

The **distance** $d_G(u, v)$ between two (not necessary distinct) vertices u and v in a graph G is the length of a shortest path between them. The subscript G is usually dropped when there is no danger of confusion. When u and v are identical, their distance is 0. When u and v are unreachable from each other, their distance is defined to be infinity ∞.

The **eccentricity** $\varepsilon_G(v)$ of a vertex v in a graph G is the maximum distance from v to any other vertex. The **diameter** diam(G) of a graph G is the maximum eccentricity over all vertices in a graph; and the **radius** rad(G), the minimum. When there are two components in G, diam(G) and rad(G) defined to be infinity ∞. Trivially, diam(G) $\leq$ 2 rad(G). Vertices with maximum eccentricity are called **peripheral vertices**. Vertices of minimum eccentricity form the **center**. A tree has at most two center vertices.

The **Wiener index of a vertex** v in a graph G, denoted by $W_G(v)$ is the sum of distances between v and all others. The **Wiener index of a graph** G, denoted by $W(G)$, is the sum of distances over all pairs of vertices. An undirected graph's **Wiener polynomial** is defined to be $\Sigma\ q^{d(u,v)}$ over all unordered pairs of vertices u and v. Wiener index and Wiener polynomial are of particular interest to mathematical chemists.

The ***k*-th power** G^k of a graph G is a supergraph formed by adding an edge between all pairs of vertices of G with distance at most k. A *second power* of a graph is also called a **square**.

A ***k*-spanner** is a spanning subgraph in which every two vertices are at most k times as far apart on S than on G. The number k is the **dilation**. k-spanner is used for studying geometric network optimization.

Genus

A **crossing** is a pair of intersecting edges. A graph is **embeddable** on a surface if its vertices and edges can be arranged on it without any crossing. The **genus** of a graph is the lowest genus of any surface on which the graph can embed.

A **planar graph** is one which *can be* drawn on the (Euclidean) plane without any crossing; and a **plane graph**, one which *is* drawn in such fashion. In other words, a planar graph is a graph of genus 0. The example graph is planar; the complete graph on n vertices, for $n> 4$, is not planar. Also, a tree is necessarily a planar graph.

When a graph is drawn without any crossing, any cycle that surrounds a region without any edges reaching from the cycle into the region forms a **face**. Two faces on a plane graph are **adjacent** if they share a common edge. A **dual**, or *planar dual* when the context needs to be clarified, G^* of a plane graph G is a graph whose vertices represent the faces, including any outerface, of G and are adjacent in G^* if and only if their corresponding faces are adjacent in G. The dual of a planar graph is always a planar *pseudograph* (e.g. consider the dual of a triangle). In the familiar case of a 3-connected simple planar graph G (isomorphic to a convex polyhedron P), the dual G^* is also a 3-connected simple planar graph (and isomorphic to the dual polyhedron P^*).

Furthermore, since we can establish a sense of "inside" and "outside" on a plane, we can identify an "outermost" region that contains the entire graph if the graph does not cover the entire plane. Such outermost region is called an **outer face**. An **outerplanar graph** is one which *can be* drawn in the planar fashion such that its vertices are all adjacent to the outer face; and an **outerplane graph**, one which *is* drawn in such fashion.

The minimum number of crossings that must appear when a graph is drawn on a plane is called the **crossing number**.

The minimum number of planar graphs needed to cover a graph is the **thickness** of the graph.

Weighted graphs and networks

A **weighted graph** associates a label (**weight**) with every edge in the graph. Weights are usually real numbers. They may be restricted to rational numbers or integers. Certain algorithms require further restrictions on weights; for instance, Dijkstra's algorithm works properly only for positive weights. The **weight of a path** or the **weight of a tree** in a weighted graph is the sum of the weights of the selected edges. Sometimes a non-edge is labeled by a special weight representing infinity. Sometimes the word **cost** is used instead of weight. When stated without any qualification, a graph is always assumed to be unweighted. In some writing on graph theory the term **network** is a synonym for a **weighted graph**. A network may be directed or undirected, it may contain special vertices (nodes), such as **source** or **sink**. The classical network problems include:

- minimum cost spanning tree,
- shortest paths,
- maximal flow (and the max-flow min-cut theorem)

Direction

A **directed arc**, or *directed edge*, is an ordered pair of endvertices that can be represented graphically as an arrow drawn between the endvertices. In such an ordered pair the first vertex is called the *initial vertex* or **tail**; the second one is called the *terminal vertex* or **head** (because it appears at the arrow head). An **undirected edge** disregards any sense of direction and treats both endvertices interchangeably. A **loop** in a digraph, however, keeps a sense of direction and treats both head and tail identically. A set of arcs are **multiple**, or *parallel*, if they share the same head and the same tail. A pair of arcs are **anti-parallel** if one's head/tail is the other's tail/head. A **digraph**, or *directed graph*, or **oriented graph**, is analogous to an undirected graph except that it contains only arcs. A **mixed graph** may contain both directed and undirected edges; it generalizes both directed and undirected graphs. When stated without any qualification, a graph is almost always assumed to be undirected.

A digraph is called **simple** if it has no loops and at most one arc between any pair of vertices. When stated without any qualification, a digraph is usually assumed to be simple.

In a digraph Γ, we distinguish the **out degree** $d_\Gamma^+(v)$, the number of edges leaving a vertex v, and the **in degree** $d_\Gamma^-(v)$, the number of edges entering a vertex v. If the graph is oriented, the degree $d_\Gamma(v)$ of a vertex v is equal to the sum of its out- and in- degrees. When the context is clear, the subscript Γ can be dropped. Maximum and minimum out degrees are denoted by $\Delta^+(\Gamma)$ and $\delta^+(\Gamma)$; and maximum and minimum in degrees, $\Delta^-(\Gamma)$ and $\delta^-(\Gamma)$.

An **out-neighborhood**, or *successor set*, $N^+_\Gamma(v)$ of a vertex v is the set of tails of arcs going from v. Likewise, an **in-neighborhood**, or *predecessor set*, $N^-_\Gamma(v)$ of a vertex v is the set of heads of arcs going into v.

A **source** is a vertex with 0 in-degree; and a **sink**, 0 out-degree.

A vertex v **dominates** another vertex u if there is an arc from v to u. A vertex subset S is **out-dominating** if every vertex not in S is dominated by some vertex in S; and **in-dominating** if every vertex in S is dominated by some vertex not in S.

A **kernel** is an independent out-dominating set. A digraph is **kernel perfect** if every induced sub-digraph has a kernel.

An **Eulerian digraph** is a digraph with equal in- and out-degrees at every vertex.

The **zweieck** of an undirected edge $e = (u, v)$ is the pair of diedges (u, v) and (v, u) which form the simple dicircuit.
An **orientation** is an assignment of directions to the edges of an undirected or partially directed graph. When stated without any qualification, it is usually assumed that all undirected edges are replaced by a directed one in an orientation. Also, the underlying graph is usually assumed to be undirected and simple.

A **tournament** is a digraph in which each pair of vertices is connected by exactly one arc. In other words, it is an oriented complete graph.

A **directed path**, or just a *path* when the context is clear, is an oriented simple path such that all arcs go the same direction, meaning all internal vertices have in- and out-degrees 1. A vertex v is **reachable** from another vertex u if there is a directed path that starts from u and ends at v. Note that in general the condition that u is reachable from v does not imply that v is also reachable from u.

If v is reachable from u, then u is a **predecessor** of v and v is a **successor** of u. If there is an arc from u to v, then u is a **direct predecessor** of v, and v is a **direct successor** of u.

A digraph is **strongly connected** if every vertex is reachable from every other following the directions of the arcs. On the contrary, a digraph is **weakly connected** if its underlying undirected graph is connected. A weakly connected graph can be thought of as a digraph in which every vertex is "reachable" from every other but not necessarily following the directions of the arcs. A **strong orientation** is an orientation that produces a strongly connected digraph.

A **directed cycle**, or just a *cycle* when the context is clear, is an oriented simple cycle such that all arcs go the same direction, meaning all vertices have in- and out-degrees 1. A digraph is **acyclic** if it does not contain any directed cycle. A finite, acyclic digraph with no isolated vertices necessarily contains at least one source and at least one sink.

An **arborescence**, or *out-tree* or *branching*, is an oriented tree in which all vertices are reachable from a single vertex. Likewise, an *in-tree* is an oriented tree in which a single vertex is reachable from every other one.

Directed acyclic graphs

The partial order structure of directed acyclic graphs (or DAGs) gives them their own terminology.

If there is a directed edge from *u* to *v*, then we say *u* is a **parent** of *v* and *v* is a **child** of *u*. If there is a directed path from *u* to *v*, we say *u* is an **ancestor** of *v* and *v* is a **descendant** of *u*.

The **moral graph** of a DAG is the undirected graph created by adding an (undirected) edge between all parents of the same node (sometimes called *marrying*), and then replacing all directed edges by undirected edges. A DAG is **perfect** if, for each node, the set of parents is complete (i.e. no new edges need to be added when forming the moral graph).

Colouring

Vertices in graphs can be given **colours** to identify or label them. Although they may actually be rendered in diagrams in different colours, working mathematicians generally pencil in numbers or letters (usually numbers) to represent the colours.

Given a graph G (V,E) a ***k*-colouring** of G is a map $\phi : V \rightarrow \{1, \dots, k\}$ with the property that $(u, v) \in E \Rightarrow \phi(u) \neq \phi(v)$ - in other words, every vertex is assigned a colour with the condition that adjacent vertices cannot be assigned the same colour.

The **chromatic number** χ(G) is the smallest *k* for which G has a *k*-colouring.

Given a graph and a colouring, the **colour classes** of the graph are the sets of vertices given the same colour.

Various

A **graph invariant** is a property of a graph *G*, usually a number or a polynomial, that depends only on the isomorphism class of *G*. Examples are the order, genus, chromatic number, and chromatic polynomial of a graph.

References

- [1] Graph Triangulation, Jayson Rome, October 14, 2002 [1]
- Bollobás, Béla (1998). *Modern Graph Theory*. New York: Springer-Verlag. ISBN 0-387-98488-7. [*Packed with advanced topics followed by a historical overview at the end of each chapter.*]
- Diestel, Reinhard (2005). *Graph Theory* [2] (3rd edition ed.). Graduate Texts in Mathematics, vol. 173, Springer-Verlag. ISBN 3-540-26182-6. [*Standard textbook, most basic material and some deeper results, exercises of various difficulty and notes at the end of each chapter; known for being quasi error-free. Free electronic version is available.*]
- West, Douglas B. (2001). *Introduction to Graph Theory* (2ed). Upper Saddle River: Prentice Hall. ISBN 0-13-014400-2. [*Tons of illustrations, references, and exercises. The most complete introductory guide to the subject.*]
- Weisstein, Eric W., "Graph [2]" from MathWorld.
- Zaslavsky, Thomas. Glossary of signed and gain graphs and allied areas. *Electronic Journal of Combinatorics*, Dynamic Surveys in Combinatorics, # DS 8. http://www.combinatorics.org/Surveys/

Article Sources and Contributors

Social network *Source*: http://en.wikipedia.org/w/index.php?oldid=404640258 *Contributors*: 2004-12-29T22:45Z, 2bar, AK Auto, Aapo Laitinen, Abeg92, Abeusher, AbsolutDan, Acm1989, Adam78, AdamRetchless, Alan Au, Alansohn, Alessandriana, Alessandro De Rossi, Alex Kosorukoff, Alxeedo, Ancheta Wis, AndreNatas, Andrevan, Angrysusan, Antandrus, Anthon.Eff, Antipastor, Antiselfpromotion, Antonej, Anurag Garg, Artgirl88, Ashesofwind, Athkalani, Ayumijomori, Azumanga1, BAxelrod, Barek, Barticus88, Bbpen, Beetstra, Bellagio99, Berti, Betacommand, Bhadani, Bigmantonyd, Bill.albing, Billhansen5, Bluemosquito, Boomgate, Boud, Briaboru, Brian0918, Buburuza, Buridan, CAkira, CNHolbein, Calenet, Camoz87, Cansem, Capricorn42, Caseyhelbling, Cathyatscholar360, Cdagnino, CesarB, Cg2916, Chicago andrew, Chocolateboy, Chriscf, Christion123, Ciphergoth, Cleared as filed, Clicketyclack, Cms3rd, Cnawan, Cocal2, Coethnic, Cokoli, Corrossive, Crazycomputers, D, DARTH SIDIOUS 2, DabMachine, Dan53, DancingPhilosopher, Dancter, Danellicus, Danlev, Danski14, Darcfudg, DarwinPeacock, Dasch, Dave Runger, David Gerard, DavidLevinson, Davodd, DeLarge, Deborah909, Deep1256, Deon Steyn, Dhn, Dickreuter, Dillardjj, Dimz793, Discospinster, Dmcgrew1, Doitinpublicpr, DoubleBlue, Douglas R. White, Drderail, Dreadstar, DreamGuy, E31029, EPM, EagleWang, Ebricca, Edcolins, EddieMo, Edward, Effeietsanders, Einphrey, ElKevbo, Elsesteban, Emily GABLE, EmilyChew, Emulsionla, Enchanter, Engineman, Enisten, EnriqueMurillo, Epbr123, Eras-mus, Eredux, Erianna, Eric-Wester, Everyking, Excirial, FatalError, Favonian, Faznar, Federalostt, Ferran.cabrer, Fieldday-sunday, Flewis, Flower Priest, Fox, Franamax, FreplySpang, FreshBreeze, Freud's Genius, Fæ, G S P, GUIDIUnknown, Gaius Cornelius, Gary King, Gerrit, Getcrunk, Getmoreatp, Gilliam, Ginsengbomb, Glenn, Goapsy, Gobeshock Gobochondro Gyanotirtho, Gogo Dodo, GoingBatty, Golubchikav, Goochelaar, Goodoldpolonius, Goodoldpolonius2, GraemeL, Gragus, Grick, Habit 247, Hak686, Hakeem.gadi, Hamadashow, Hariva, Harvey the rabbit, Heisss, Histree, Hmains, Hmrox, Hodzha, Houdani, Hu12, IGeMiNix, Ianmw, Igorberger, IllinoisTeacher, Imersion, Isepellissery, Isis07, Isomorphic, Itzcyndi, IvanLanin, J.delanoy, JForget, JFreeman, JJL, JRR Trollkien, JVans, Jackollie, Jan1nad, JanaDiesner, Jauerback, Jazzguy3333, Jbtrout, Jc3s5h, Jcbutler, Jcoplien, Jehochman, Jelena filipov, JeremyA, Jezramsforth, Jheuristic, Jimernst, Jmundo, Joeboy, Joetroll, JohnDoe0007, JohnUpp, Jonobennett, JoshXF, Joshbuddy, Joshua.hammond, Jpbowen, Jrtayloriv, Jthewombat, Jtneill, JulieWohlberg, Justin534, Jwdietrich2, JzG, Kai-Hendrik, Karada, Kazazz, Kevinshroff, Khalid hassani, Khaydarian, Kiefer.Wolfowitz, Kierant, Kku, Kl4m, Kmouly, KnowledgeOfSelf, Korny O'Near, Korotayev, Koskimaki, Kousu, Ksyrie, Kukini, Kurieeto, Kuru, Kwinkunks, L Kensington, Lambiam, LaurensvanLieshout, LeaveSleaves, Legion fi, Leki, Levineps, LilHelpa, Logos.noreply, Logos.undp, Lordknox, Luisrull, Lvsubram, M0joer1013, MER-C, Maallyis, Madcoverboy, Magioladitis, Malo, Mamawrites, Mark.Crosby, Marktan, Markusvinzent, Martin Fasani, Mashable, Master&Expert, Matt B., Mattisse, Matěj Grabovský, Maurice Carbonaro, Maxelrod, McSly, Mdd, Me1krishna, Meeples, Memming, Mephistophelian, Michael Hardy, Mike Restivo, Mikeblas, Mikiemike, Mind123, Mindmatrix, Mitsubishi Zero, Mmmalexand, Mmontala, Mmorabito67, Mobnews, Mokshjuneja, Mosiah, MrOllie, MultimediaGuru, Mxn, My-dfp, Myasuda, Mydogategodshat, Myrockstar, N5iln, NMChico24, NYNetwork, Nae'blis, Nathanbanton, NawlinWiki, NellieBly, NeniPogarcic, Nero8858, Neurolysis, Nigelcopley, Nihiltres, Nikai, Niteowlneils, NithinBekal, Nnehaa, Nohat, Nowa, Nsaa, Ohnoitsjamie, Oldhamlet, Oliver.perry, Olv 26, OneVeryBadMan, Onorem, Ot, Oxymoron83, PGPirate, Pasquale, Pboodark, Pdixon, Pealmasa, Peasantwarrior, Pelayo el Sabio, Per7, Perfecto, Petiatil, Phanhaitrieu, Phauly, Phidman, Philcomputing, Philthecow, Pill, Piotrus, Pip2andahalf, Pixel, Pleasantville, Pm master, Pne, Polmorry, Postdlf, Ppg90828302187, Prcoulson, Principe massimo, Proofreader77, Psantora, Punkandbarbies, Qiob, Qtoktok, Quietquite1234, Qxz, R Lowry, R'n'B, R3m0t, RJBurkhart, Radicalsubversiv, RadioActive, Ramanbasu, Ramir, Rankiri, Raul Lapeira, Rdepontb, Rdsmith4, Reach Out to the Truth, Realglobalist, Rebecita.angle, Reconsider the static, Recury, Redskinsfan1, Reedy, Refsworldlee, Renice, Rgerstley, Rhobite, Richie.lukas, Rigadoun, Rimapacowa, Rizen, Robin klein, Robykiwi, RockMFR, Romanm, Ronhjones, Ronz, Rosti99, RoyBoy, Rprout520, Rubyji, STOwiki, Sadi Carnot, Salvio giuliano, Sam Hocevar, Sayyid Muhammad Emadi, Sdsds, Sean.hoyland, Seaphoto, Sether, SeventhHell, Shaddack, Shadowjams, ShakataGaNai, Sharon08tam, Shavonbradshaw, Shelliwright, Shergul, Shi Hou, Shoecream, Shrikrishnabhardwaj, SimonD, SimonP, SirReid, SkiLLru, Sky Attacker, Skyebend, Slifty, Slproxy, Sluzzelin, SocialNetworking, Socialgovernance, Socialnetworks, Sogge, Solidcore, Somesh, Sparsefarce, Sroel, Stephenb, SteveWilhelm, Stevietheman, Sudipdasin, Sunray, Superborsuk, SusanLesch, Susko, Syrthiss, TACD, Taeshadow, Tdoug870, Tdunvan, Tech2blog, Tectonicura, Tedstanton, Terryoleary1981, Tesfatsion, Tetracube, Tezpur4u, The Anome, The lorax, Themfromspace, Thomasmeeks, Timclare, Timwi, Tizzey, Tjwood26, Tmp373, Tomsega, Toracle, Tore.opsahl, Tresiden, Treyjp, Trusilver, Tseeker, Typewriter, USchick, Uba33, UkPaolo, Unforgiven24, UnitedStatesian, Uttoransen, Vaughan, Vegaswikian, Veinor, Versageek, Vespristiano, VictorAnyakin, Vicvicvicvic, Viper80, Waldir, Wassermann7, Wavelength, Wayne Slam, WelshAspie, Westendgirl, WgMp, WikiLaurent, Wikiborg, Wikidemon, Wikigregor, Wikiklrsc, Winquan1, Wnjr, Wrs1864, Wtmitchell, Wykis, XEsup, Yuwiepal, ZeroOne, ZimZalaBim, Zzuuzz, 1033 anonymous edits

Social network analysis software *Source*: http://en.wikipedia.org/w/index.php?oldid=404289265 *Contributors*: AVRS, Aidan connolly, Alex rosenberg35, Allentien, Anthon.Eff, Bellagio99, Billhansen5, Billlion, Chowbok, Chrisguyot, ClMueller, Cnorvell, DHunscher, DamarisC, DataAnalyzer, Edill3484, Edward Vielmetti, Executive override, Gergelypalla, Gogo Dodo, Gsavin, Imamhoque, InesMergel, Inspired2apathy, JHunterJ, JanaDiesner, JonHarder, Joshua O'Madadhain, Leycec, LilHelpa, MDeReno, Mamatha srirama, Marc A Smith, Markhuisman, Mebden, Mentatseb, Michael Hardy, Michelran, Nplotkin, Orgnet, PaulHBroome, Perstar, Psantora, Refsworldlee, Rimapacowa, Rjwilmsi, Sampiofune, Sogge, Sophus Bie, Supersonnig, Terrillfrantz, ThaddeusB, Tore.opsahl, Vlad43210, Waldir, 128 anonymous edits

Betweenness *Source*: http://en.wikipedia.org/w/index.php?oldid=295052452 *Contributors*: BAxelrod, Bkkbrad, Borgatts, David Eppstein, Douglas R. White, Eassin, EbrushERB, Ergotius, Fabercap, Flyingspuds, Frozen4322, Giftlite, Harthur, Hathawayc, Headbomb, Icairns, J.delanoy, Kazastankas, Kku, Loodog, Mdvs, Mgwalker, Michael Hardy, Michael Rogers, Minimac, Nburden, Piotrus, RafG, Reedy, Rjwilmsi, Sandal bandit, Seanandjason, Sevenp, Stochata, Tgr, Thegzak, Tore.opsahl, Vyznev Xnebara, 36 anonymous edits

Centrality *Source*: http://en.wikipedia.org/w/index.php?oldid=402473434 *Contributors*: BAxelrod, Bkkbrad, Borgatts, David Eppstein, Douglas R. White, Eassin, EbrushERB, Ergotius, Fabercap, Flyingspuds, Frozen4322, Giftlite, Harthur, Hathawayc, Headbomb, Icairns, J.delanoy, Kazastankas, Kku, Loodog, Mdvs, Mgwalker, Michael Hardy, Michael Rogers, Minimac, Nburden, Piotrus, RafG, Reedy, Rjwilmsi, Sandal bandit, Seanandjason, Sevenp, Stochata, Tgr, Thegzak, Tore.opsahl, Vyznev Xnebara, 36 anonymous edits

Equivalence relation *Source*: http://en.wikipedia.org/w/index.php?oldid=405154497 *Contributors*: 345Kai, ARandomNicole, Adam majewski, AlexG, Alexb@cut-the-knot.com, Algebraist, Andy.melnikov, Antares5245, Anthonystevens2, Arthur Rubin, Arved, Aspects, AugPi, AxelBoldt, Bethre, Billinghurst, C1wang, CBM, COGDEN, CRGreathouse, Captain Wacky, Cdegremo, Charles Matthews, Cikicdragan, Cjfsyntropy, Classicalecon, Concerned cynic, Conversion script, CredoFromStart, Davehi1, Dominus, Draicone, Dyaa, Dysprosia, Egriffin, Elwikipedista, Epitome83, Feraudyh, Fibonacci, FirstPrinciples, Foxjwill, Fredrik, Fropuff, GaborLajos, General Rommel, Giftlite, Gilliam, Googl, Gregbard, Gyro Copter, Haham hanuka, Hans Adler, Henry Delforn, Herbee, Ht686rg90, Hyacinth, Ideyal, Interwal, JForget, JHunterJ, JamesBWatson, Jim.belk, Jon Awbrey, Jopxton, Kclchan, Kierano, Kurykh, LarryLACa, Lenore, Lethe, LiDaobing, Linas, MFH, Magioladitis, Mark J, Melchoir, Mets501, Michael Hardy, Michael Kinyon, Msh210, Nils Grimsmo, Obradovic Goran, Oleg Alexandrov, Palnot, Patrick, Paul August, Pekaje, PepijnvdG, Phys, PierreAbbat, Pomte, Pred, Qwfp, ReneGMata, Revolver, Rlupsa, Rnealh, Robin S, Romanm, RonnieBrown, Ryguasu, Salgueiro, Salix alba, ShaunMacPherson, Silverfish, Sjn28, SpaceFlight89, Spearhead, SpeedyGonsales, Station1, Stevertigo, Stifle, Taifunbrowser, TakuyaMurata, Tobias Bergemann, Toby Bartels, Tosha, Vanished User 0001, WilliamH, Wmli, Wshun, Xantharius, Zundark, 235 anonymous edits

Centralization *Source*: http://en.wikipedia.org/w/index.php?oldid=272176073 *Contributors*: Altenmann, Amina.imran100, Ayrton Prost, Bongwarrior, CALR, Danielkueh, DrJos, Electionworld, Fudge-o, Grillo, Hectorthebat, Hede2000, J04n, Jonasaurus, Khalid, Krakatoa, Maurreen, Mydogategodshat, Nirvana2013, Olivier, Parande, Phantomsteve, Pizza Puzzle, Pm master, Pnm, Primordial, Reinyday, Richardjyoung, Santas back1, Spongefrog, The Parting Glass, UberScienceNerd, Vicki Rosenzweig, WereSpielChequers, 25 anonymous edits

Clustering coefficient *Source*: http://en.wikipedia.org/w/index.php?oldid=401654396 *Contributors*: AlexKarpman, Cbouyio, Cloois, DS1953, Davepape, David Eppstein, DearPrudence, Dennishouston, FernandoCela, Fnielsen, Giftlite, Hwttdz, Kku, Meekohi, Michael Hardy, Netzwerkerin, Nonstandard, Oleg Alexandrov, Possie11, Psyklic, Rjwilmsi, RobinK, Samkung, Stochata, Taxipom, Tore.opsahl, TripleF, Tsaitgaist, VoytekG, Zanturaeon, 46 anonymous edits

Structural cohesion *Source*: http://en.wikipedia.org/w/index.php?oldid=394157014 *Contributors*: Arlaplace, Bobfrombrockley, David Eppstein, Dondegroovily, Douglas R. White, Fram, HappyCamper, John KB, Malcolm, Pichpich

Graph (mathematics) *Source*: http://en.wikipedia.org/w/index.php?oldid=405195681 *Contributors*: 3ICE, ABF, Abdull, Ahoerstemeier, Aitias, Ajraddatz, Aknorals, Akutagawa10, Algont, Altenmann, Amintora, Anabus, Anand jeyahar, Andros 1337, Barras, BenRG, Bethnim, Bhadani, BiT, Bobo192, Booyabazooka, Borgx, Brentdax, Burnin1134, CRGreathouse, Can't sleep, clown will eat me, Catgut, Cbdorsett, Cbuckley, Ch'marr, Chinasaur, Chmod007, Chris the speller, Chronist, Citrus538, Cjfsyntropy, Cornflake pirate, Corti, Crisófilax, Cybercobra, DARTH SIDIOUS 2, Danrah, David Eppstein, Davidfstr, Dbenbenn, Dcoetzee, Ddxc, Debamf, Debeolaurus, Den fjättrade ankan, Deor, Dicklyon, Djk3, Dockfish, Dreamster, Dtrebbien, Dureo, Dysprosia, E mraedarab, Editor70, Erhudy, Eric Lengyel, Falcon8765, Gaius Cornelius, Gandalf61, Gauge, Gene.arboit, George100, Giftlite, Gutin, Gwaihir, Hairy Dude, Hannes Eder, Harp, Headbomb, Henry Delforn, Ijdejter, Ilia Kr., Ilya, J.delanoy, JNW, Jiang, Joeblakesley, Jokes Free4Me, Jon Awbrey, Jon har, Joriki, Jpeeling, Jpiw, Jwanders, Karada, Karlscherer3, Kine, Knutux, Kruusamägi, Kuru, L Kensington, Liao, LuckyWizard, MER-C, Magister Mathematicae, Mahanga, Marc van Leeuwen, MathMartin, Matt Crypto, McKay, Mdd, Michael Hardy, Michael Slone, MikeBorkowski, Mindmatrix, Miym, Mountain, Mymyhoward16, Neilc, Netrapt, Nowhither, Nsk92, Ohnoitsjamie, Oliphaunt, Oxymoron83, Pafcu, Paleorthid, Patrick, Paul August, PaulTanenbaum, Peter Kwok, Peterl, Phegyi81, PhotoBox, Possum, Powerthirst123, Quaeler, Radagast3, RaseaC, Repied, Requestion, Rettetast, Rgclegg, Rich Farmbrough, Rjwilmsi, RobertBorgersen, Robertsteadman, RobinK, Royerloic, Salix alba, Sdrucker, Shadowjams, Shanel, Siddhant, Silversmith, Someguy1221, SophomoricPedant, Sswn, Stevertigo, Struthious Bandersnatch, Stux, Suchap, Super-Magician, Svick, THEN WHO WAS PHONE?, Tangi-tamma, Tbc2, Tempodivalse, Tgv8925, The Anome, Theone256, Tom harrison, Tomhubbard, Tomo, Tompw, Tomruen, Tosha, Tslocum, Twri, Tyw7, Ulric1313, Urdutext, Vaughan Pratt, W, Wandrer2, Wesley Moy, Wgunther, White Trillium, WikiDao, Wikidsp, Willking1979, Wshun, XJamRastafire, Xiong, Yath, Yecril, Yitzhak, Ylloh, Zachlipton, Zaslav, Zero0000, Zocky, Zven, Пика Пика, 324 anonymous edits

Bridge (graph theory) *Source*: http://en.wikipedia.org/w/index.php?oldid=396087687 *Contributors*: Algebraist, At-par, BearMachine, Bedwyr, Booyabazooka, DaedalusInfinity, David Eppstein, Dcoetzee, Flyrev, Giftlite, GregorB, Headbomb, Herr Satz, Joshuancooper, Klundarr, McKay, Mjaredd, Phys, Vanish2, XJamRastafire, 19 anonymous edits

Graph theory *Source*: http://en.wikipedia.org/w/index.php?oldid=404410018 *Contributors*: APH, Aaronzat, Abeg92, Agro1986, Aknxy, Akutagawa10, Alan Au, Alansohn, Alcidesfonseca, Alex Bakharev, Almi, Altenmann, Ams80, Andre Engels, Andrea105, Andreas Kaufmann, Andris, Ankitbhatt, Anna Lincoln, Anonymous Dissident, Anthonynow12, Aquae, Arbor, Armoreno10, Arvindn, Astrophil, AxelBoldt, Ayda D, Bact, Beda42, Berteun, Bkell, BlckKnght, Boleslav Bobcik, Booyabazooka, Brent Gulanowski, Brick Thrower, Brona, Bumbulski, C S, Camembert, CanadianLinuxUser, Caramdir, Cerber, CesarB, Chalst, Charles Matthews, Chas zzz brown, Chmod007, Conversion script, Corpx, Csl77, D75304, DFRussia, Damien Karras, Daniel Quinlan, David Eppstein, Davidfstr, Dbenbenn, Delaszk, DerHexer, Dgrant, Dicklyon, Diego UFCG, Dina, Disavian, Discospinster, Dittaeva, Doctorbozzball, Dr.S.Ramachandran, Duncharris, Dycedarg, Dysprosia, Dze27, ElBenevolente, Eleuther, Elf, Eric.weigle, Eugeneiiim, Favonian, FayssalF, Fchristo, Feeeshboy, Fivelittlemonkeys, Fred Bradstadt, Fredrik, FvdP, GGordonWorleyIII, GTBacchus, Gaius Cornelius, Gandalf61, Garyzx, Geometry guy, George Burgess, Gianfranco, Giftlite, GiveAFishABone, Glinos, Gmelli, Goochelaar, Gragragra, Graham87, GraphTheoryPwns, GregorB, Grog, Gutza, Gyan, Hannes Eder, Hbruhn, Headbomb, Henning Makholm, Hirzel, Idiosyncratic-bumblebee, Ignatzmice, Igodard, Ino5hiro, Jakob Voss, Jalpar75, Jaxl, JeLuF, Jeronimo, Jheuristic, Jinma, Joespiff, Jojit fb, Jon Awbrey, Jon har, Jorge Stolfi, Jwissick, Kalogeropoulos, Karl E. V. Palmen, KellyCoinGuy, King Bee, Knutux, Kope, Kpjas, Kruusamägi, LC, LOL, Ldecola, Liao, Linas, Looxix, Low-frequency internal, Lpgeffen, Lupin, Madewokherd, Magister Mathematicae, Magmi, Mahlerite, Mailer diablo, Mani1, Marek69, Marianocecowski, Mark Foskey, Mark Renier, MathMartin, Matusz, Maurobio, Maxime.Debosschere, McKay, Meekohi, Melchoir, Michael Hardy, Michael Slone, Miguel, Mild Bill Hiccup, Miym, Mlpkr, Morphh, Msh210, Mxn, Myanw, MynameisJayden, Naerbnic, Nanshu, Nichtich, Ntsimp, Obankston, Obradovic Goran, Ohnoitsjamie, Oleg Alexandrov, Oli Filth, Onorem, Optim, OrangeDog, Oskar Flordal, Pashute, Patrick, Paul August, Paul Murray, PaulHoadley, PaulTanenbaum, Pcb21, Peter Kwok, Piano non troppo, PierreAbbat, Poor Yorick, Populus, Powerthirst123, Protonk, Pstanton, Puckly, Pugget, Quaeler, Qutezuce, RUVARD, Radagast3, Ratiocinate, Renice, Requestion, Restname, Rev3nant, RexNL, Rmiesen, Roachmeister, Robert Merkel, Robin klein, RobinK, Ronz, Ruud Koot, SCEhardt, Sacredmint, Shanes, Shd, Shepazu, Shikhar1986, Shizhao, Sibian, SlumdogAramis, Smoke73, Solitude, Stochata, Sundar, Sverdrup, TakuyaMurata, Tangi-tamma, Tarotcards, Taw, Taxipom, Tckma, Template namespace initialisation script, The Cave Troll, The Isiah, Thesilverbail, Thv, Titanic4000, Tomo, Tompw, Trinitrix, Tyir, Tyler McHenry, Uncle Dick, Vonkje, Watersmeetfreak, Whiteknox, Whyfish, Womiller99, Wshun, Wsu-dm-jb, Wsu-f, XJamRastafire, Xdenizen, Xiong, XxjwuxX, Yecril, Ylloh, Youngster68, Zaslav, Zero0000, Zoicon5, Zundark, Александър, 386 anonymous edits

Network theory *Source*: http://en.wikipedia.org/w/index.php?oldid=403541869 *Contributors*: Ahoerstemeier, Argon233, Bellagio99, Beno1000, Benschop, Biblbroks, CRGreathouse, Camw, Charles Matthews, Commander Keane, DarwinPeacock, Delaszk, Denoir, Dicklyon, Doncqueurs, Door2, Douglas R. White, Dreftymac, Ems2715, Floorsheim, Floridi, Giftlite, Goodoldpolonius, Gregbard, IanManka, Imersion, Jason Quinn, Jimmaths, Jp3z, Jsarmi, Jwdietrich2, Kingturtle, Knowledge Seeker, Lankiveil, Magmi, Mange01, Mants, Mark Elliott, Maxal, Mentatseb, Mitar, Miym, Mootros, Oatmealcookiemon, Orgnet, Otus, PAR, Pasquale, PhilKnight, Possum, Red Johnny, Ronz, Sdrtirs, Ses, Shanes, Silvertje, SiobhanHansa, Stochata, Supernet, Thorwald, Tiddly Tom, Tokigun, WikHead, Wile E. Heresiarch, Woohookitty, Zzuuzz, 50 anonymous edits

Closeness (graph theory) *Source*: http://en.wikipedia.org/w/index.php?oldid=295052340 *Contributors*: BAxelrod, Bkkbrad, Borgatts, David Eppstein, Douglas R. White, Eassin, EbrushERB, Ergotius, Fabercap, Flyingspuds, Frozen4322, Giftlite, Harthur, Hathawayc, Headbomb, Icairns, J.delanoy, Kazastankas, Kku, Loodog, Mdvs, Mgwalker, Michael Hardy, Michael Rogers, Minimac, Nburden, Piotrus, RafG, Reedy, Rjwilmsi, Sandal bandit, Seanandjason, Sevenp, Stochata, Tgr, Thegzak, Tore.opsahl, Vyznev Xnebara, 36 anonymous edits

Dense graph *Source*: http://en.wikipedia.org/w/index.php?oldid=380038726 *Contributors*: Chortos-2, DanDanRevolution, David Eppstein, Goochelaar, Jitse Niesen, Kku, Koavf, Larry V, Miym, Mwtoews, Twri, 11 anonymous edits

Directed graph *Source*: http://en.wikipedia.org/w/index.php?oldid=400750076 *Contributors*: Altenmann, Andreas Kaufmann, Anne Bauval, BiT, Booyabazooka, Bryan.burgers, Calle, Catgut, Corruptcopper, Dzied Bulbash, Giftlite, Grue, Hamaryns, Headbomb, Henry Delforn, Justin W Smith, M-le-mot-dit, Marcuse7, Mark Renier, Miym, NuclearWarfare, PaulTanenbaum, Pcap, Ricardo Ferreira de Oliveira, Rinix, Twri, WookieInHeat, Zaslav, آرمان, 22 anonymous edits

Vertex (graph theory) *Source*: http://en.wikipedia.org/w/index.php?oldid=395956133 *Contributors*: Altenmann, Anoko moonlight, BiT, Cburnett, Ciphers, DARTH SIDIOUS 2, David Eppstein, Dbenbenn, Giftlite, JaGa, Kl4m, MathMartin, Miym, Niemeyerstein en, Orphan Wiki, Phil1881, Purestorm, Rich Farmbrough, Twri, Univer, XJamRastafire, Ylloh, 8 anonymous edits

Flow network *Source*: http://en.wikipedia.org/w/index.php?oldid=401080485 *Contributors*: Andreas Kaufmann, BenFrantzDale, Bfishburne, Bubba73, Cazort, Charles Matthews, Cobi, Craigyjack, Crowsnest, Darkwind, David Eppstein, Dcoetzee, Debamf, Dysprosia, EconoPhysicist, Expiring frog, Freederick, Giftlite, Grumpyland, Heineman, Hermel, Jalpar75, Jirka6, Jittat, Jjurski, Jogloran, Kiefer.Wolfowitz, LOL, Leafyplant, Ligulem, Mange01, Maurice Carbonaro, Mdd, Misza13, Mormegil, Myleslong, Neilc, Nejko, Nethgirb, Nils Grimsmo, Oddbod7, Oleg Alexandrov, Paul August, Rich Farmbrough, Sembrestels, Serknap, SeventyThree, Silverfish, Spoon!, Ssnseawolf, Svick, That Guy, From That Show!, User A1, Vikas.menon, Vonkje, Womiller99, Ycl6, Zaslav, ماني, 79 anonymous edits

Cycle (graph theory) *Source*: http://en.wikipedia.org/w/index.php?oldid=391168221 *Contributors*: Aegis Maelstrom, Altenmann, AxelBoldt, David Eppstein, Drbreznjev, Funandtrvl, Giftlite, LOL, Linas, MathMartin, Michael Hardy, Oleg Alexandrov, Robert Samal, Tomo, Twri, Uogl, Uselesswarrior, 12 anonymous edits

Adjacency matrix *Source*: http://en.wikipedia.org/w/index.php?oldid=396068155 *Contributors*: Abdull, Abha Jain, Aleph4, Arthouse, AxelBoldt, Beetstra, BenFrantzDale, Bitsianshash, Booyabazooka, Burn, Calle, Cburnett, David Eppstein, Dcoetzee, Debresser, Dreadstar, Dysprosia, ElBenevolente, Fredrik, Garyzx, Gauge, Giftlite, JackSchmidt, JamesBWatson, Jean Honorio, Jogers, John of Reading, Jpbowen, Juffi, Kneufeld, Lipedia, MarkSweep, MathMartin, Mbogelund, Mdrine, Michael Hardy, Miym, Morre, Natelewis, Oleg Alexandrov, Olenielsen, Only2sea, Periergeia, Phils, Reina riemann, Rhetth, Rich Farmbrough, RobinK, SPTWriter, Salgueiro, Schneelocke, Sławomir Biały, TakuyaMurata, Tbackstr, Tim Q. Wells, Timendum, Tomo, Ttzz, Twri, Yoav HaCohen, YuryKirienko, Zaslav, سعی, 63 anonymous edits

Tree (graph theory) *Source*: http://en.wikipedia.org/w/index.php?oldid=404406578 *Contributors*: 3D-Grabber, Adking80, Algebraist, Almit39, Altenmann, Andreas Kaufmann, Andrevan, Angr, Anna Lincoln, Anonymous Dissident, Arunkumar, AxelBoldt, Battamer, Booyabazooka, Breno, Bryan Derksen, Charles Matthews, Dagh, David Eppstein, Dbenbenn, Debivort, Dominus, Dysprosia, Edelac, Edemaine, Ezrakilty, Gadykozma, Giftlite, Gurch, Hermel, Hippophaë, Hyad, Inhumandecency, Ioakar, Jaredwf, Jitse Niesen, Jon Awbrey, Josh Cherry, Karl-Henner, Knutux, Kuszi, Kzollman, Lambiam, Linas, Lssilva, MSGJ, Marsupilcoalt, MathMartin, McKay, Michael Hardy, Michael Slone, Miguel, Miym, N4nojohn, Naerbnic, Neilc, Nguyen Thanh Quang, Obradovic Goran, OdedSchramm, Patrick, Paul August, PaulTanenbaum, Philippe Giabbanelli, Philly jawn, PhotoBox, Physis, Pjvpjv, Poor Yorick, RaitisMath, Rp, Salgueiro, Salix alba, Sms97, Sori, StaticVision, Timwi, Tizio, Topology Expert, Tosha, Trovatore, Twri, Uv1234, Vanished User 0001, Wrstark, XJamRastafire, Xcelerate, Ylloh, Zarvok, Zero0000, سعی, 69 anonymous edits

Path (graph theory) *Source*: http://en.wikipedia.org/w/index.php?oldid=387998998 *Contributors*: Adking80, Altenmann, Ashesh0326, Booyabazooka, CBM, Charles Matthews, Chmod007, DRE, David Eppstein, David-Sarah Hopwood, Dcoetzee, Derbeth, Djr36, Doradus, Dysprosia, ElommolE, Eomund, Flamholz, Gdr, Giftlite, Ideal gas equation, Jamelan, Jittat, Joriki, Justin W Smith, Linas, MSGJ, MathMartin, Miym, Modify, Neilc, Olego, Paul August, Rich Farmbrough, SWAdair, Serkan Kenar, Stannered, Svick, Twri, VictorAnyakin, Yaninass2, Zero0000, 9 anonymous edits

Glossary of graph theory *Source*: http://en.wikipedia.org/w/index.php?oldid=404592695 *Contributors*: 3mta3, A1kmm, Aarond144, Achab, Algebraist, Altenmann, Altoid, Archelon, ArnoldReinhold, Bender235, Bethnim, Bkell, Booyabazooka, Brick Thrower, Brona, Buenasdiaz, Charles Matthews, Citrus538, Closedmouth, D6, DVanDyck, Damian Yerrick, David Eppstein, Dbenbenn, Dcljr, Dcoetzee, Doc honcho, Doradus, Dysprosia, Edward, Eggwadi, El C, Elwikipedista, Eric119, Ferris37, GGordonWorleyIII, GTBacchus, Gaius Cornelius, Giftlite, Grubber, Happynomad, Headbomb, HorsePunchKid, Hyacinth, Ivan Štambuk, JLeander, Jalal0, Jfitzell, Jokes Free4Me, Joriki, Justin W Smith, Jwanders, Jérôme, Kjoonlee, Lansey, Lasunncty, Linas, Markhurd, MathMartin, Maximus Rex, Meand, MentorMentorum, Mgreenbe, Michael Hardy, Michael@nosivad.com, Miym, Mzamora2, N8wilson, Nonenmac, Oleg Alexandrov, Ott2, Patrick, Paul August, PaulTanenbaum, Peter Kwok, Pojo, Populus, Quaeler, Quintopia, Quuxplusone, R'n'B, Ratfox, Rdvdijk, Reina riemann, Rich Farmbrough, Rick Norwood, Ricky81682, Rsdetsch, Ruud Koot, Salgueiro, Salix alba, SanitySolipsism, Scarpy, Shadowjams, Skaraoke, Spanningtree, Starcrab, Stux, Sunayana, Sundar, TakuyaMurata, The Transliterator, Thechao, Thehotelambush, Tizio, Tomo, Twri, Vonkje, Whorush, Wshun, XJamRastafire, Xiong, Yitzhak, Zaslav, 111 anonymous edits

Image Sources, Licenses and Contributors

License