Component of Statistics Canada
Catalogue no. 12-001-X Business Survey Methods Division

Article

The application of graph theory to the development and testing of survey instruments

SURVEY
METHODOLOGY

A JOURNAL
PUBLISHED BY
STATISTICS CANADA

by Steven Elliott

June 2012

Survey Methodology, June 2012
Vol. 38, No. 1, pp. 11-21
Statistics Canada, Catalogue No. 12-001-X

The application of graph theory to the development and testing of survey instruments

Steven Elliott [1]

Abstract

This paper focuses on the application of graph theory to the development and testing of survey research instruments. A graph-theoretic approach offers several advantages over conventional approaches in the structure and features of a specifications system for research instruments, especially for large, computer-assisted instruments. One advantage is to verify the connectedness of all components and a second advantage is the ability to simulate an instrument. This approach also allows for the generation of measures to describe an instrument such as the number of routes and paths. The concept of a 'basis' is discussed in the context of software testing. A basis is the smallest set of paths within an instrument which covers all link-and-node pairings. These paths may be used as an economic and comprehensive set of test cases for instrument testing.

Key Words: Graph theory; Computer Assisted Interviewing (CAI); Questionnaire development; Software testing; Basis testing; Test cases.

1. Introduction

Graph theory is a branch of mathematics which deals with collections of nodes and links. A visual representation of a collection of nodes and links is referred to as a 'graph'. Graphs have been used in many areas of study to model real-world phenomena. The earliest examples appear in the analysis of transportation logistics (Berge 1976, page VII). In such analyses, a graph-theoretic approach is useful for determining such things as a maximally efficient set of paths to cover a number of locations. The locations are represented by the nodes of the graph, and the links represent routes from one location to another.

Graph theory has applications also in survey methodology. If the questions in a survey questionnaire are represented as nodes and the routes of flow between questions are represented as links, then a graph may be used to model a questionnaire. As such, many of the theorems and descriptive measures from graph theory pertain to questionnaires. In addition, the processes of documenting and testing survey instruments benefit from a graph-theoretic approach. For example, a documentation system that contains one table for questions and another for response alternatives has the ability to verify the connectedness of all instrument components as well as perform simulations of a working instrument. A testing procedure in which the set of test cases minimally spans the 'basis' of an instrument graph guarantees that all combinations of consecutive links and nodes are tested with the smallest possible number of cases.

A graph-theoretic representation is not necessary for the development, documentation, or testing of most survey instruments. In most cases, survey instruments have relatively few questions and the routing through an instrument does not have many branching points. Examples of this are customer satisfaction surveys and short, paper-and-pencil surveys such as the U.S. Census. For these types of instruments, conventional documentation and testing procedures are adequate. However, large and complex surveys, like many current survey efforts, may benefit from a graph-theoretic approach. For example, the Canadian Financial Capability Survey (CFCS) is a survey that was conducted in 2009 to determine Canadians' knowledge and behavior with respect to financial decision making. It was a computer-assisted telephone interview comprised of 12 sections each of which had approximately 12 questions (Statistics Canada 2010). Another example is the Consumer Expenditure Surveys Quarterly Interview CAPI Survey (2010) conducted by the United States Department of Labor, Bureau of Labor Statistics. This survey has 22 sections most of which have 3 or more subsections, and within each subsection there may be as few as six or as many as 90 questions (US Bureau of Labor Statistics 2010). Either of these examples would be a good candidate for a graph-theoretic approach to documentation and testing.

This paper addresses the application of mathematical graph theory to survey research instruments. The next section of the paper which follows immediately below contains a description of a questionnaire as a graph and a delineation of the special properties that set apart a questionnaire graph from other types of graphs. The third section outlines the implications of a graph-theoretic representation on the structure of databases used for documentation/specifications systems for computer-assisted surveys. In Section 4, the specific features of graph-theoretic data structures are discussed. Sections 5 and 6 pertain to software testing and

1. Steven Elliott, Westat, Incorporated, 1500 Research Blvd., Rockville, MD 20850-3158, U.S.A. E-mail: sdelliott2@verizon.net.

the implications of graph theory on testing. A rationale is presented for the use of a 'basis' set of test cases which covers all pairs of linked nodes. This set of paths constitutes a comprehensive set of test cases for instrument testing.

2. A questionnaire as a graph

A graph may be represented as follows: $G = (V, E)$, where $V = \{v_1, v_2, v_3, ..., v_n\}$ is a set of nodes or vertices and $E = \{(v_i, v_j), (v_1, v_k), ...\}$ is a set of links or relations between pairs of vertices. Links are referred to as 'edges' in the terminology of graph theory, and hence the common usage of "E" to represent them (Chartrand 1985, page 27). A graph need not have any additional special characteristics. However, graphs which are attributed special characteristics are useful in modeling many phenomena in science and engineering. For example, graphs with undirected edges (*i.e.*, where both of the nodes attached to a link may be a predecessor or successor) may be used to model AC electric circuits, and graphs with directed edges may be used to model problems in traffic-pattern design. Other graphs with special characteristics are utilized to model networks in computer science, communications, sociology, and psychology.

In the case of survey questionnaires, the nodes of the graph represent different components or parts of a survey instrument. Most frequently, these are the substantive questions of a survey or decision points where routing is determined. The edges represent the response alternatives or outcomes associated with a node. Edges also represent the routing from one node to the next, and each edge has a unique predecessor and successor node. The graph depicted in Figure 1 represents a simple, 12-question survey instrument. The black circles (*i.e.*, nodes) represent the components of the instrument, and the lines connecting the black circles represent the edges that join one question to another. For example, the first node could represent a question with two response alternatives such as 'yes' and 'no'. The second node could represent a question with five response alternatives, where the first three alternatives branch to node 3, and the fourth and fifth alternatives branch to node 4.

When a graph is used to represent a questionnaire, there are a number of special properties that are attributed to the graph. These properties define the logical nature of a questionnaire. Bethlehem and Hundepool (2004) pointed out a number of these properties. First, a questionnaire has a starting node and an ending node. Second, all nodes other than the starting and ending nodes are connected. This means that for each node in the graph there is at least one route to it from the starting node, and one route away from it to the ending node. A third property of a questionnaire graph is that each of the edges is directed. This means that the route of flow from one node to another is always in one direction. A fourth characteristic of a questionnaire graph is that it may have multiple edges between a single pair of nodes. Many types of graphs are restricted such that only one edge may join a pair of nodes. This restriction does not apply to a questionnaire graph, because questionnaires commonly have more than one response alternative leading from one question to another. A final characteristic is that looping structures are permitted. This means that a node may appear multiple times on a single route. Looping structures are used frequently in questionnaires to modify responses that are determined to be incorrect. For example, financial or time-usage questions may be checked with edits that loop back if component questions do not sum to the correct total.

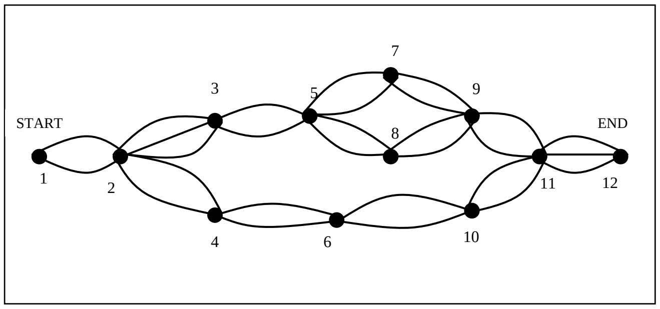


Figure 1 Representation of a Survey Instrument as a Graph

The characteristics of a questionnaire graph may be summarized as follows:

1. a starting node and an ending node,
2. connectedness (*i.e.*, each node is connected to the start and end nodes),
3. all edges are directed,
4. pairs of nodes may have multiple or parallel edges connecting them, and
5. nodes may appear more than once on a route.

Given a set of defining properties, it is possible to determine a number of descriptors including the number of routes and a basis. It is possible also to model a documentation system on the structure of the graph as illustrated in the next section.

3. Documentation and specification systems for survey questionnaires

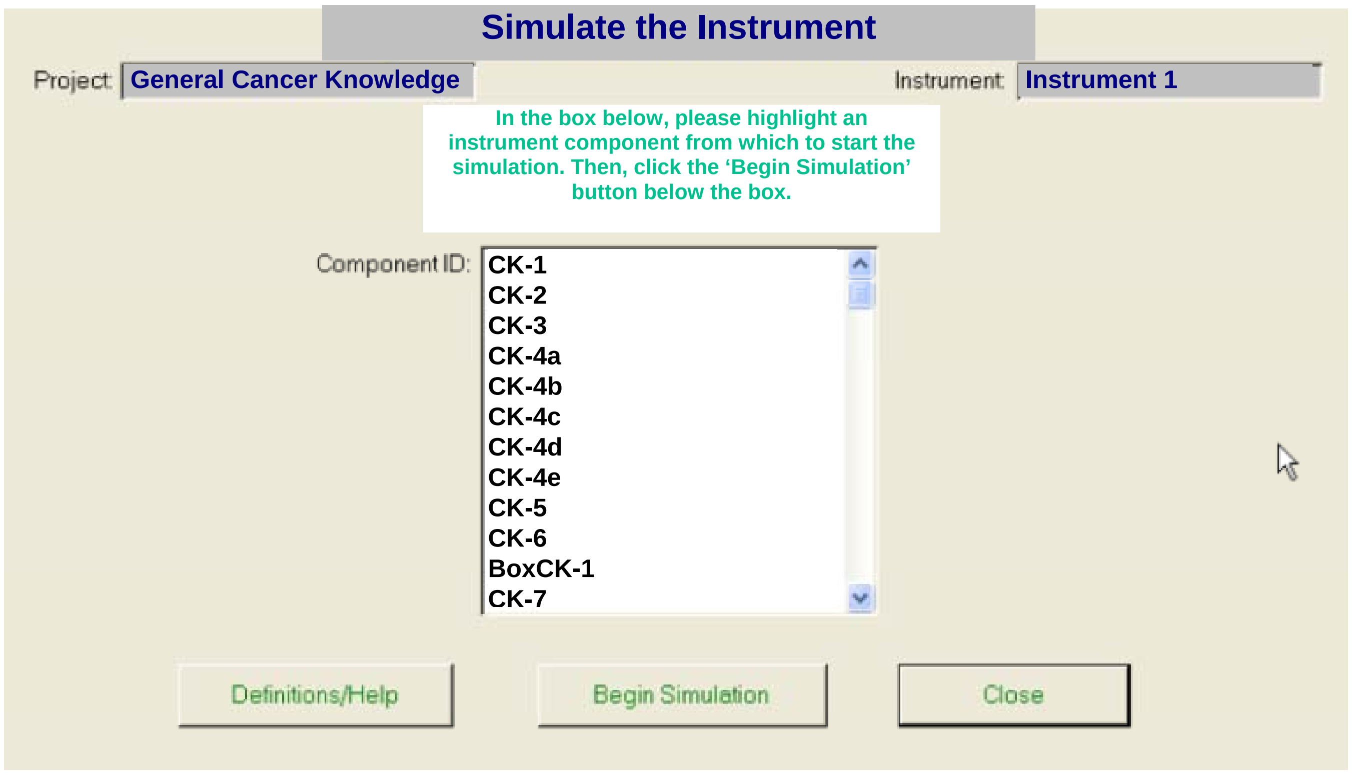
Questionnaire documentation systems are typically one of two types: a text document or a relational database. For text-document systems, the information pertaining to a substantive question or other type of instrument component is most often presented as a section of the document. It consists of the question text, response alternatives, routing, and instructions for programmers. The documentation system itself has no functionality aside from the search and print capabilities available in the word-processing software used to create the documentation. Systems using a relational database, on the other hand, are typically structured as a table where the rows represent the questions of the survey, and the columns represent attributes of the questions. Each record in the table is an n-tuple of question attributes. For example, the attributes of a question might include: name, sequence number, text of the question, response alternatives, routing information, and technical notes. One such specifications system is the *Tool for the Analysis and Documentation of Electronic Questionnaires* (TADEQ) (Bethlehem and Hundepool 2004). Other examples include systems developed at Westat Inc. for the *Medicare Current Beneficiary Survey* (MCBS) sponsored by the US Centers for Medicare and Medicaid Services (Medicare Current Beneficiary Survey: Overview 2010) and the *Medical Expenditure Panel Survey* (MEPS) sponsored by the US Department of Health and Human Services (MEPS: Survey Instruments and Associated Documentation 2010). These database systems have in common a structure of one primary table where each record represents a question.

Despite the advantages afforded by the straightforward nature of conventional systems, a specification system modeled like a graph has capabilities beyond those possible with a conventional structure. Before describing those capabilities and the necessary underlying structure, it should be noted that there are multiple ways in which a graph-theoretic data structure may be constructed (the interested reader is referred to Gibbons (1985, page 73) who described and categorized a number of those structures). The system proposed here is a relational list structure with two primary tables. One table represents the nodes of the graph, and the second table represents the edges. In the table representing nodes, each record or row represents an individual instrument component (*i.e.*, survey question, edit, or routing decision point). The second table represents edges where each record represents an individual edge (*i.e.*, a response alternative or a specific condition existing at a decision point). Each record from either table contains attributes associated with the record. Individual attributes are contained in the columns of the table. In the table of nodes, each column represents a specific attribute such as the component ID and component type. In the table of edges, each column represents an attribute such as the text of a response alternative. Two important distinctions between a documentation system with this structure versus a more conventional documentation system are: 1) the information pertaining to edges is not contained in the table for instrument components and 2) the table of edges (*i.e.*, links) contains identifiers for the predecessor and successor of an edge. As described in the next section, these distinctions allow a documentation system to perform in ways not possible with conventional systems.

4. Features of a graph-based specifications system

The use of separate tables for nodes and links as the building blocks of a specifications systems has several advantages. Most important of these advantages is the ability to simulate an interview. A developer or tester can move through an instrument selecting response alternatives while being routed from one instrument component to another just as if they were administering the instrument to a respondent. Figure 2 is an example of a screen display for simulating an instrument. The component from which simulation begins is selected from this screen. Figure 3 is the actual simulation screen itself. It shows the current component with the question text or conditional in the center of the screen. The lower left is a display of all components from which one may have come in order to arrive at the current component (*i.e.*, predecessors). These are referred to as 'origination points' in the screen display. The lower right is a display of destination points or components to which one may go from the current component (*i.e.*, successors). Thus, one may move through an instrument one component at a time in either direction by selecting either an origination point or a destination point. In Figures 2 and 3, the questionnaire used as an example is one on general knowledge about cancer, and the question depicted in Figure 4 has only one predecessor and one successor. This will be the case for most survey questions, however if multiple predecessors or successors did exist, they would be listed in the display.

The ability to simulate the operation of a survey instrument is made possible because a separate table is utilized for links. This table may be queried to find all predecessors and successors for any component in the questionnaire. During the design phase of development, this feature can be used to insure that all sections and questions are properly connected and all routing is correct. In the testing phase of development, this feature may be used to perform side-by-side comparisons of an instrument and the specifications upon which it was built. A tester could have the specifications system simulating the instrument on one monitor while running the actual instrument on a second. Such comparisons can be used to check not only the wording and formatting of questions and response alternatives, but also to verify that the instrument is going to the appropriate question at the appropriate time. Reports of errors or problems may then be entered directly into the specification system as an attribute of an instrument component.



Simulate the Instrument

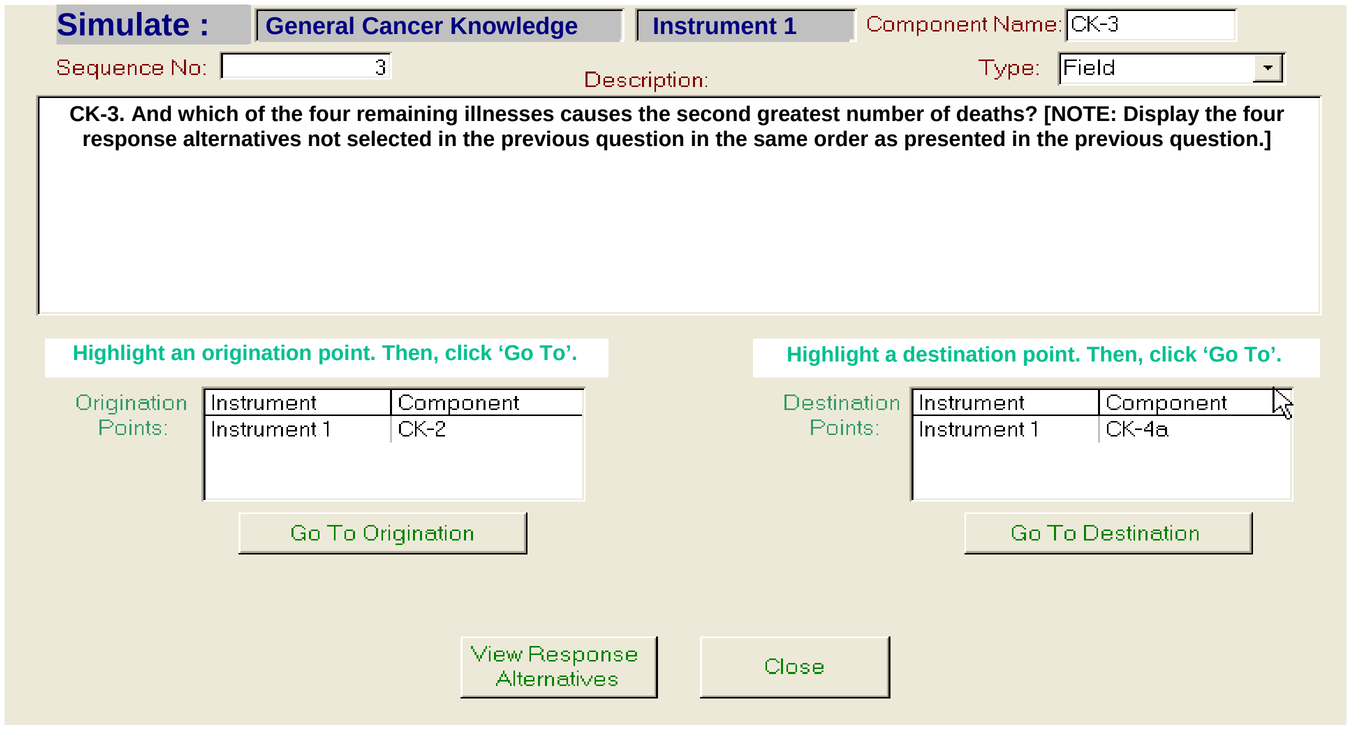
Project: General Cancer Knowledge

Instrument: Instrument 1

In the box below, please highlight an instrument component from which to start the simulation. Then, click the 'Begin Simulation' button below the box.

Component ID:
- CK-1
- CK-2
- CK-3
- CK-4a
- CK-4b
- CK-4c
- CK-4d
- CK-4e
- CK-5
- CK-6
- BoxCK-1
- CK-7

Figure 2 Begin simulation screen



Simulate : General Cancer Knowledge | Instrument 1 | Component Name: CK-3

Sequence No: 3

Type: Field

Description:

CK-3. And which of the four remaining illnesses causes the second greatest number of deaths? [NOTE: Display the four response alternatives not selected in the previous question in the same order as presented in the previous question.]

Highlight an origination point. Then, click 'Go To'.

Origination Points:

Instrument	Component
Instrument 1	CK-2

Highlight a destination point. Then, click 'Go To'.

Destination Points:

Instrument	Component
Instrument 1	CK-4a

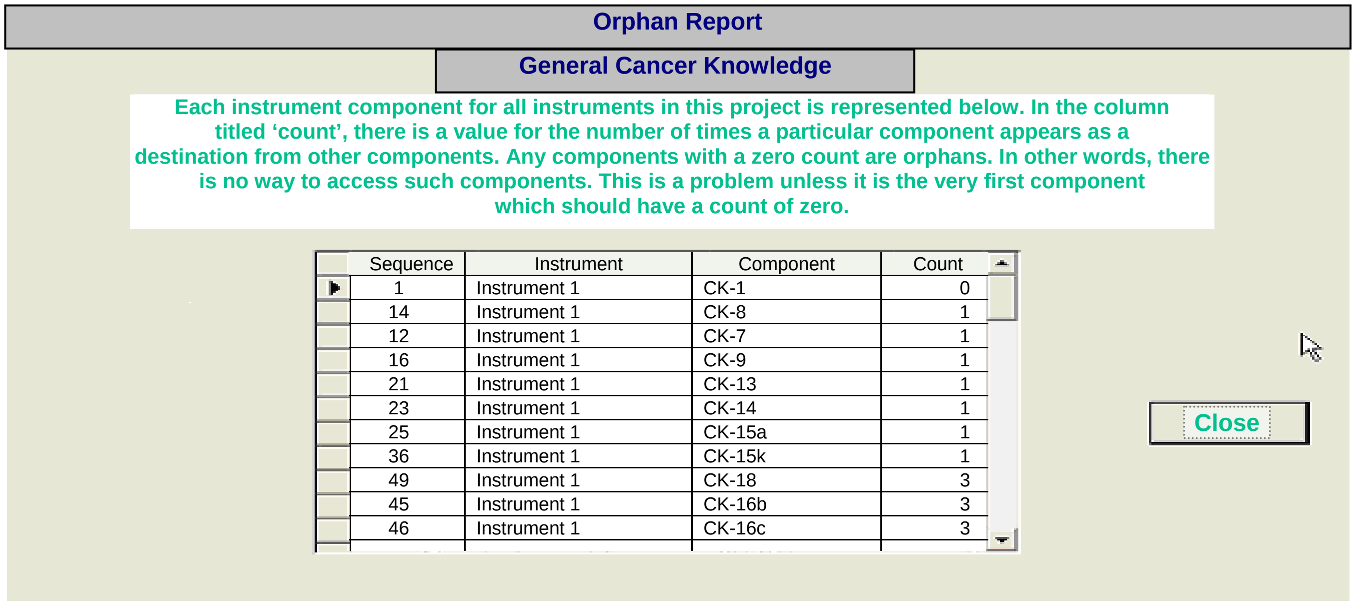
Figure 3 Simulation screen

Another method for evaluating the integrity of a questionnaire is to identify ‘orphan’ instrument components. Sometimes in the course of creating or modifying a questionnaire, an instrument component may become inaccessible. Such components are referred to as ‘orphans’. Since a table exists for links (*i.e.*, response alternatives and conditions), it is possible to run queries on this table to determine if a particular question appears as the successor to any link. If the question does not appear as a successor, then it is an orphan. Figure 4 contains the screen display for a listing of instrument components sorted by the frequency with which each appears as a successor. This is called an ‘Orphan Report’ in the figure. It shows that the first question in the survey has no origination points. This is as it should be since the first component cannot have predecessors. Any other component having zero origination points is an orphan. The orphan report is useful also in characterizing instrument components. For example, a question or component with a large number of originations may be the first question of a section devoted to handling premature terminations. Such a section is accessible from any other section of the interview, and therefore it would have a large number of predecessors.

5. Testing

Testing a computer-assisted survey instrument is the process of verifying that the behavior of the instrument is consistent with the design specifications. Several approaches have been utilized to accomplish this. One is to test first the building block components of a system, and then move to increasingly larger and more integrated assemblages of components (*i.e.*, ‘bottom-up’ testing). Testing the building block components is referred to as ‘unit testing’ (Beizer 1995, page 5). After each of the building blocks has been tested separately, the blocks are assembled, and testing is concentrated on how the components interact. This is referred to as ‘integration testing’ (Hetzel 1984, page 11). The final stage of integration testing is ‘system testing’ where the entire system as a whole just as it would be used in a true production environment (Myers 1979, page 110).

Other approaches and terminology have also been applied to testing procedures. These include ‘black-box’, ‘white-box’, and ‘regression’ testing. In black-box testing, a program is treated as if it were in a black box where the inner workings not visible. Inputs and outputs are the only observable aspects of program function (Beizer 1995, page 8). White-box testing utilizes knowledge of the program code to decide how to conduct the tests and which cases are used in testing (Patton 2006, page 55). For example, a programmer might conduct a series of white-box tests such that every line of code is ‘exercised’ (*i.e.*, ‘code coverage’) or such that every branching point is exercised (*i.e.*, ‘branch coverage’). Regression testing is used to insure code integrity after changes or additions have been introduced to an operational program (Beizer 1995, page 235). Regression tests utilize a set of test cases. This set is selected such that each of the major branches of the program is exercised. Other types of testing (*e.g.*, alpha, beta, usability) are also used in software development, and there are many sources for a more comprehensive description of testing procedures (see Kaner, Falk and Nguyen 1999, page 277).



Each instrument component for all instruments in this project is represented below. In the column titled ‘count’, there is a value for the number of times a particular component appears as a destination from other components. Any components with a zero count are orphans. In other words, there is no way to access such components. This is a problem unless it is the very first component which should have a count of zero.

Sequence	Instrument	Component	Count
1	Instrument 1	CK-1	0
14	Instrument 1	CK-8	1
12	Instrument 1	CK-7	1
16	Instrument 1	CK-9	1
21	Instrument 1	CK-13	1
23	Instrument 1	CK-14	1
25	Instrument 1	CK-15a	1
36	Instrument 1	CK-15k	1
49	Instrument 1	CK-18	3
45	Instrument 1	CK-16b	3
46	Instrument 1	CK-16c	3

Figure 4 Orphan report

In any testing procedure, a major concern is testing bias. This results when some components or functionality of an instrument are excluded from testing. For example, questions which appear toward the end of a survey or in an obscure section may be more likely to be excluded. Testing bias is eliminated completely if a set of test cases is selected such that all instrument components, links between components, and aspects of functionality are included. However, given the length and complexity of some surveys, comprehensive testing is not a practical option. Consider, for example, the questionnaire represented in Figure 1. This questionnaire has only 12 questions and 28 response alternatives, and yet, there are 672 possible routes through the instrument. In large surveys such as those mentioned above, the number of routes could be well over 10,000. Thus, if comprehensive coverage is not a viable approach for large surveys, it is possible to avoid testing bias by taking a probability sample of potential test cases. A graph-theoretic approach can be useful in both the specification of the universe of test cases and in the determination of a rational approach to sampling test cases.

6. A graph-theoretic approach to testing

A universe of test elements can be defined in several different ways. One could use the elements already discussed - test cases, where each case is a mock interview. Alternatively, a universe of test elements could be survey questions, response alternatives, or any of a variety of combinations of questions and response alternatives. The discussion here is limited to test cases, and therefore, it will be helpful to provide precise definitions of a test case and two closely related terms, 'path' and 'route'.

A path is a unique, ordered set of nodes, which traverses an instrument from beginning to end. Each node in a given path, provided that it is not a starting or ending node, is linked to a predecessor and a successor (this definition is consistent with Bethlehem and Hundepool 2004). A unique path results whenever a component has more than one successor component. In Figure 1, multiple successors appear for components 2 and 4. These two branching nodes result in three paths:

Path 1 - 1, 2, 3, 5, 7, 9, 11, 12
Path 2 - 1, 2, 3, 5, 8, 9, 11, 12
Path 3 - 1, 2, 4, 6, 10, 11, 12

A 'route', on the other hand, is a unique, alternating series of nodes and links beginning with the starting node and terminating with the ending node. Like a path, a route must satisfy the properties of connectedness and direction. 'Route' is the graph-theoretic term which is synonymous with what is commonly called a 'test case' in software testing. Since a route takes into account which link connects a pair of nodes, the number of routes in a graph is greater than or equal to the number of paths. The number of routes contained within a particular a path is equal to the product of the number of links between each pair of nodes along the path. Thus for the example in Figure 1, the number of routes for each path is:

Path 1 - 2 x 3 x 2 x 2 x 2 x 2 x 3 = 288
Path 2 - 2 x 3 x 2 x 2 x 2 x 2 x 3 = 288
Path 3 - 2 x 2 x 2 x 2 x 2 x 3 = 96

The total number of routes is the sum of routes over all paths (*i.e.*, 288 + 288 + 96 = 672). A formula for computing the number of routes is:

$$\text{Routes} = \sum_i^P \prod_j^{NP_i} \text{links}_{ij}$$

where $\boldsymbol{i}$ represents the i^{th} path, $\mathbf{P}$ represents the total number of paths, $\boldsymbol{j}$ represents the j^{th} set of links on a given path, $\mathbf{NP}_{\boldsymbol{i}}$ represents the number of pairs of connected nodes on a given path, and *links* represents the number of links connecting a pair of nodes.

If a testing protocol is based on a sample of routes, then a minimum and comprehensive suite or universe of test cases is contained in the 'basis' of a graph. The term, 'basis', in this context is analogous to a 'basis' in geometry. The basis of a geometric space is a set of vectors which is sufficient to span the space, or in other words, a basis is a set of vectors sufficient to locate any point in the space. Likewise, the basis of a graph is a set of paths sufficient to include all predecessor-successor pairings of nodes. This implies that all nodes and at least one of the links between any connected pair of nodes are included. A basis is a subset of all possible paths. All questionnaires have a set of paths (**P**) in which each member satisfies the definition of a path as stated above (*i.e.*, a unique sequence of nodes). Within this set is a subset which has the special characteristic that each member path contains at least one pair of connected nodes that is not contained in any other path within the subset. This subset will be referred to as 'basis paths' (**BP**).

In order to gain a better understanding of the difference between the paths in **BP** and those in the complement of **BP** (*i.e.*, **P** – **BP**), consider the graph presented in Figure 5. The set of all paths (**P**) for the graph in Figure 5 is:

Path 1 - 1, 2, 4, 5, 7
Path 2 - 1, 2, 4, 6, 7
Path 3 - 1, 3, 4, 5, 7
Path 4 - 1, 3, 4, 6, 7

Any one of the four paths could be eliminated and the remaining three would include each pair of connected nodes, and therefore any three constitutes a set of basis paths (**BP**). For example if Path 1 were eliminated, each of the node pairings would still be contained in Paths 2, 3, and 4. However, if both Paths 1 and 2 were eliminated, then node pairings 1 - 2 and 2 - 4 would be excluded. Thus, the set of two paths would be insufficient to span all of the independent sequences of nodes in the graph.

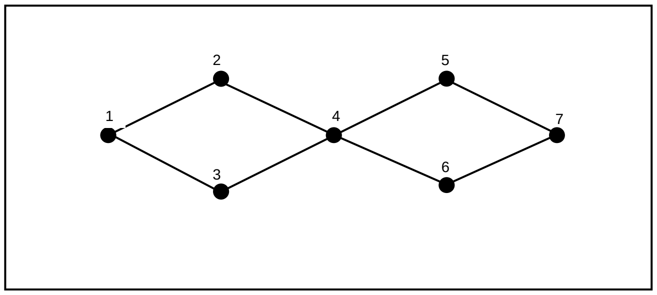


Figure 5 Representation of paths and basis paths

As illustrated above in Figure 1, many questionnaires encountered in practice have so many routes that testing all routes is not practical. Further, a typical route within an instrument has one or more similar routes which involve the same set of nodes, and these routes may be so similar that they differ by only a single, parallel link. Therefore, testing all routes would be not only impractical due to the large number of routes, but also redundant due to the similarity of many routes. The task for a test designer is to select a subset of routes that maximizes coverage and minimizes redundancy. This may be accomplished by using **BP** as a first step in sampling from the universe of routes. The utilization of **BP** in this manner is equivalent to beginning the sampling process with a purposive sample (Cochran 1977, page 10). Another way to think of this first step is as a redefinition of the universe of elements for the purpose of eliminating redundancy. This universe is comprehensive in its coverage, and it contains the smallest set of cases necessary to include all connected node pairs. A second stage of sampling could then be to select one or more routes from each of the paths contained in **BP**. This could be accomplished in several ways. One way would be to consider each path as a cluster of test cases and then take a probability sample from each cluster. Another way would be to select one route from each cluster by randomly selecting one parallel link at each node.

If one accepts the notion of basis testing, then it must be determined how much of the basis should be tested. If all paths in **BP** are tested, then the only elements of an instrument excluded from testing are redundant links. While redundant links may contain spelling or formatting errors, they are unlikely to contain routing errors. This stems from the nature of the programming task involved in creating CAI instruments. Response alternatives are typically 'bundled' in the sense that alternatives which lead to the same next question are likely to be either all misdirected or none misdirected. For this reason, comprehensive testing of a basis is an effective method for minimizing errors of a type most likely to lead to loss of data.

On the other hand, non-comprehensive testing may be the only reasonable strategy if constraints due to time or level of effort exist and the number of paths in a basis is large. Despite the fact that any part of an instrument not tested may contain an error, any fraction of the paths in a basis may constitute an unbiased test. Thus, the percentage of paths to be included in a test should probably depend on factors specific to a particular development situation. For example, an instrument may contain modules which have been used previously or modules that have had only minor modification since previous use. These modules need not be tested as thoroughly as newer ones. As a general rule, a minimum sample of test cases should include each distinct section of an instrument in one or more paths, and paths should be included to cover all inter-sectional connections.

7. Discussion and conclusions

A graph-theoretic approach to software development has two major advantages over conventional approaches. First, it allows for a documentation system that can simulate the behavior of a computer-assisted interview. This is useful in verification of routing and as an aid to testers in side-by-side comparisons of instrument behavior versus design specifications. The second major advantage is in selecting cases for testing. The use of the basis of a questionnaire allows for the specification of a universe of test cases which covers all node pairings with a minimum number of paths. Probability sampling from this universe insures that no bias is incorporated into the testing procedures.

In practice, the first advantage can be achieved by structuring the database behind a specifications system such that it contains a table for nodes and a table for links. If the links table specifies a predecessor and a successor node, then queries of the tables will provide the functionality for verification of routing and simulation. The second advantage can be achieved with an algorithm for the identification of a basis. As pointed out by Poole (1995), one of the most important things to do when setting out to test software is to determine which test cases to use. He presented an algorithm for doing this that is based on the flowgraph of a program. Using a flowgraph for this purpose is useful as long as the program is not too large. With large and complicated programs, flow diagrams become unwieldy. The same is true of large and complicated questionnaires (Bethlehem and Hundepool 2004). The appendix contains output from an algorithm which generates a basis, counts routes, and specifies basis paths for an example questionnaire graph (the algorithm used to generate the output appearing in the appendix is available from the author (sdelliott2@verizon.net). This algorithm does not handle looping structures as would be inherent in edits or 'go back' features. These structures may be tested as separate from the questionnaire graph. An algorithm which handles looping is under development).

A graph-theoretic approach is valuable also in that it allows for the use of a number descriptive measures of questionnaires such as the number of routes, the number of paths, cyclomatic complexity (cyclomatic complexity is a measure of complexity in software code (see Hetzel 1984; McCabe 1976; and Watson and McCabe 1996). It is equal also to the number of paths in the basis of a graph. For directed graphs where parallel links are not permitted, cyclomatic complexity $(\mathbf{CC}) = \mathbf{L} - \mathbf{N} + 2,$ where $\mathbf{L}$ is the number of links and $\mathbf{N}$ is the number of nodes), and several types of descriptive matrices (see appendix). Future enhancements to a graph-theoretic approach will likely involve such things as: 1) taxonomies for components, links, and errors; 2) secondary tables in the specification database containing attributes specific to different types of nodes and links; 3) sophisticated sampling plans for selecting test cases; and 4) purposive route sampling.

Taxonomies will promote the specification of special types of instrument components and the incorporation of secondary tables in the documentation system. An example of a special type of instrument component is one with a randomization feature. Such a component would be used in multi-phase respondent selection where a respondent reporting a particular disease, for example, has an increased probability of being routed to a follow-up section pertaining to that disease. In this case, the initial question pertaining to the disease may be a special type called 'respondent selection'. A secondary table in the documentation system for 'respondent selection' questions may have attributes pertaining to a random number generator such as generator seed and selection threshold.

Enhancements to sampling may include stratified sampling (Cochran 1977, page 89) and sampling with probability proportional to size (*i.e.*, PPS). Stratified sampling could be used to insure that all sections within a questionnaire are included with certainty. Paths would be stratified according to the sections they traverse. With PPS sampling, size might be a measure of path length, and the probability of selection for a particular path would be dependent on the number of nodes included in the path. Thus, longer paths could be included with greater frequency. Purposive route sampling may be utilized for testing instrument characteristics other than programming errors. For example, later phases of questionnaire development might target specific sequences of questions for tests of the cognitive characteristics of an instrument.

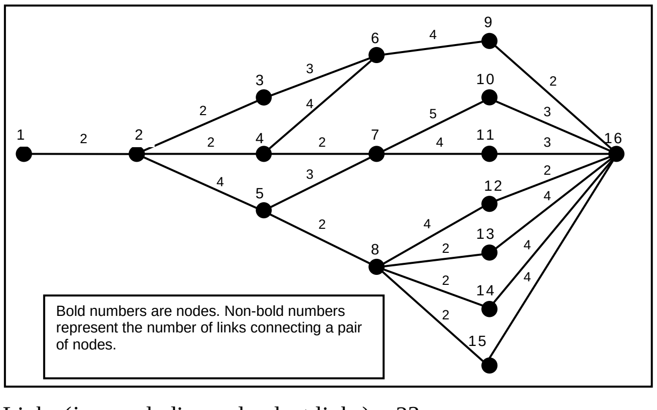
Other researchers in this area likely will provide further enhancements to the application of graph theory to questionnaire development. It does seem clear that graph theory lends itself well to the description, development, and testing of complex CAI instruments. The current trends in CAI usage seem to be in the direction of more sophisticated and larger instruments. For this reason, tools which help to document instrument components and identify errors are valuable to development efforts.

Acknowledgements

A significant portion of the work that went into this paper was supported by Westat, Incorporated Rockville Maryland. Substantial improvements and contributions to this paper were made by the reviewers and editors of *Survey Methodology*.

Appendix

Example of Basis Generation


Bold numbers are nodes. Non-bold numbers represent the number of links connecting a pair of nodes.

Links (*i.e.*, excluding redundant links) = 23
Nodes = 16

Figure 6 Questionnaire graph

Table 1
Branches count for each node

Node Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Branches	1	3	1	2	2	1	2	4	1	1	1	1	1	1	1	0

Table 2
Link matrix

Node	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1		2														
2			2	2	4											
3						3										
4						4	2									
5							3	2								
6									4							
7										5	4					
8												4	2	2	2	
9																2
10																3
11																3
12																2
13																4
14																4
15																4
16																

Each cell contains a value for the number of links between the row and column nodes.

Table 3
Path matrix

	1st node	**2nd node**	**3rd node**	**4th node**	**5th node**	**6th node**	**7th node**	**8th node**	**9th node**	**10th node**
Path 1	1	2	3	6	9	16				
Path 2	1	2	4	6	9	16				
Path 3	1	2	5	7	10	16				
Path 4	1	2	4	7	10	16				
Path 5	1	2	5	8	12	16				
Path 6	1	2	5	7	11	16				
Path 7	1	2	4	7	11	16				
Path 8	1	2	5	8	13	16				
Path 9	1	2	5	8	14	16				
Path 10	1	2	5	8	15	16				

Cell values represent nodes. Each row represents a path.
[*Note*: *The paths in this example all have 6 nodes. However in general, all paths will not have the same number of nodes.*]

Table 4
Link counts and number of routes for each path

	Node Pairings					**Routes**
	1st to 2nd	**2nd to 3rd**	**3rd to 4th**	**4th to 5th**	**5th to 6th**	
Path 1	2	2	3	4	2	96
Path 2	2	2	4	4	2	128
Path 3	2	4	3	5	3	360
Path 4	2	2	2	5	3	120
Path 5	2	4	2	4	2	128
Path 6	2	4	3	4	3	288
Path 7	2	2	2	4	3	96
Path 8	2	4	2	2	4	128
Path 9	2	4	2	2	4	128
Path 10	2	4	2	2	4	128

Paths = 10 Total Routes = 1,600
Cells represent the number of links between successive nodes in a path.

Table 5
Basis path matrix

	1st node	2nd node	3rd node	4th node	5th node	6th node	7th node	8th node	9th node	10th node
Basis Path 1	1	2	3	6	9	16				
Basis Path 2	1	2	4	6	9	16				
Basis Path 3	1	2	5	7	10	16				
Basis Path 4	1	2	4	7	10	16				
Basis Path 5	1	2	5	8	12	16				
Basis Path 6	1	2	5	7	11	16				
Basis Path 7	1	2	5	8	13	16				
Basis Path 8	1	2	5	8	14	16				
Basis Path 9	1	2	5	8	15	16				

Cell values represent nodes. Each row represents a basis path.

Table 6
Link counts and number of routes for each basis path

	Node Pairings					Routes
	1st to 2nd	2nd to 3rd	3rd to 4th	4th to 5th	5th to 6th	
Basis Path 1	2	2	3	4	2	96
Basis Path 2	2	2	4	4	2	128
Basis Path 3	2	4	3	5	3	360
Basis Path 4	2	2	2	5	3	120
Basis Path 5	2	4	2	4	2	128
Basis Path 6	2	4	3	4	3	288
Basis Path 7	2	4	2	2	4	128
Basis Path 8	2	4	2	2	4	128
Basis Path 9	2	4	2	2	4	128

Basis Paths = 9 Total Routes in Basis = 1,504
Cells represent the number of links between successive nodes from the Basis Paths Matrix above.

References

Balakrishnan, V.K. (1997). *Graph theory*. New York: McGraw Hill, Inc.

Beizer. B. (1995). *Black-box testing*. New York: John Wiley & Sons, Inc.

Berge, C. (1976). *Graphs and hypergraphs*. New York: North-Holland Publishing Company - Amsterdam, London and American Elsevier Publishing Company, Inc.

Bethlehem, J., and Hundepool, A. (2004). TADEQ: A tool for the documentation and analysis of electronic questionnaires. *Journal of Official Statistics*, 20, 233-264.

Centers for Medicare and Medicaid Services (2010). *Medicare Current Beneficiary Survey*: *Overview*. August 2002. Internet address: https://www.cms.gov/LimitedDataSets/11_MCBS.asp.

Chartrand, G. (1985). *Introductory Graph Theory*. New York: Dover Publications, Inc., Mineola.

Cochran, W.G. (1977). *Sampling Techniques*, 3rd Edition. New York: John Wiley & Sons, Inc.

Cohen, J. (1997). *Design and methods of the Medical Expenditure Panel Survey Household Component*. Rockville (MD): Agency for Health Care Policy and Research. MEPS Methodology Report No. 1. AHCPR Pub. No. 97-0026.

Couper, M.P., Baker, R.P., Bethlehem, J., Clark, C.Z.F., Martin, J., Nicholls, W.L. and O'Reilly, J.M. (1988). *Computer Assisted Information Collection*. New York: John Wiley & Sons, Inc.

Gibbons, A. (1985). *Algorithmic Graph Theory*. Cambridge University Press.

Harary, F., and Palmer, E. (1973). *Graphical Enumeration*. New York: Academic Press.

Hetzel, W. (1983). *The Complete Guide to Software Testing, QED*. Massachusetts: Information Sciences, Inc., Wellesley.

Kaner, C., Falk, J. and Nguyen, H. (1999). *Testing Computer Software*, 2nd Edition. New York: John Wiley & Sons, Inc.

Medical Expenditure Panel Survey (MEPS) (2010). *Survey Instruments and Associated Documentation*. http://www.meps.ahrq.gov/mepsweb/.

Myers, G.J. (1979). *The Art of Software Testing*. New York: John Wiley & Sons, Inc.

Patton, R. (2006). *Software Testing*, 2nd Edition. Sams Publishing, Inc.

Poole, J. (1995). NISTIR 5737 – A Method to Determine a Basis Set of Paths to Perform Program Testing, National Institute of Standards and Technology, Gaithersburg, MD, November, 1995.

Statistics Canada (2010). *Canadian Financial Capability Survey* (*CFCS*): *Questionnaire 2009*. http://www.statcan.gc.ca.

Statistics Netherlands (2002). Blaise Developer's Guide. Department of Statistical Informatics, Statistics Netherlands, Heerlen.

US Bureau of Labor Statistics (2010). *Consumer Expenditure Surveys Quarterly Interview CAPI Survey 2010.* United States Department of Labor, http://www.bls.gov/cex/capi/2010/cecapihome.htm.

Watson, A., and McCabe, T. (1996). NIST Special Publication 500-235 Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric. National Institute of Standards and Technology, Gaithersburg, MD, September, 1996.