

OPTIMAL CONNECTIONS BETWEEN HIGH-TECH COMPANIES: AN ALGORITHM AND ITS APPLICATIONS*

Song Yang

Assistant Professor
Department of Sociology
211 Old Main
University of Arkansas
Fayetteville, AR, 72701

Henry Hexmoor

Assistant Professor
Department of Computer Science & Engineering
Engineering Hall #328
University of Arkansas
Fayetteville AR 72701

*Direct all correspondences to Song Yang at yangw@cavern.uark.edu We thank professor David Knoke for providing helpful comments and his network dataset to this paper. A summer funds of Graduate Research Partnership Program from the College of Liberal Arts at University of Minnesota supported this project.

OPTIMAL CONNECTIONS BETWEEN HIGH-TECH COMPANIES: AN ALGORITHM AND ITS APPLICATIONS *

Abstract

Identifying several problems in measuring the strongest path connecting pairs of actors in valued graphs, researchers proposed that average path value (APV) be used to indicate optimal connections between dyads. However, a lack of proper computer algorithm and programming implementation has hindered a wide-range application of the proposed APV solution. We develop a computer algorithm and fully implement it with four JAVA programs, which are available on request. The programs produce an optimal connection matrix, which is subsequently input into UCINET for further multidimensional scaling and clustering analysis. We illustrate this procedure with a data matrix containing 38 organizations in information technology. We discuss our methodological contributions to future social network studies.

Keywords: Graph theory, Average path value, Computer algorithm, Valued graph

OPTIMAL CONNECTIONS BETWEEN HIGH-TECH COMPANIES: AN ALGORITHM AND ITS APPLICATIONS *

Introduction

Network researchers have developed methods for measuring strength and distance between pairs of nodes in binary graphs, where dichotomous values of 1 and 0 indicate presence or absence of ties between pairs of nodes (Wasserman and Faust 1994). In contrast, research on measuring distance between dyads in valued graphs has made little progress. Valued graphs contain different weights attached to the ties connecting nodes, which results in difficulty in measuring distance among different nodes, especially those connected via indirect ties. Researchers have proposed using average path value (APV) to indicate the optimal connections between a pair of nodes in valued graphs (Yang and Knoke 2001). The APV solution encompasses two steps. First, APV is produced by dividing the weakest path value in a chain of connection (a path) by the distance between a pair (distance is number of lines in a path). Second, because each path has its APV and a pair of nodes may be connected via multiple paths, including a direct path and some indirect paths, the largest value among all the APVs is used to indicate the optimal connection between the pair.

However, attempts to implement this mathematic solution using a programming language encounter problems due to lack of an appropriate computer algorithm. Researchers wrote an artificial program to trace indirect ties between actors separated by no more than 3 intermediate nodes within matrices containing up to five actors (Yang and Knoke 2001: 294). In reality, network analysts often produce matrices with tens or hundreds of actors, which normally require traces of indirect steps for up to $N-1$ links for a pair connected via N nodes. Continuing in this

line of work, we propose an implemented algorithm with JAVA programming language. Our programs input a matrix indicating direct path values among multiple nodes in a valued graph, and output a matrix containing average path values for pairs of nodes. The resulting APV matrix is then input into UCINET, a well-known social network analysis package, for further clustering analysis and multidimensional scaling (for detailed description of these methods and programming procedure, see Kruskal and Wish 1978; Aldenderfer and Blashfield 1984, and Borgatti, Everett, and Freeman 1999). We illustrate this procedure with our programs and network data containing strategic alliances among 38 Information Technology (IT) companies from the Global Information Sector project (Genereux and Knoke 1999).

The APV Solution

In this section we briefly review the APV solution to summarize scholarly efforts to capture optimal connections between pairs of actors in valued graphs. Addressing the issue that network measures of the strength of relations between pairs of nodes in valued graphs commonly ignore the binary distance concept between the pairs (Peay 1980; Flament 1963), Yang and Knoke (2001) proposed to use average path value (APV) to take into account the costs required for indirectly connected dyads to reach one another through a varying number of intermediaries. The APV between a pair of nodes is obtained by dividing the smallest value of any line in the path connecting the pair by the binary distance between the pair. Consider organizations A and C are indirectly connected via an intermediate organization B, in which organization A is tied to organization B with 5 partnerships, and organization C to organization B by 3 partnerships. The APV solution uses the smallest value in the path (3) divided by the binary distance between A and C (2) to produce 1.5 ($3/2 = 1.5$). Consider a variation where organizations A and C are not

only indirectly connected via organization B, but also directly connected with, say, 1 partnership. The APV solution then requires comparison of the indirect APV of 1.5 with the direct tie value of 1 and picks up the larger value of the two ($\text{Max}(1.5, 1) = 1.5$) to indicate the optimal connection between organization A and C.

Yang and Knoke (2001) demonstrated that UCINET could not compute the proposed measures of optimal connections in a valued graph. Instead, they developed a customized program to compute the APV measures for indirectly connected pairs for up to 3 steps for a network matrix for up to five nodes. However, network researchers often deal with a matrix containing hundreds of nodes, and require comparison of $N-1$ intermediate links to determine the smallest value in the path connecting any two nodes. The difficulty in fully implementing the proposed APV solution lies in the lack of a suitable algorithm. A common sense solution requires researchers to trace the indirect steps literally from the first intermediate node to the last intermediate node in a path connecting two nodes. Aside from incurring high cost of N^N run time for a valued matrix with N nodes, this common sense solution is extremely difficult to program as the network size N and the number of indirect steps for each indirectly connected nodes are not available until after the program completes. In other words, because programmers cannot pre-determine the size of the matrix and the indirect steps to connect each pair in the graph, they can only instruct the program to trace a pre-fixed number of indirect steps regardless of the size and the number of the indirect steps in the input matrix. In this paper, we developed an algorithm capable of tracing $N-1$ indirect paths in the APV computation while assuming no prior knowledge of network size and the number of intermediate nodes. The following section discusses our algorithm in detail and illustrates its application with an artificial network matrix.

The Algorithm

In this section, we first introduce a concept called connected component (Wasserman and Faust 1994: 109-110), which is defined in our analyses as a set of nodes, in which each node can reach every other node in the set. A network dataset can contain several such connected components. Our programs first process the matrix to identify different components in the matrix with a standard algorithm called Union Find that discovers Connected Components (for further elaboration on Connected Component and Union Find Algorithms, please refer to Cormen, Leiserson, and Rivest, 2002). Once components are identified, we then process each individual component using our algorithm called Maximum APVC (MAPVC). Once all components are processed, the results are formatted in the input matrix format for further multidimensional scaling and clustering analysis. Outline of our algorithm is given below:

Algorithm (G)

1. $C \leftarrow \text{Union-Find}(G)$
2. **for** each connected component $c_i \in C$ **do** $o_i = \text{MAPVC}(c)$
3. **for** each o_i **do** $O \leftarrow O \cup \{o_i\}$

We now describe operations of MAPVC on a single connected component. MAPVC is a polynomial time algorithm that finds the path with the highest APV value among all combinations of nodes in a valued graph. A polynomial time algorithm provides computational tractability, which ensures that the time to process information is a finite power of its input size. In particular, running time of our algorithm is N^4 , where N indicates the number of nodes in the input file; we will prove its run time of N^4 in later discussion.

For a given connected component c_i , MAPVC considers each node v_j one at a time and incrementally constructs a path from that node to all other nodes. MAPVC calls a subroutine Maximum APV (MAPV) to process each node. Details of MAPV process are described next.

When considering a node v_i , another node v_j is selected such that it has the maximum APV (path values/number of lines) thus far. That node v_j is then marked as visited for paths starting at v_i . This means that for node v_i , v_j is considered as the node with ideal path value and it will not be visited again. Therefore, the partial path from v_i to v_j is considered for extension in subsequent processing. Every node k that extends the path from v_i to v_j and yields a lower edge weight than the existing weight for the path will be used to calculate the APV between nodes i and k . An edge is a direct link between a pair of nodes and an edge weight is the value attached to the edge. For every extension, the algorithm compares all APVs and extends the path with the largest APV and NEVER extended before. The process continues until every path in the connected component matrix was either extended or was a terminal path, which was because either no other nodes is reachable or circular path occurs (the path connects back to the beginning node i). At the end of this process, the algorithm compares different APVs during each stage of path extension and picks up the largest APV to indicate the optimal connection between the node i and node k in a single connected component matrix. Once the processing of a given node completes, MAPV picks up the next node from c_i for processing and continues this process.

To illustrate this process for a single node, we produced three tables and two figures. Figure 1 shows an artificial valued graph with 6 nodes. Table 1 shows the adjacency list representation of that valued graph in a matrix format. The infinite sign “ ∞ ” in Table 3 denotes absence of a direct path between two nodes. For example, Nodes A and C are not connected

directly, which is indicated with an infinite sign in their cell entry. Table 2 illustrates the processing procedure of MAPV algorithm on the node A in the graph. Figure 2 displays the pseudocode of the MAPV algorithm. Table 3 shows the output APV matrix of the valued graph after processed by the JAVA programs, which fully implement our MAPVC algorithm. Note that the two furthest nodes A and F in the graph are not directly connected, as is shown in Figure 1 and Table 1. Table 2 shows that our algorithm constructs an optimal connection path (AEDF) between A and F. Our programs subsequently calculate the APV for nodes A and F to be $1/3$, which is shown in Table 3. Because understanding Table 2 is very essential to understanding our algorithm, we devote the following text to describing Table 2 in great detail.

Suppose we start with the node A. First step out, node A sees node B and E. Because the APV for AB ($2/1=2$) is larger than the APV of AE ($1/1=1$), the algorithm picks AB to extend and record the APVs for both AB and AE and mark the path AB as visited. Extending AB, the only other reachable node is C. So the second step has three paths: AB, ABC, and AE. Because AB was marked as visited, ABC and AE became the candidate path to be extended in later stage. Because they have the same APV (ABC $2/2=1$; AE $1/1=1$), the algorithm randomly picks up AE to extend (Extending ABC produces the same end result as extending ABC, we leave this as an exercise for the readers). At step 3, AE is extended to the only reachable node along this path D, thus AED is produced with an APV of $1/2$. Because paths AB and AE were marked as visited and ABC ($2/2=1$) has a larger APV than AED ($1/2$), ABC is picked for an extension. At step 4, ABC extends to ABCD as node D is the only reachable node from ABC. All the paths but ABCD was marked as visited. Consequently ABCD was chosen for an extension. At step 5, ABCD was extended to nodes E and F. Because AB, ABC, ABCD are all marked and ABCDE and ABCDF are terminal paths, step 5 has only AED to be extended, which was skipped over in

step 3. Step 6 extends AED to two reachable nodes C and F, producing two new paths: AEDC (1/2) and AEDF (1/3). Because AB, AED, and AE are extended and AEDF is a terminal path, steps 6 can only extend AEDC. Step 7 extends AEDC to the only reachable node along the path B, producing a new path AEDCB (1/4). At this stage, all the paths are either marked or terminal, thus leaving no path to extend. Step 8 compares all paths and chooses the one with the highest APV along every column to indicate optimal connections between node A and other nodes in this single connected component. The resulting APVs between node A and other nodes that are displayed at the last row of Table 2 match up with the first row/column of Table 3. For example, A and C has an optimal connection path ABC with an APV of 1, which is shown in both Table 2 and Table 3.

The famous shortest path Dijkstra algorithm is not sufficient for our purposes because it does not take into account the number of edges in a path (path length). For example, the Dijkstra algorithm makes no distinction between nodes A and C in Figure 1 because A and C are connected either via AEDC (1+1+2=4) or ABC (2+2=4), both paths add up to 4 in their edge weight. However, our MAPVC algorithm decides that ABC is preferable than AEDC because the APV for ABC is (2/2=1) in contrast to the APV for AEDC of 1/3=0.33. The next two sections formalize our argument by providing a theorem and the proof of the correctness and run time for the algorithm.

Theorem: MAPV is correct. I.e., If we run MAPV on a valued bi-directional graph $G = (V, E)$, which is a single connected component with positive weight function starting at node \mathbf{s} , then at termination $APV(\mathbf{s}, \mathbf{u})$ for all nodes $\mathbf{u} \in V$.

Proof: We show that for each node $\mathbf{d} \in V$, we have $APV(\mathbf{s}, \mathbf{d})$. Let's consider the non-circular optimal path from \mathbf{s} to \mathbf{d} discovered by MAPV to be \mathbf{p} , where the path starts at \mathbf{s} and

ends with \mathbf{d} . This path is the optimal path with the maximum APV value among all other paths \mathbf{p}' that start at \mathbf{s} and end with \mathbf{d} . For the purposes of contradiction, let's assume \mathbf{p} is not optimal and \mathbf{p}' is a superior path. This means that \mathbf{p}' has a higher APV value than \mathbf{p} . If this was the case, either \mathbf{p}' would have been selected to be extended or the corresponding path marked as visited was not really the node to be marked as visited. This contradicts our scheme for node selection and path extension. Therefore \mathbf{p}' is not really a superior path and \mathbf{p} must in fact be optimal. This contradiction proves our algorithm correct. Θ

Because the paths that are extended are at most as long as the number of edges, the running time for tracing each node is $O(|V| * |E|)$ or roughly $O(n^2)$, whereby V indicates number of nodes and E denotes the number of edges. Let's consider a graph $G = (V, E)$ and $n = |V|$, because $\text{MAPV}(v_i)$ works on all possible paths starting from v_i , the running time of MAPV is proportional to n^2 and at each step the algorithm makes $n-1$ comparisons. Therefore, running time of MAPV is $O(n^3)$. Since we run MAPV on a connected component, and MAPVC calls MAPV as many times as the number of nodes, running time of MAPVC climbs up to $O(n^4)$.

Union-Find algorithm used in MAPVC has a running time of $O(n * \log(n))$. This does not increase MAPVC's running time since it is a sequential step. Also post-MAPVC process comparing different APVs for a pair connected via multiple paths costs a constant unit of time, which is negligible. Therefore, MAPVC has a running time of $O(n^4)$. As our example shows, the average case running time is far better than the worst case.

The Data

Our dataset is a cross-sectional network matrix data containing strategic alliances forged in 1998 among 38 companies in the Informational Technology, whereby strategic alliances are

defined as those inter-firm agreements that can reasonably be assumed to effect the long-term product market positioning of at least one partner (Hagedoorn and Schakenraad 1992: 164). This dataset comes from a large longitudinal database containing strategic alliances among 145 organizations in Information Technology from 1989 to 1998 (For detailed data collection procedure, see Genereux and Knoke 1999). The 38 organizations comprise a core set of actors in 1998 organizational strategic alliance, as they are the top 38 organizations in the number of strategic alliances among the total 135 organizations in 1998 (the 10 organizations less than the initial 145 organizations in 1989 is due to the organizational merger or demise). Table 4 shows the matrix data, in which each cell value indicates the number of different strategic alliances between that pair of organizations. For example, Compaq and Microsoft forged 15 different strategic alliances in 1998, indicated by 15 in the cell entry of Compaq and Microsoft. The matrix is symmetric, suggesting bi-directionality in the connections. For example, looking down the rows, the “Microsoft” row has a cell entry of 10 with the “HP” column. Likewise, the “HP” row also has a cell entry of 10 with the “Microsoft” column. Both indicate that the two companies forged 10 different strategic alliances during 1998.

Results

Our programs read the raw data of strategic alliance matrix and produce the APV optimal connection matrix among the 38 organizations. We conducted hierarchical clustering analysis and multidimensional scaling on the APV matrix using UCINET. Figure 3 shows five circles in different colors that indicate five different clusters, which result from UCINET’s clustering analysis. The hierarchical clustering routine classifies organizations in different groups based on their social distance to each other. Each cluster contains organizations with great social distance

in terms of their patterns of direct and indirect partners than to the organizations occupying the other clusters. We use UCINET's nonmetric multidimensional scaling routine to produce the two-dimensional map of this alliance network structure, with acceptable fit to the matrix (stress = 0.124).

In general, organizations with numerous alliances with different partners are located in the center of the two-dimensional map, whereas firms with fewer connections appear in the periphery. Figure 3 not only vividly describes the centrality of many prominent organizations in the current IT industry, but also shows a cluster-based competition, in which organizational clusters, composing of several densely connected organizations, compete against each other for market acquisition, new technology research and development, or product standardization (Gomes-Casseres, 1994). Located on the central positions, which are indicated by the 0, 0 coordinate in the two-dimensional scaling, are several major players such as Microsoft, Intel, Cisco, Dell, IBM, 3com, Lucent, Compaq, HP, and AT&T. Those organizations forged a central cluster by dense connections among them. Core to this cluster is the decade long alliance between Microsoft and Intel that forge "Wintel" duo alliances to gain dominant positions in the personal computer, business network, internet, and cable transmission markets (Knoke and Yang 2000). Starting from mid-1990s, several other Silicon Valley companies led by Sun Microsystems, Netscape, and Oracle forged dense alliance to counteract Wintel duo coalition. They successfully convinced the U.S. Department of Justice to charge Microsoft with anti-trust suit for bundling its Internet Explorer with its Windows operating systems. The incident ends as Microsoft survived the federal lawsuit but was forced to make some concessions designed to weaken its domination, while Netscape was taken over by AOL.

The cluster in the upper left corner of the map consists of three Japanese multi-national conglomerates, Toshiba, Hitachi, and Sony, one European conglomerate, Philips, and two U.S. telecommunication companies, Unisys and Bell South (BS). This cluster integrates companies in different interconnected industries such as electronic devices, telecommunications, and computer technology. The cluster right above the Wintel duo core coalition contains entertainment companies such as Disney and Time Warner (TW), and electronic computer companies such as NEC, Alcatel, and Tele-Communications Inc (TCI). As the core products for Alcatel, NEC, and TCI are mass media equipments such as various electronic devices, and fast-speed fiber optics, the two entertainment companies (Disney and TW) are their major customers. On the one hand, dense connections permit frequent information exchanges that ensure steady supply of customized products for Disney and TW. On the other hand, the three suppliers (Alcatel, NEC, and TCI) also secured a stable relation with their major customers through multiple alliances. The most sparse and peripheral cluster locates at the right hand side of the map, consisting of 6 organizations in telecommunication, computer mainframes, network equipments, and mobile semiconductor products. Asides from France Telecom (FT), a telecommunication giant, and SIEMENS, a German based multinational conglomerate, the other four organizations (Ericsson, Bull, Motorola, and EMC) are competitors in semiconductors, network equipments, mainframes, and mobile products. Researchers have long argued that firms competing in the same market forge horizontal alliances to avoid over-competition (Pfeffer 1987), this finding vividly illustrates how companies in headon competitions use horizontal alliances to control and coordinate their competition.

Conclusion

Although research needs of network analysts vary considerably from one to another, describing the social distance is always critical for understanding the network structural characteristics. With an appropriate computer algorithm, we overcame difficulties in implementing the proposed APV solution in computing optimal connections for pairs of nodes in valued graphs. We implement the algorithm with four JAVA programs that produce an APV matrix for 38 IT companies in 1998 organizational network structure. Upon request, the programs are available to interesting readers. The APV matrix was subsequently input in UCINET for clustering analysis and multidimensional scaling. We identified five different organizational clusters and graph a multidimensional map to visually illustrate social distance among those organizations. We offer some plausible explanations to some findings in this network configuration, which subject to further empirical scrutiny with pertinent data. For example, by examining the purposes of the strategic alliance, researchers can reveal the detail processes companies use to accomplish different goals such as market acquisition, competition management, product development, or industrial standardization. Several intriguing research questions lie ahead to be addressed with suitable data. A longitudinal research can reveal how the network structures among the same set of companies evolve over time. Furthermore, explanatory focused studies can investigate the causal factors for the changing patterns. For example, researchers can study whether organizational nationalities affect an organization's choice of strategic partners, and how the impact of nationalities evolves over time. We develop an applicable software program and provide an illustrative graph to describe the network structure for 38 IT companies, which serves as the first step toward uncovering the casual factors for such a network configuration.

References:

- Aldenderfer, Mark and Roger Blashfield. 1984. *Cluster Analysis*. Beverly Hills, California: Sage Publications
- Borgatti, Stephen P., Martin Everett, and Linton C. Freeman. 1999. *UCINET 5 for Windows: Software for Social Network Analysis* Natick: Analytic Technologies.
- Cormen, Thomas, Charles Leiserson and Ronald Rivest. 2002. *Introduction to Algorithms*. Cambridge, Mass.: MIT Press.
- Flament, Claude. 1963. *Applications of Graph Theory to Group Structure*. Englewood Cliffs, N. J.: Prentice-Hall.
- Genereux, Anne and David Knoke. 1999. "Identifying Strategic Alliances in the Global Information Sector." Paper presented at European Group for Organizational Studies (EGOS) Colloquium July 4-6 in England.
- Gomes-Casseres, Benjamin. 1994. "Group Versus Group: How Alliance Networks Compete." *Harvard Business Review* 72: 62-74.
- Hagedoorn, John and Jos Schakenraad. 1992. "Leading Companies and Networks of Strategic Alliances in Information Technologies." *Research Policy* 21:163-190.
- Knoke, David and Song Yang. 2000. "Implications of Network Structures for Managing Strategic Alliances in the Global Information Sector." Paper presented to EGOS Colloquium, Helsinki, Finland.
- Kruskal, Joseph and Myron Wish. 1978. *Multidimensional Scaling*. Beverly Hills, California: Sage Publications
- Peay, Edmund. 1980. "Connectedness in a General Model for Valued Networks" *Social Networks* 2: 385-410.
- Pfeffer, Jeffrey. 1987. "A Resource Dependence Perspective on Intercorporate Relations." Pp. 25-55 in *Intercorporate Relations: The Structural Analysis of Business*, edited by Mark Mizruchi and Michael Schwartz. New York: Cambridge University Press.
- Wasserman, Stanley and Katherine Faust. 1994. *Social Network Analysis: Methods and Applications*. New York: Cambridge University Press.
- Yang, Song and David Knoke. 2001. "Optimal Connections: Strength and Distance in Valued Graphs." *Social Networks* 23:285-295.

Table 1: Input Matrix Data of the Valued Graph

Nodes	A	B	C	D	E	F
A	0	2	∞	∞	1	∞
B	2	0	2	∞	∞	∞
C	∞	2	0	2	∞	∞
D	∞	∞	2	0	1	1
E	1	∞	∞	1	0	∞
F	∞	∞	∞	1	∞	0

Table 2: Processing Illustration for Node A

Steps (below)	Nodes (right)	A	B	C	D	E	F
1	APV Path AB for step 2		2/1 AB Mark			1/1 AE	
2	APV Path AE for Step 3		2/1 AB Marked	2/2 ABC		1/1 AE Mark	
3	APV Path ABC for Step 4		2/1 AB Marked	2/2 ABC Mark	1/2 AED	1/1 AE Marked	
4	APV Path ABCD for Step 5		2/1 AB Marked	2/2 ABC Marked	2/3 ABCD Mark	1/1 AE Marked	
5	APV Path AED for Step 6		2/1 AB Marked	2/2 ABC Marked	2/3 ABCD Marked	1/4 ABCDE Terminal	1/4 ABCDF Terminal
6	APV Path AEDC for Step 7		2/1 AB Marked	1/3 AEDC Mark	1/2 AED Marked	1/1 AE Marked	1/3 AEDF Terminal
7	APV Path		1/4 AEDCB Terminal	1/3 AEDC Marked	1/2 AED Marked	1/1 AE Marked	1/3 AEDF Terminal
	Optimal Path		AB	ABC	ABCD	AE	AEDF
	Maximum APV		2/1	2/2	2/3	1/1	1/3

Table 3: Output APV Matrix of the Valued Graph After the Processing

Nodes	A	B	C	D	E	F
A	0	2	1	$2/3$	1	$1/3$
B	2	0	2	1	$1/2$	$1/3$
C	1	2	0	2	$1/2$	$1/2$
D	$2/3$	1	2	0	1	1
E	1	$1/2$	$1/2$	1	0	$1/2$
F	$1/3$	$1/3$	$1/2$	1	$1/2$	0

Table 4: Strategic Alliance Between 38 IT Organizations

	3COM	ALCATEL	AOL	ATT	BA	BAAN	BCE	BS	BULL	CAI	CISCO	COMPAQ	DELL	DISNEY	EMC	ERICSSON	FT	FUJITSU	HITACHI	HP	IBM	INTEL	LUCENT	MICROSOFT	MOTOROLA	NEC	NETSCAPE	ORACLE	PHILIPS	SBC	SIEMENS	SONY	SUN	TCI	TOSHIBA	TW	UNISYS	USW				
3COM	0	1	0	1	1	0	2	1	1	1	2	5	4	0	0	2	0	2	2	5	5	2	4	2	1	0	0	1	0	1	2	0	0	0	0	4	0	0	1			
ALCATEL	1	0	0	0	1	0	1	1	0	0	2	1	0	0	0	1	1	2	0	0	0	1	2	2	0	2	0	0	0	0	1	1	0	0	0	1	0	0	1	0	0	1
AOL	0	0	0	3	1	0	0	0	0	0	1	3	1	1	0	0	0	0	0	1	3	0	0	2	0	1	4	2	0	0	0	0	1	2	0	0	1	0	0	1	0	0
ATT	1	0	3	0	1	0	2	1	0	0	3	3	4	2	0	0	0	0	0	5	4	0	2	5	0	0	2	2	1	1	0	1	1	1	1	0	1	0	1	0	1	
BA	1	1	1	1	0	0	2	2	0	0	2	3	1	1	0	1	0	1	0	1	1	2	2	4	1	0	1	1	1	5	1	0	1	0	1	0	0	1	0	4		
BAAN	0	0	0	0	0	0	0	1	0	0	2	1	0	1	0	0	0	0	1	3	1	0	3	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	
BCE	2	1	0	2	2	0	0	1	0	0	3	3	0	0	0	1	1	1	0	1	0	1	3	6	1	0	0	0	0	2	1	1	0	0	0	0	0	0	0	0	1	
BS	1	1	0	1	2	0	1	0	0	0	1	2	0	0	0	1	0	1	0	0	0	2	1	4	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	2	
BULL	1	0	0	0	0	1	0	0	0	0	0	2	0	0	2	1	1	0	0	0	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
CAI	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	0	1	0	1	0	1	0	1	0	0	0	0	
CISCO	2	2	1	3	2	0	3	1	0	0	0	2	3	1	0	1	1	1	1	5	1	2	2	4	0	0	1	2	1	1	1	0	1	0	1	0	0	1	0	4		
COMPAQ	5	1	3	3	3	2	3	2	2	1	2	0	5	2	0	2	0	3	2	4	8	7	3	1	5	1	0	2	3	1	3	1	0	1	0	3	2	2	2	2		
DELL	4	0	1	4	1	1	0	0	0	1	3	5	0	1	0	1	0	2	2	3	5	1	0	3	1	0	1	2	0	2	0	0	1	0	3	1	3	2	2			
DISNEY	0	0	1	2	1	0	0	0	0	0	1	2	1	0	0	0	0	0	0	1	1	1	0	2	0	0	1	1	0	0	0	1	1	0	0	1	1	0	0	1	0	
EMC	0	0	0	0	0	1	0	0	2	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	
ERICSSON	2	1	0	0	1	0	1	1	1	0	1	2	1	0	0	0	1	1	1	0	1	2	2	1	3	1	0	1	0	1	2	0	1	0	1	0	1	0	0	1		
FT	0	1	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
FUJITSU	2	2	0	0	1	0	1	1	0	0	1	3	2	0	0	1	0	0	2	1	4	2	1	4	0	1	1	0	1	1	1	1	1	0	0	2	0	1	1	1		
HITACHI	2	0	0	0	0	0	0	0	0	0	1	2	2	0	1	1	0	2	0	1	2	1	0	2	0	1	0	1	2	0	1	2	0	0	4	1	0	0	0	0		
HP	5	0	1	5	1	1	1	0	0	0	5	4	3	1	1	0	0	1	1	0	5	0	3	1	1	0	1	4	0	0	1	1	3	0	1	1	2	1	1	2	1	
IBM	5	0	3	4	1	3	0	0	1	0	1	8	5	1	0	1	1	4	2	5	0	4	2	7	1	0	2	3	0	0	0	2	4	0	3	1	2	0	3	1	2	0
INTEL	2	1	0	0	2	1	1	2	0	0	2	7	1	1	0	2	0	2	1	0	4	0	1	9	1	0	1	0	2	2	1	2	0	0	2	0	0	2	0	1	2	
LUCENT	4	2	0	2	2	0	3	1	0	0	2	3	0	0	1	2	0	1	0	3	2	1	0	3	1	2	0	0	1	3	1	0	2	0	0	0	0	0	1	2		
MICROSOFT	2	2	2	5	4	3	6	4	1	0	4	1	3	2	0	1	0	4	2	1	7	9	3	0	0	1	2	3	1	2	2	4	2	2	0	3	1	4	4	4		
MOTOROLA	1	0	0	0	1	0	1	0	1	0	0	1	1	0	0	3	0	0	0	1	1	1	1	0	0	0	0	1	0	0	2	0	1	0	1	0	0	0	0	0		
NEC	0	2	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	1	0	0	0	2	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	
NETSCAPE	0	0	4	2	1	0	0	0	2	1	2	1	1	0	0	0	1	0	1	2	1	0	2	0	0	0	4	0	0	0	0	2	0	0	1	0	0	1	0	0	0	
ORACLE	1	0	2	2	1	1	0	0	1	2	2	3	2	1	1	0	0	1	4	3	0	0	3	1	0	4	0	0	0	1	0	2	0	0	1	1	1	1	1	1	1	
PHILIPS	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	1	2	0	0	2	1	1	0	0	0	0	0	0	1	0	2	0	0	1	1	1	1	1	1	1	
SBC	1	1	0	1	5	0	2	2	0	0	1	3	2	0	0	1	0	1	0	0	0	2	3	2	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	4	
SIEMENS	2	1	0	0	1	0	1	1	0	1	1	1	0	0	1	2	0	1	1	1	0	1	1	2	2	1	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	1
SONY	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	2	1	2	2	0	4	0	0	0	0	2	0	0	0	1	0	2	1	0	0	0	0	0	
SUN	0	0	2	1	1	1	0	0	0	1	1	1	1	1	0	1	0	0	0	3	4	0	2	2	1	1	2	2	0	0	0	1	0	2	0	1	0	1	0	1		
TCI	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	2	0	0	
TOSHIBA	4	1	0	0	0	0	0	0	0	1	0	3	3	0	0	1	0	2	4	1	3	2	0	0	1	0	0	0	1	0	1	2	0	0	0	1	0	0	1	0	0	
TW	0	0	1	1	1	0	0	0	0	0	1	2	1	1	0	0	0	0	1	1	1	0	0	3	0	0	1	1	1	0	0	1	1	2	1	0	0	0	0	0	0	
UNISYS	0	0	0	0	0	1	0	0	0	0	0	2	3	0	1	0	0	1	0	2	2	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
USW	1	1	0	1	4	0	1	2	0	0	4	2	2	0	0	1	0	1	0	1	0	2	2	4	0	0	0	1	1	4	1	0	1	0	0	0	0	0	0	0	0	

Figure 1: A Valued Graph with 6 Nodes

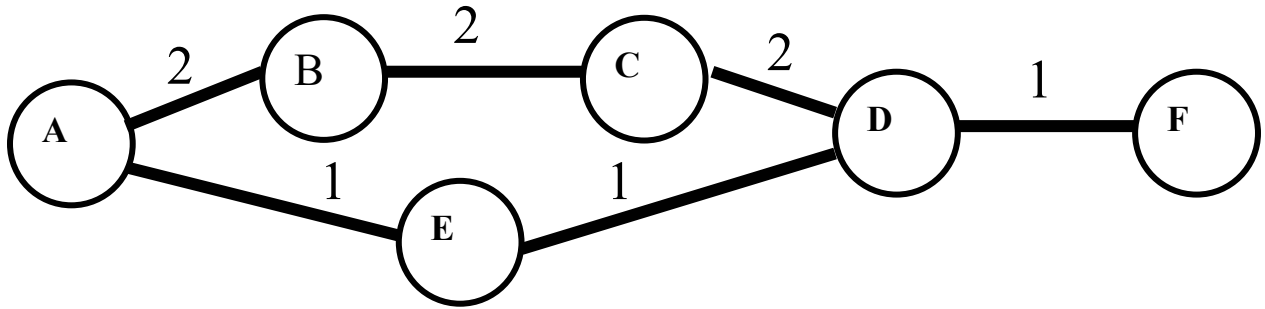


Figure 2: The Pseudocode of the MAPVC Algorithm with a single connected component as input

```

MAPVC( $c_i$ )                                     /*  $c_i$  is a connected component */

1.  for each connected component  $v \in c_i$  do {
1.1  Initialize a path along with distance of 1 and APV value which is the path value
      between  $v$  and every directly connected node

1.2  for each node  $i \in c_i$  do {                /*  $i$  is nodes other than  $v$  */
1.2.1 Select the next node with an unmarked path that has the highest APV value for
      node  $v$  and mark the corresponding path. Let's call this path as  $p$  and node as  $m$ .
1.2.2 Mark  $p$  as visited. Consider the path  $p$  and APV to extend in the following loop
1.2.3 for each node  $j \in c_i$  do {                /*  $j$  is not  $m$  or  $v$  */
1.2.3.1 If the path from  $v$  to  $i$  can be augmented by a path to  $j$  and it yields a smaller APV
      than the existing APV recorded for node  $j$ , append  $j$  to the path and record the
      new APV value of that path at node  $j$ .}
1.2.4 for each node  $j \in c_i$  do {                /*  $j$  is not  $v$  */
1.2.4.1 Among all paths attempted for node  $j$  in step 1.2.3, select the path for node  $j$ 
      with the largest APV and record the path along with the APV value
      }

1.3  for every node  $k \in c_i$  do {
1.3.1 Select the APV that is larger between a direct path from  $v$  to  $k$  and the indirect
      APV computed this far for  $k$ .}
}

```

Figure 3: Multidimensional Scaling of 38 IT companies

