Group Decision and Negotiation (2019) 28:543–563
https://doi.org/10.1007/s10726-018-9607-5

# The Analysis of Interconnected Decision Areas: A Computational Approach to Finding All Feasible Solutions

Ion Georgiou[1] · Joaquim Heck[1] · Andrej Mrvar[2]

Published online: 8 December 2018

**Abstract**
This paper provides a method for finding the complete set of feasible solutions to a problematic situation, whose structure is that of a network amenable to the analytical approach known as "analysis of interconnected decision areas", or AIDA. In doing so, the paper not only resolves a long-standing computational problem, but also offers means for examining all solutions in either lists or diagrams, thus empowering decision-makers to make informed judgments as to how to tackle an entire problem or its subsets. The analytical advantage of using a signed graph in AIDA computations is demonstrated, proffering an innovative contribution to the approach. The paper concludes by identifying potentially fruitful avenues of future research as well as interdisciplinary opportunities.

**Keywords** Computational model · Computer supported design · Networks · Decision-making · *Pajek* software

## 1 Analysis of Interconnected Decision Areas: Characteristics and Questions

In the mid-1960s, Harary et al. (1965) published a paper that described the existence of problematic situations defined by the following characteristics:

✉ Ion Georgiou
Phokion.Georgiou@fgv.br; iongeorgiou@gmail.com

Joaquim Heck
joaquimheck@uol.com.br

Andrej Mrvar
Andrej.Mrvar@fdv.uni-lj.si

[1] Departamento de Informática e Métodos Quantitativos (IMQ), Fundação Getulio Vargas (FGV), Escola de Administração de Empresas de São Paulo (EAESP), Rua Itapeva 474 – 9 andar, sala 901, Bairro Bela Vista, São Paulo, SP 01332-000, Brazil

[2] Faculty of Social Sciences, University of Ljubljana, Kardeljeva pl. 5, 1000 Ljubljana, Slovenia

- The problematic situation is composed of a number of decision areas;
- Each decision area indicates an aspect of the situation where a choice must be made between two or more mutually exclusive options;
- An option in one decision area may or may not be compatible with an option in another decision area; and,
- A combination of compatible options, one from each decision area, provides a feasible solution[1] to the problematic situation.

Given these characteristics, Harary et al. (1965) then asked two questions:

1. How many feasible solutions exist? and,
2. What options compose each of the feasible solutions?[2]

Harary et al. (1965) called their computational approach the "analysis of interconnected decision areas", or AIDA, and, using graph theory, provided a solution for a special case of seven options distributed across three decision areas. Although they asserted the generalization of their approach for any number of decision areas, with any number of options distributed between them, they did not demonstrate this. Instead, some years later, Harary (1969: 151) demonstrated a much simpler, and elegant, solution for the aforementioned special case, using the cubed adjacency matrix. The provision of a generically applicable approach poses a non-trivial combinatorial challenge in the form of a network. What follows addresses this challenge, and thereby provides a means for answering the two questions for any number of decision areas and options.

The paper begins by reviewing the manner in which AIDA has been discussed across numerous disciplines. Attention is paid to discussions to be found in the planning literature, in particular those on the Strategic Choice Approach. The conclusion is drawn that the computational challenge posed by AIDA has yet to be fully addressed, and practical reasons for resolving it are given. The point is also made that, notwithstanding the advantages of incorporating it into other methods, AIDA is a stand-alone method with a clearly defined, specific purpose, and is treated as such in the ensuing discussion.

Addressing the computational challenge is sourced in graph theory, with the innovative introduction of a signed graph as central to the resolution. Relevant theory is introduced from subgraph isomorphism procedures and backtrack programming. Pseudocode of an algorithm that performs the analytical task is given in full, and software in which the pseudocode is programmed is introduced. Throughout, the discussion is accompanied by a published, worked example drawn from the decision-making field, and illustrated with multiple figures that also include additional details. The paper ends by emphasizing how the procedure empowers decision-makers, and the conclusion discusses potentially fruitful avenues of future research as well as interdisciplinary opportunities.

[1] The term "feasible solution" is one used by Harary et al. (1965), who also use the term "α-combination". Synonymous terms in the literature include "solution stream" (Hickling 1978: 473), "feasible strategy" (Friend 1992: 160), "compatible set" (Weas and Campbell 2004: 233), and "decision scheme" (Friend and Hickling 2005: 37–38, 67–69, 130–135).

[2] Harary et al. (1965) also asked a third question concerning the "cost", or weight, of each feasible solution. This is a simple matter of appending coefficients to options that constitute a feasible solution, and is, therefore, not addressed in this paper.

## 2 Motivations for Answers

The need for a formal computational approach for the general case was first recognized in the field of design studies. Smith and Morrow (1999: 257) describe AIDA as a method that:

> assumes that there is a number of subproblems within every design problem, and every subproblem has a variety of possible solutions. Some of the subsolutions will be incompatible with solutions to other subproblems. If we know what these incompatibilities are, then it is possible to find feasible sets of solution with known compatibilities.

They note that AIDA highlights an important part of a design process: “that many design subcomponent decisions are interrelated.” They then add, however, that there is a “lack of application” of AIDA and suspect “that there are difficulties in its applicability.” In particular, they point out:

> It is possible that the amount of predictive knowledge concerning whether certain subsolutions are incompatible exceeds the amount typically available at the time that decisions are made, or collecting this information is prohibitive.

In other words, Smith and Morrow are asking for a non-prohibitive computational approach that can provide decision-makers with complete information on all feasible solutions and their constituent options. With such information at hand, informed decision-making may proceed, either to tackle the entire problem or some subset.

Despite Smith and Morrow’s appeal, design studies has not tackled the challenge. One of the field’s pioneers, Jones (1970/1992: 310–315), in his seminal book *Design Methods*, describes AIDA as “one of the most powerful and reliable of the design methods [for exploring problem structure].” Nevertheless, since the inception of the flagship journal, *Design Studies*, interest in AIDA has remained on a theoretical level (Jones 1979: 35; Bayazit et al. 1981: 219; Gat and Gonen 1981: 171, 174; Hsiao and Chuang 2003: 156; Dorst and Royakkers 2006: 654).

Computational discussions, from fields as varied as production/manufacturing, engineering, and computing itself, have made little progress. Instead, they betray reversions to laborious manual manipulations, as well as mathematical diversions. The former case is illustrated by Burbidge (1973: 320) and Blandford and Hope (1985: 209) who offer the multiplication rule for finding all possible combinations between options. In what amounts to spadework for finding feasible solutions, the approach of these authors ignores incompatibilities between options as well as their mutual exclusivity within decision areas. The case of mathematical diversions is illustrated by Hope and Sharp (1989) and Wu et al. (2015) who choose to focus on weighted options. These authors take a methodological leap sideways which enriches AIDA through use of the Analytical Hierarchy Process (Saaty 1980). The question of how the total number of feasible solutions, and their constituent options, may be found in the first place, however, remains unanswered. Occasionally, such as in the field of ergonomics (Hsiao and Chou 2004: 432–433), researchers have simply stated they found all solutions, but no verifiable computational method is presented.

The practical application of AIDA has largely been of interest in the planning and related decision science fields. Here, AIDA is invariably discussed in the context of the Strategic Choice Approach (SCA) (Friend and Hickling 2005), wherein it performs a crucial role in identifying feasible solutions, albeit within a reduced computational context. For SCA is less concerned with complete computation and more with facilitating interactive groups to explore manifold aspects of a shared problematic situation. Aspiring to offer such assistance, SCA is designed to be sensitive to the human context of sense-making, negotiation, and accommodation, the focus being to aid judgment. Thus, in its first two "shaping" and "designing" phases (and, especially, in the latter where AIDA is explicitly used), SCA is not concerned in uncovering all possible feasible solutions to a problematic situation; it consciously reduces the number of decision areas, and (usually) the number of constituent options, as a means of avoiding cognitive overload (Kirsh 2000) in the interest of working toward some commitment to implementable decisions (Friend and Hickling 2005: 105). SCA, therefore, is an approach focused on tackling issues far beyond the computational challenge posed by AIDA (e.g. Rosenhead 1996; Friend and Hickling 2005: 295–360; e Costa et al. 2014; Sharifzadegan et al. 2014; Norese et al. 2015; Todella et al. 2018). For this reason, SCA uses AIDA as "an informal kind of 'intermediate technology' to aid communication within decision-making groups" by providing "graphical expression of complex problem structures" (Friend et al. 1988: 709).

Perhaps due to the emphasis on an "intermediate" use of AIDA, proponents of SCA have offered only scattered and limited comments on AIDA's computational procedure. For example, references are made to the possibility of using hand calculations or computers, but no instructions are given (Luckman 1967: 351–353, 357; Friend 1989: 138, 139). Hickling (1978: 459, 473) mentions a seemingly "straightforward logic" for finding the number of "combinations [read: feasible solutions]", but does not elaborate. Friend (2001: 129–131) and Friend and Hickling (2005: 37–39, 130–131, 135, 249–251) mention a rule of thumb for estimating answers, but provide no means for finding exact answers. Phahlamohlaka and Friend (2004: 687) state that "basic combinatorial methods" suffice to uncover feasible solutions, but they neither state what these "basic" methods are, nor demonstrate their sufficiency. As will be shown below, AIDA presents a networked combinatorial problem that demands somewhat more versatile methods brought together in a computer algorithm.

To be sure, computer software for resolving an AIDA combinatorial problem is available. Its documented history, however, is either ambiguous or resorts to a reduced computational context. In the late 1960s, Luckman (1967: 353) mentioned that "a computer programme has been developed" but gave no further details. Ten years later, Jones (1979: 35) wrote that AIDA "make[s] use of computer programs," but again no details were offered. Instead, both in the early 1970s and early 1990s, Jones (1970/1992: 313) simply pointed to "a computer program available at the Institute for Operational Research, 42B New Union Street, Coventry, Warwickshire, England, and suitable for use on any machine which has an ALGOL or a FORTRAN compiler." This is echoed by Friend and Hickling (2005: 245) who mention that "a program to deal with the combinatorial aspects of AIDA was developed in FORTRAN by Hadley Hunter with other colleagues in the Institute of Operational Research," but no additional information is given. Friend (1992: 160) has written about software, known as STRAD, that returns feasible solutions "available under any given set of assumptions about the compatibility or otherwise of each pair of options drawn from different but inter-connected decision areas". This software, however, is limited to a computational maximum of 24 decision areas and, furthermore, no details on how the calculation is done are available (Friend et al. 2002; Stradspan Ltd. 2014). In addition, the software is operationally limited: the latest version of the software, STRAD 2.3, was released in 2002 and "will run on 16-bit and 32-bit versions of Microsoft Windows 95, 98, NT4, 2000, XP and Vista" (Stradspan Ltd. 2014). For meeting the intricate combinatorial complexity of AIDA, the above provide restrictive assistance to would-be practitioners. In what follows, all limitations are removed.

In removing such limitations, this paper responds to a variety of requirements identified by Friend (2014) in a keynote presentation to the Public Policy stream at the Operational Research Society's 56th annual conference in September 2014, a presentation expanded into a 40-page review, available at the IOR Legacy Section of the Document Repository of the Society. With especial regard for "the design of interactive and inclusive processes for developing public policies," "strategies for interactive policy design," and "interactive methods for problem structuring," Friend (2014: 1, 18, 21, 23, 31) calls for "the refinement for wider operational use of analytical tools that were developed in prototype form in earlier IOR [Institute of Operational Research] projects." Such refinement is to include: "building further momentum in developing technology; [providing] advice on directions of development, with suggestions for linkage to other information systems or forms of decision support software that may already be in use; [and] re-engineer[ing] the existing STRAD software to be compatible with the latest internet-enabled operating systems," and "upgrad[ing] [STRAD as a] proven decision process support software" (Friend 2014: 27, 35). The overall objective is to "[design] software tools to enable researchers and policy-makers to develop multi-layered maps of both structural and adaptive relationships in any domain of public policy [and] to help researchers and project managers in tracing the impacts of the different programmatic influences that impinge on those involved in negotiated project engagements" (Friend 2014: 31).

Friend's keynote makes it clear that the usefulness of SCA, and, by implication, any method addressing "interactive policy design," is increased by the availability, should it be required, of extended computerized computational assistance. For AIDA, this means that, given any number of decision areas, composed of any number of options, decision-makers should have the means, if they so desire, to obtain the total number of feasible solutions and their constituent options. In other words, where Friend requests means for "tracing the impacts of *different* programmatic influences," this paper goes one step further and provides, for a situation structured using AIDA, the means for tracing *any* and/or *all* such influences. In so doing, this paper "refines" one of the best known "tools that were developed in prototype form in earlier IOR projects"—i.e. AIDA; contributes to "building momentum in developing technology;" provides "suggestions for linkage to other information systems or forms of decision support software that may already be in use;" opens an opportunity to "upgrade [STRAD as a] proven decision process support software," and thereby contributes to enhancing the power of SCA; and, in addition, hints at the possibility for developing "multi-layered maps."

In order to do this, the wide range of issues discussed thus far must be temporarily set to one side—the Conclusion will return to them. What follows addresses AIDA as an independent approach to uncovering feasible solutions—that is, outside the context of SCA or any other method that might wish to incorporate it. This is in keeping with the manner in which AIDA was first presented by Harary et al. (1965). The position taken is that the quantity and quality of AIDA analysis, and the level of confidence with which decision-makers may use it, may be enhanced by the availability of a generically applicable computational method that can uncover the exact number of all feasible solutions, along with their constituent options, no matter the size of the problem as characterized by its decision areas. In pursuing this position, what follows is based within the tradition in which AIDA was originally presented by Harary et al. (1965), that is, the field of graph theory.

## 3 Terms of Reference

A graph consists of "vertices" and undirected "edges". The "order" of a graph is its number vertices, and its "size" is its number of edges (Chartrand (1977: 11). Individual edges join pairs of vertices according to some criterion. Two vertices so joined are said to be "adjacent" to each other. Vertices may be labeled, in which case the graph is termed a "labeled" graph.

Following Brualdi (2018: 27), if we consider the vertices of a graph as a set, $S$, then a "partition" of $S$ is a collection $S_1$, $S_2$,…,$S_m$ of subsets of $S$ such that each vertex of $S$ is in exactly one of those subsets, and:

$$S = S_1 \cup S_2 \cup \ldots \cup S_m,$$

$$S_i \cap S_j = \emptyset,\ (i \neq j).$$

Thus, the subsets $S_1$, $S_2$,…,$S_m$ are pairwise disjoint subsets, and their union is $S$. The subsets $S_1$, $S_2$,…,$S_m$ are called the "parts" of the partition. Parts are enumerated, such as part 1, part 2, and so on. A partition, $P$, constituted by $n$ parts, therefore, may be said to be of order $n$. The partitioning of vertices in this manner is an understanding adopted by network theory (Batagelj et al. 2014: 10) and will prove useful in what follows.

A graph of labeled vertices (options), partitioned into labeled parts (decision areas), and joined by edges to indicate pairwise incompatibilities between them, has been called an "option graph" by Friend and Hickling (2005: 37). The top left of Fig. 1 provides an example: the option graph, $G$, consists of four decision areas, differentiated by color, three of which have three options and one of which has two options.[3]

Figure 1 also shows the graph $\bar{G}$, which has the same set of vertices as the option graph $G$. The difference is that, in $\bar{G}$, two vertices are adjacent if and only if they are not adjacent in $G$. $\bar{G}$ is a type of opposite graph known as the "complement" of $G$:

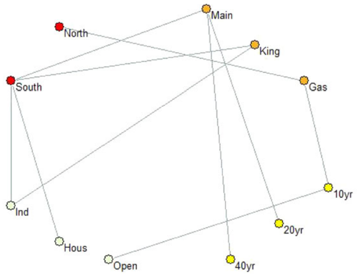
[3] Since the present paper is concerned solely with the computational problem, all designations in the example of Fig. 1 serve merely as convenient labels. For contextual information concerning the data shown in Fig. 1, readers may follow the indicated reference.

Decision Areas legend:

● DIST LIFE? ● SHOP LOC'N? ● ROAD LINE? ○ CENT'L SITE?

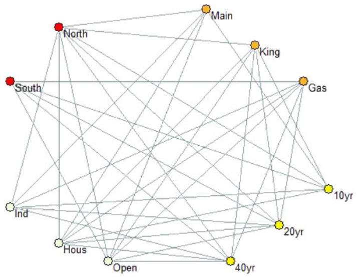
## $G$: Option Graph

Links indicate incompatibilities



## $\bar{G}$: Complement of Option Graph

Links indicate compatibilities



## Compatibility Grid of $G$

| Decision Areas | | DIST LIFE? | | | SHOP LOC'N? | | | ROAD LINE? | | CENT'L SITE? | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Options** | 10yr | 20yr | 40yr | Main | King | Gas | North | South | Ind | Hous | Open |
| DIST LIFE? | 10yr | | | | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| | 20yr | | | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 40yr | | | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SHOP LOC'N? | Main | | | | | | | 0 | 1 | 0 | 0 | 0 |
| | King | | | | | | | 0 | 1 | 1 | 0 | 0 |
| | Gas | | | | | | | 1 | 0 | 0 | 0 | 0 |
| ROAD LINE? | North | | | | | | | | | 0 | 0 | 0 |
| | South | | | | | | | | | 1 | 1 | 0 |
| CENT'L SITE? | Ind | | | | | | | | | | | |
| | Hous | | | | | | | | | | | |
| | Open | | | | | | | | | | | |

## Compatibility Grid of $\bar{G}$

| Decision Areas | | DIST LIFE? | | | SHOP LOC'N? | | | ROAD LINE? | | CENT'L SITE? | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Options** | 10yr | 20yr | 40yr | Main | King | Gas | North | South | Ind | Hous | Open |
| DIST LIFE? | 10yr | | | | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| | 20yr | | | | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 40yr | | | | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| SHOP LOC'N? | Main | | | | | | | 1 | 0 | 1 | 1 | 1 |
| | King | | | | | | | 1 | 0 | 0 | 1 | 1 |
| | Gas | | | | | | | 0 | 1 | 1 | 1 | 1 |
| ROAD LINE? | North | | | | | | | | | 1 | 1 | 1 |
| | South | | | | | | | | | 0 | 0 | 1 |
| CENT'L SITE? | Ind | | | | | | | | | | | |
| | Hous | | | | | | | | | | | |
| | Open | | | | | | | | | | | |

The entries of 1 in each compatibility grid reflect the links evident in its associated graph. The sum of the entries in a compatibility grid is, therefore, equal to the number of links evident in its associated graph. The diagonal entries in blue, are empty, since no option can be in/compatible with itself or, due to mutual exclusivity, with other options within its decision area. The lower diagonal area, in grey, is conventionally left empty for its entries would only be a reflection of those of the upper diagonal area. Note that registering the in/compatibilities in the grey lower diagonal area, and filling the blue diagonal area with zeroes, yields the adjacency matrix of the respective graph.

## Adjacency Matrix, $A$, of $G$

| ***A*** | 10yr | 20yr | 40yr | Main | King | Gas | North | South | Ind | Hous | Open |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 10yr | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 20yr | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 40yr | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Main | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| King | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Gas | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| North | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| South | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| Ind | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| Hous | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Open | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## Adjacency Matrix, $\bar{A}$, of $\bar{G}$

| $\bar{A}$ | 10yr | 20yr | 40yr | Main | King | Gas | North | South | Ind | Hous | Open |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 10yr | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 20yr | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 40yr | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Main | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| King | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| Gas | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| North | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| South | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| Ind | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Hous | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Open | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

$G$ and $\bar{G}$ may be represented by respective square adjacency matrices, $A$ and $\bar{A}$. In each case, the index of rows and columns is denoted by the options. The entries of 1 reflect the links evident in the associated graph.

$\bar{A}$ is found by the following formula (Hage and Harary, 1983: 117): $\bar{A} = (J\text{-}I)\text{-}A$, where $J$ is known as a "universal" matrix, and is of the same size as $A$ but with all entries being 1; and $I$, known as the "identity" matrix, is of the same size as $A$, but with entries along the diagonal being 1 and all others zero.

The diagonal entries in the adjacency matrices are necessarily zero since an option cannot be in/compatible with itself. Furthermore, due to the mutual exclusivity of options within decision areas, all corresponding entries must also be set to zero. The blue shading reflects both these issues.

In the adjacency matrices, the color shadings match those of the compatibility grids for ease of comparison.

**Fig. 1** Graphs, compatibility grids, and adjacency matrices using data from an example by Friend and Hickling (2005: 34, 36)

the lines that are present in the complement $\bar{G}$ are precisely those that are absent in the original option graph $G$ (Harary 1969: 15). It follows that the complement $\bar{G}$ of an option graph $G$ is that graph where labeled, partitioned options are joined by one edge to indicate pairwise compatibilities (as opposed to incompatibilities) between them.

Figure 1 also shows how both, the option graph $G$ and its complement $\bar{G}$, may be represented by compatibility grids[4] (Friend 2001: 128) and, especially useful for what follows, by adjacency matrices (Harary 1969: 150), $A$ and $\bar{A}$ respectively.

## 4 Task Delineation

The task of AIDA is one of finding all feasible solutions that conform to the following conditions:

1. The number of options in each feasible solution is equal to the number of decision areas;
2. The options constituting each feasible solution respectively belong to different decision areas; and,
3. Options linked in the option graph (i.e. incompatible options) do not appear together in any feasible solution.

In other words, feasible solutions are constructed by iteratively cycling through the decision areas, taken in some consecutive sequence, from each of which one option is chosen, such that the options constituting any one feasible solution are related in the complement $\bar{G}$ of the option graph $G$, and not related in the option graph $G$ itself. Consequently, the relations between the options constituting any one feasible solution form a cycle, rendering feasible solutions themselves structurally cyclical. This understanding is relevant when moving toward a graph theoretical statement of the task at hand.

In graph theoretical terms, given any $\bar{G}$ with labeled vertices exclusively distributed across a partition, $P(\bar{G})$, and linked according to compatibility, the task is to find all labeled cyclical subgraphs of $\bar{G}$ that conform to the following conditions:

1. Each subgraph extracted from $\bar{G}$ is of order $P(\bar{G})$;
2. The vertices constituting each subgraph respectively belong to different parts of $P(\bar{G})$; and,
3. Those vertices that are linked in $G$ do not appear together in any of the subgraphs extracted from $\bar{G}$.

In graph theory, the task falls under the remit of the subfield known as "subgraph isomorphism", recent reviews of which are provided by Conte et al. (2004), Foggia et al. (2014), and Vento (2015). Its application to "mining graph data" is extensively discussed in the eponymous book edited by Cook and Holder (2007). An example of a field that makes extensive empirical use of subgraph isomorphism is Chemistry, in its search for molecular substructures (Bonnici et al. 2013; Randic 1978; Randic and Wilkins 1979; Barnard 1993; Willett et al. 1998).

[4] Also known as "compatibility matrix" (Friend and Hickling 2005: 35; Hickling 1978: 472), "incompatibility matrix" (Blandford and Hope 1985: 209), and "interaction matrix" (Jones 1970/1992: 311–312).

Subgraph isomorphism is concerned with searching a graph for subgraphs that conform to a given subgraph sample. Network theory (Batagelj et al. 2014: 53, 94–95) provides terms which will prove useful in what follows: the given subgraph sample is known as a “fragment”, $F$, and the problem is solved through what is known as “fragment searching”, whereby a graph is searched for all instances that correspond to a given sample subgraph/fragment $F$. Constraints may be stipulated for $F$, such as those described by conditions (1) and (2), above. For the present case, therefore, $F$ is a labeled cyclical graph, whose order, as well as partition, *P(F),* is equal to the number of parts of $P(\bar{G})$, and whose vertices are partitioned in such a manner that each vertex is assigned to one and only one part of *P(F)*. AIDA, however, also requires condition (3), above; that is, any vertices linked in the original option graph, $G$, may not appear together in any of the subgraphs extracted from its complement $\bar{G}$. Consequently, AIDA requires a very particular type of fragment searching, one that refers simultaneously to the option graph $G$, and to its complement $\bar{G}$.

The links in an option graph $G$ denote incompatibilities between options. By contrast, the links in its complement, $\bar{G}$, denote compatibilities between options. Given the same set of vertices, therefore, two types of relations may occur, each being the opposite of the other. Such simultaneity implies the availability of a graph where, both, incompatible and compatible links may be considered. A graph that depicts vertices between which, both, a relation and its opposite may occur, is called a “signed graph” (Cartwright and Harary 1956: 282ff; Harary 1957; Hage and Harary 1983: 40–64). For the case in question here, a signed graph, $U$, is the union of relations depicted in the option graph, $G$, and its complement $\bar{G}$, for the vertex set they both share. This is shown in Fig. 2, along with its adjacency matrix, $S$. The availability of a signed graph, that unites the relations depicted in an option graph, $G$, and in its complement $\bar{G}$, enables fragment searching to be done under the aforementioned three conditions required by AIDA.

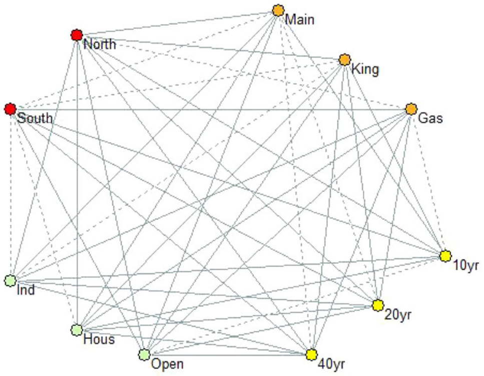
In brief, in order to find the feasible solutions, and their constituent options, the required inputs are:

- A labeled signed graph, $U$, composed of the vertices (options) shared by the labeled graphs, $G$ (option graph) and $\bar{G}$ (its complement), and uniting the incompatible and compatible relations depicted therein (in graph theory, such relations are respectively termed “negative” and “positive”);
- A partition $P(\bar{G})$ [note: this equals $P(G)$] that assigns the vertices (options) of $U$ to parts (decision areas), such that a vertex (option) is assigned to only one part (decision area), and a part (decision area) may have more than one vertex (option) assigned to it;
- A labeled fragment, $F$, whose order is equal to the number of parts of $P(\bar{G})$, and whose structure is cyclical; and,
- A partition, *P(F)*, that assigns each vertex of $F$ to one part, and each part can have only one vertex.

The details in Figs. 1 and 2 show that these inputs are directly computable from only one fundamental data structure of AIDA: the option graph $G$. It will be noted, however, that the derivations are interchangeable between $G$, $\bar{G}$, and $U$. Indeed, Harary et al. (1965) introduced their own procedure using $\bar{G}$ as an initial input. The presentation

*U*: Signed Graph

Dashed/Solid links indicate incompatibilities/compatibilities



*U* is the union of relations depicted in the graph, *G*, and its complement $\bar{G}$, for the vertex set they both share.

Adjacency Matrix, *S*, of *U*

| ***S*** | 10yr | 20yr | 40yr | Main | King | Gas | North | South | Ind | Hous | Open |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 10yr | 0 | 0 | 0 | 1 | 1 | -1 | 1 | 1 | 1 | 1 | -1 |
| 20yr | 0 | 0 | 0 | -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 40yr | 0 | 0 | 0 | -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Main | 1 | -1 | -1 | 0 | 0 | 0 | 1 | -1 | 1 | 1 | 1 |
| King | 1 | 1 | 1 | 0 | 0 | 0 | 1 | -1 | -1 | 1 | 1 |
| Gas | -1 | 1 | 1 | 0 | 0 | 0 | -1 | 1 | 1 | 1 | 1 |
| North | 1 | 1 | 1 | 1 | 1 | -1 | 0 | 0 | 1 | 1 | 1 |
| South | 1 | 1 | 1 | -1 | -1 | 1 | 0 | 0 | -1 | -1 | 1 |
| Ind | 1 | 1 | 1 | 1 | -1 | 1 | 1 | -1 | 0 | 0 | 0 |
| Hous | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | 0 | 0 | 0 |
| Open | -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

*S* is found from the adjacency matrices *A* and $\bar{A}$, as follows: $S = -1A + \bar{A}$.

**Fig. 2** Signed graph with adjacency matrix, using data from Fig. 1

here has focused on *G* as the initial input only because of claims, based on experience in using the Strategic Choice Approach, that the provision of *G* is easier than that of $\bar{G}$ (e.g. Luckman 1967: p. 350; Friend and Hickling 2005: 35–37). Overall, therefore, there is flexibility in the choice of the fundamental data structure that may initially be used, including, if available at the outset, the signed graph *U* which is central to the solution procedure to be described. What follows is a description of the subgraph isomorphism algorithmic procedure that manipulates the inputs in order to tackle the questions stipulated in the introductory section of the paper.

## 5 Procedural Description

Underlying the majority of procedures to subgraph isomorphism is backtrack programming (Krissinel and Henrick 2004), widely used in constraint satisfaction problems

(Larrosa and Valiente 2002; Tsang 1993), and classical statements of which are provided by Golomb and Baumert (1965), Wells (1971: 93–126), Tarjan (1972), and Fillmore and Williamson (1974).

Backtrack programming, or "backtracking", is commonly associated with depth-first search procedures. A depth-first algorithm proceeds deep into a graph in search of subgraph isomorphisms. When a vertex is reached from where the required subgraph structure cannot be found, the algorithm backtracks along the visited edges until it finds a vertex from which an alternative extension of edges may be explored. Due to its recursive nature, the depth-first procedure is prone to stack overflow problems, and is therefore mainly applicable to smaller graphs (where "small" is relative to temporary memory computer space) (Vento 2015: 293).

Eppstein (1999: 11–21; also see Aho et al. 1974: 172–223) shows that the stack overflow problem is alleviated by making depth-first searches conditional upon breadth-first procedures, whereby the depth search concentrates on one level at any one time. Knuth (2015: 129–130) notes that Dodgson [better known as Lewis Carroll (Cohen 1995)] was a pioneer in this conditional procedure, but his unpublished manuscript on the matter—which, Knuth notes, "would have completely changed the history of mechanical theorem proving if it had come to light earlier"—was found only in 1977, 79 years after his death. Given the advantages of conditioning depth-first searches on breadth-first procedures, and broadly following conventions set by Knuth (1997: 2–9), Fig. 3 provides the pseudocode for such an algorithm applicable to the present issue of interest.

Such a code is programmed into the freely-downloadable, award-winning, graph/network software known as "Pajek" (version 5.02 or higher), described by de Nooy et al. (2018). Pajek handles graphs and digraphs of up to a billion vertices, and runs on, both, Windows and macOS operating systems. Thus, contrary to STRAD, computer compatibility and computational maximum issues are alleviated. Furthermore, Pajek requires very simple coding in order to read inputs, and provides easily interpretable outputs and high-quality diagrams. In what follows, the computational procedure is illustrated based on the data of Fig. 1. It draws extensively on Pajek, and provides information on the sequence of Pajek commands that will yield the required answers.

## 6 Proof of Concept

Figure 4 illustrates the procedure from the user's standpoint, based on the example in Fig. 1, using standard Pajek coding for the inputs.

The Pajek menu instructions are also shown, leading to the generation of three outputs:

1. A graph containing all the possible feasible solutions;
2. A partition that enumerates the number of feasible solutions to which each option belongs; and,

**Algorithm AIDA** (*Searching for subgraphs in a partitioned signed graph*). Given any partitioned signed graph, *U*, find all cyclical subgraphs which correspond to a given partitioned cyclical fragment, *F*.

Terms of reference:

- *U*: Any signed graph that unites the relations depicted in *G* and for the vertex set they both share.
- *P(U)*: A partition of the vertices of *U*, such that each vertex is assigned to only one of the parts, a part may have more than one vertex, and no vertices within a part are related. The partition is equally applicable to *G* and $\bar{G}$.
- *F*: A cyclical graph whose order is equal to the number of parts of *P(U)*
- *P(F)*: A partition, of the same order and part names of *P(U)*, of the vertices of *F*, such that each vertex is assigned to only one of the parts, and a part may not have more than one vertex
- *Negative edge*: an edge belonging to *G*
- *Positive edge*: an edge belonging to $\bar{G}$

Steps of the algorithm:

**S1.** Provide *U*, *P(U)*, *F*, and *P(F)*.

**S2.** [Generate an ordered list of all edges in *F*]. Sort all edges in *F* according to a breadth-first order. (Sorting the edges of *F* in breadth-first order ensures that the next edge selected from the list will always reach at most one vertex that has so far not been visited. For example, for a cyclical fragment with vertices 1-2-3-4-1, if edge 1-2 is initially selected, the next edge to be selected must be attached to either vertex 1 or vertex 2, that is, edge 1-4 or edge 2-3; edge 3-4 will not be selected.)

**S3.** [Find a matching in *U* of one edge of *F*.] From the ordered list of the edges of *F*, select the first edge and corresponding incident vertices. The vertices are partitioned according to *P(F)*. Find a matching of this edge in *U*, whose corresponding incident vertices are partitioned in *P(U)* in accordance with *P(F)*. (At this point, any edge in *U* can be selected. From now on, "matching" means that all vertices visited so far in *U* have the same neighborhood structure as vertices so far visited in *F*.)

**S4.** [Find a matching in *U* of the next edge of *F*.]. Choose the next edge from the ordered list of the edges of *F* and corresponding incident vertices. Find a matching of this edge in *U*, whose corresponding incident vertices are partitioned in *P(U)* in accordance with *P(F)*.

**S5.** [Has a match been found?] Check that no vertices selected thus far from *U* are adjacent by negative edges. If such adjacency appears, discard the chosen edge and go back to S4.

**S6.** [Has a cyclical subgraph been found in *U*?]

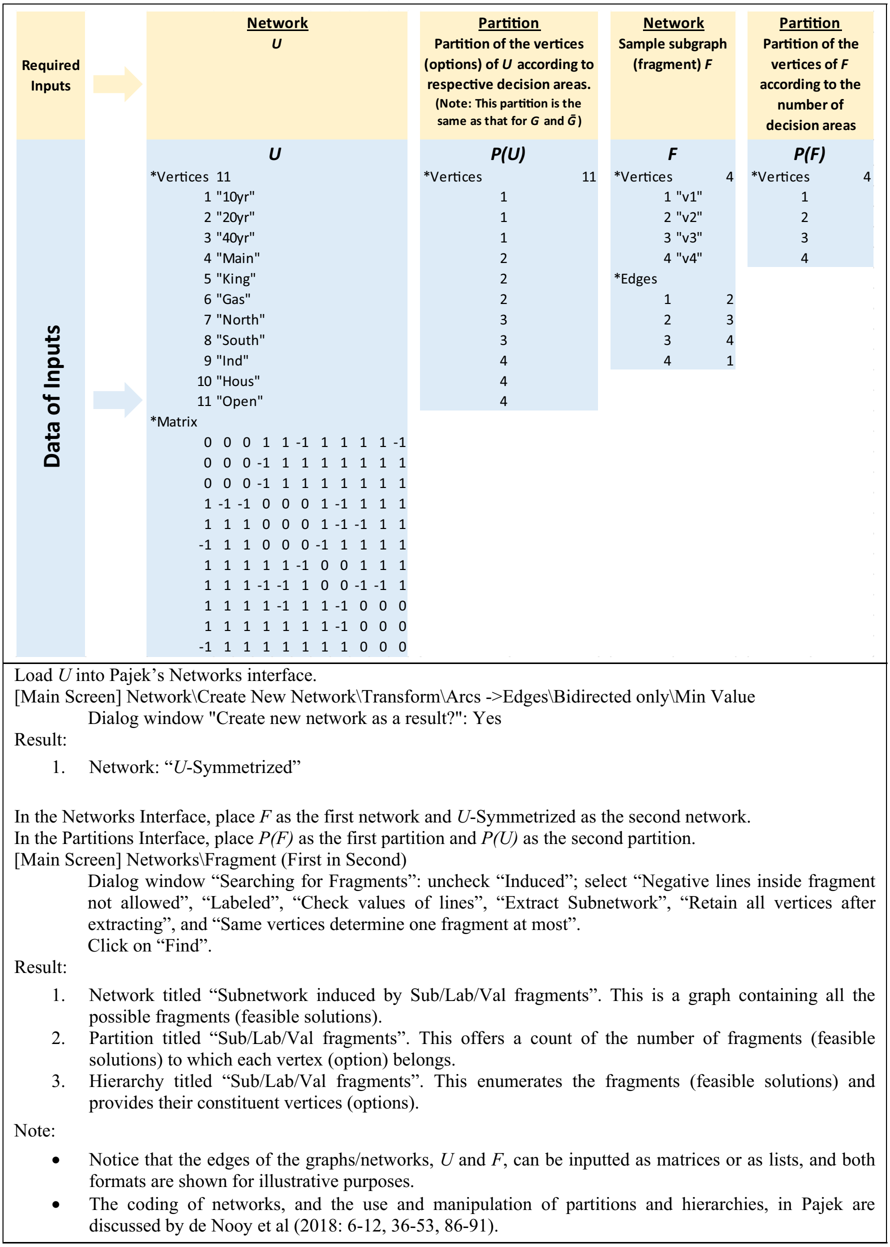
- If no, repeat steps S4 and S5. If no cyclical subgraph is found, the algorithm terminates; the signed graph *U* does not contain a subgraph that matches the properties of *F*.
- If yes, repeat steps S3 through S5 until the whole graph has been searched. (Note that many matchings are possible up to $n$ – 2 edges. Edge $n$ – 1, however, constrains the possible matchings because it transforms them from a path to a cycle.) ▮

**Fig. 3** Pseudocode for AIDA algorithm

3. A Pajek data object known as a "hierarchy" (de Nooy et al. 2018: 86–91) that enumerates the feasible solutions themselves, and, for each one, lists the constituent options.

For the example in Fig. 1, these three outputs are shown on the left in Fig. 5. Note that the Hierarchy provides an initial view listing the number of feasible solutions with their number of options in brackets (bottom left of Fig. 5); and a detailed view, available by expanding any one feasible solution from the initial list, that provides a listing of the constituent options for each feasible solution (on the right in Fig. 5).

For the example in question, Pajek's hierarchy lists 9 feasible solutions, each constituted by the options shown. The hierarchy data may be conveniently combined in a diagram. This is illustrated in Fig. 6, and the required Pajek menu instructions are also provided.



| Required Inputs | **Network**<br>*U* | **Partition**<br>Partition of the vertices (options) of *U* according to respective decision areas.<br>(Note: This partition is the same as that for *G* and *Ḡ*) | **Network**<br>Sample subgraph (fragment) *F* | **Partition**<br>Partition of the vertices of *F* according to the number of decision areas |
|---|---|---|---|---|

Data of Inputs

*U*

```
*Vertices 11
        1 "10yr"
        2 "20yr"
        3 "40yr"
        4 "Main"
        5 "King"
        6 "Gas"
        7 "North"
        8 "South"
        9 "Ind"
       10 "Hous"
       11 "Open"
*Matrix
         0  0  0  1  1 -1  1  1  1  1 -1
         0  0  0 -1  1  1  1  1  1  1  1
         0  0  0 -1  1  1  1  1  1  1  1
         1 -1 -1  0  0  0  1 -1  1  1  1
         1  1  1  0  0  0  1 -1 -1  1  1
        -1  1  1  0  0  0 -1  1  1  1  1
         1  1  1  1  1 -1  0  0  1  1  1
         1  1  1 -1 -1  1  0  0 -1 -1  1
         1  1  1  1 -1  1  1 -1  0  0  0
         1  1  1  1  1  1  1 -1  0  0  0
        -1  1  1  1  1  1  1  1  0  0  0
```

*P(U)*

```
*Vertices          11
             1
             1
             1
             2
             2
             2
             3
             3
             4
             4
             4
```

*F*

```
*Vertices      4
         1 "v1"
         2 "v2"
         3 "v3"
         4 "v4"
*Edges
         1      2
         2      3
         3      4
         4      1
```

*P(F)*

```
*Vertices      4
         1
         2
         3
         4
```

Load *U* into Pajek's Networks interface.
[Main Screen] Network\Create New Network\Transform\Arcs ->Edges\Bidirected only\Min Value
Dialog window "Create new network as a result?": Yes

Result:

1. Network: "*U*-Symmetrized"

In the Networks Interface, place *F* as the first network and *U*-Symmetrized as the second network.
In the Partitions Interface, place *P(F)* as the first partition and *P(U)* as the second partition.
[Main Screen] Networks\Fragment (First in Second)

Dialog window "Searching for Fragments": uncheck "Induced"; select "Negative lines inside fragment not allowed", "Labeled", "Check values of lines", "Extract Subnetwork", "Retain all vertices after extracting", and "Same vertices determine one fragment at most".
Click on "Find".

Result:

1. Network titled "Subnetwork induced by Sub/Lab/Val fragments". This is a graph containing all the possible fragments (feasible solutions).
2. Partition titled "Sub/Lab/Val fragments". This offers a count of the number of fragments (feasible solutions) to which each vertex (option) belongs.
3. Hierarchy titled "Sub/Lab/Val fragments". This enumerates the fragments (feasible solutions) and provides their constituent vertices (options).

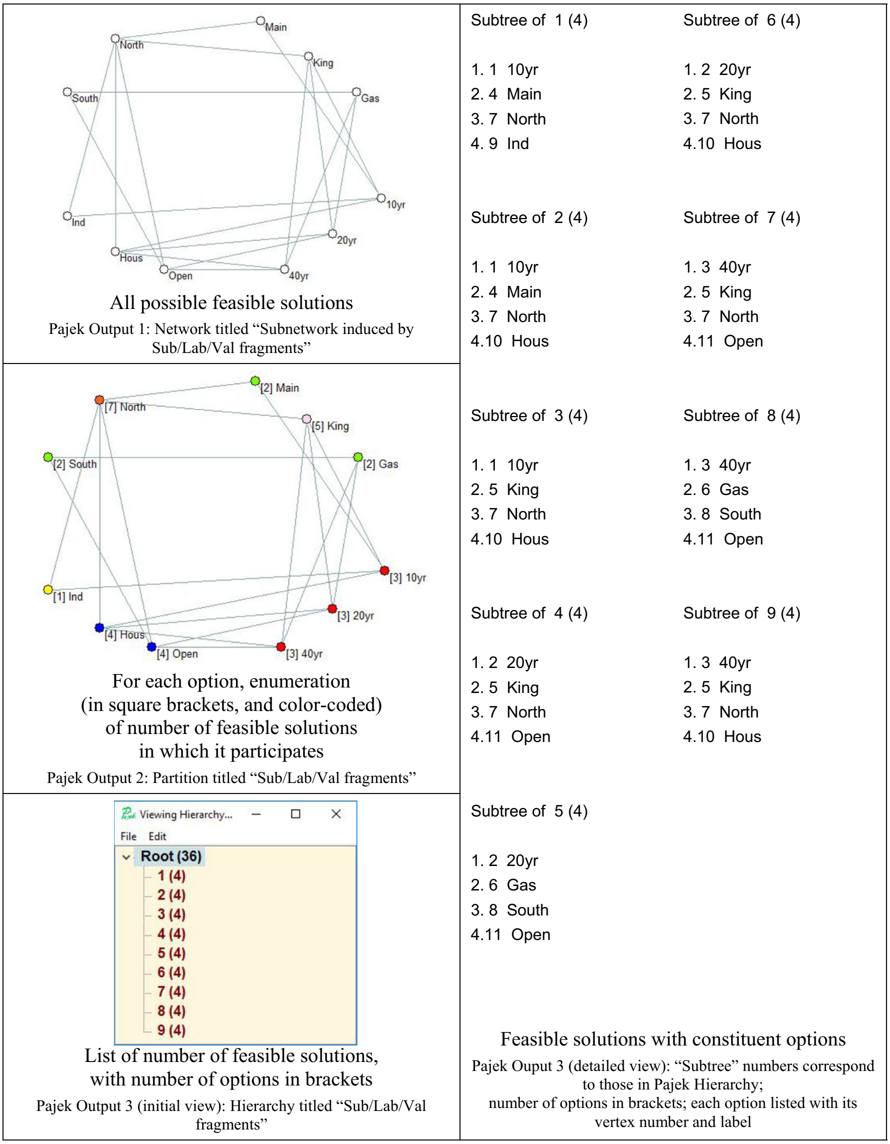
Note:

- Notice that the edges of the graphs/networks, *U* and *F*, can be inputted as matrices or as lists, and both formats are shown for illustrative purposes.
- The coding of networks, and the use and manipulation of partitions and hierarchies, in Pajek are discussed by de Nooy et al (2018: 6-12, 36-53, 86-91).

**Fig. 4** Sample Pajek application, using data from Fig. 1

The analytical advantages of access to lists, such as that on the right of Fig. 5, along with diagrams and summaries, such as those of Fig. 6, may be exemplified as follows:

- Option "Ind" occurs in only one combination. Should this option be deemed important, under some given criteria and context, then the results show that this option is



All possible feasible solutions

Pajek Output 1: Network titled "Subnetwork induced by Sub/Lab/Val fragments"

For each option, enumeration
(in square brackets, and color-coded)
of number of feasible solutions
in which it participates

Pajek Output 2: Partition titled "Sub/Lab/Val fragments"

```
Root (36)
    1 (4)
    2 (4)
    3 (4)
    4 (4)
    5 (4)
    6 (4)
    7 (4)
    8 (4)
    9 (4)
```

List of number of feasible solutions,
with number of options in brackets

Pajek Output 3 (initial view): Hierarchy titled "Sub/Lab/Val fragments"

```
Subtree of  1 (4)

1. 1  10yr
2. 4  Main
3. 7  North
4. 9  Ind

Subtree of  2 (4)

1. 1  10yr
2. 4  Main
3. 7  North
4.10  Hous

Subtree of  3 (4)

1. 1  10yr
2. 5  King
3. 7  North
4.10  Hous

Subtree of  4 (4)

1. 2  20yr
2. 5  King
3. 7  North
4.11  Open

Subtree of  5 (4)

1. 2  20yr
2. 6  Gas
3. 8  South
4.11  Open

Subtree of  6 (4)

1. 2  20yr
2. 5  King
3. 7  North
4.10  Hous

Subtree of  7 (4)

1. 3  40yr
2. 5  King
3. 7  North
4.11  Open

Subtree of  8 (4)

1. 3  40yr
2. 6  Gas
3. 8  South
4.11  Open

Subtree of  9 (4)

1. 3  40yr
2. 5  King
3. 7  North
4.10  Hous
```

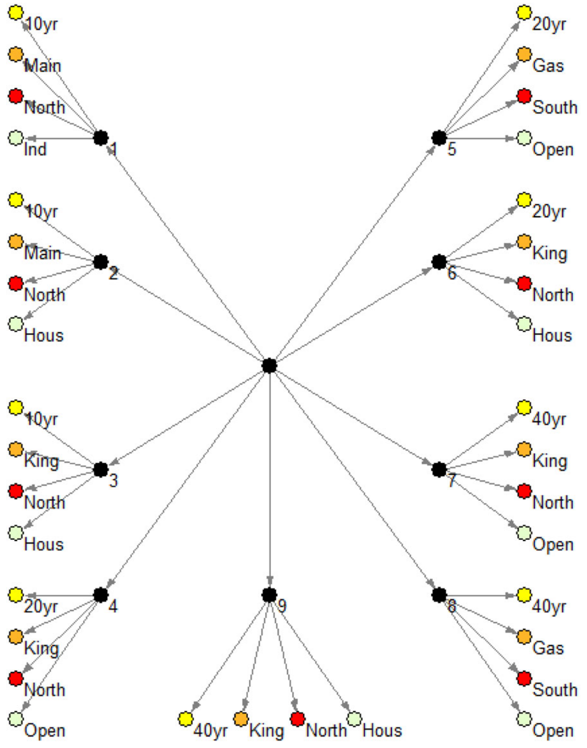
Feasible solutions with constituent options

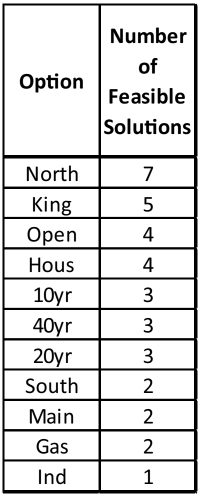
Pajek Ouput 3 (detailed view): "Subtree" numbers correspond to those in Pajek Hierarchy;
number of options in brackets; each option listed with its vertex number and label

**Fig. 5** Pajek outputs, using data from Fig. 1

highly constrained: for instance, it can only be chosen in conjunction with "10 yr," "Main," and "North." Had any one of the decision areas, to which these latter three options belong, been left out in favor of a reduced problem focus, any consideration of the option "Ind" would risk erroneous conclusions and misspent decision-making time.

- From the "Road Line" decision area, option "South" is more constrained compared to option "North". The former option not only participates in only two feasible





| Option | Number of Feasible Solutions |
|---|---|
| North | 7 |
| King | 5 |
| Open | 4 |
| Hous | 4 |
| 10yr | 3 |
| 40yr | 3 |
| 20yr | 3 |
| South | 2 |
| Main | 2 |
| Gas | 2 |
| Ind | 1 |

Frequency table showing the number of feasible solutions in which each option participates.

To obtain a diagram of the hierarchy:

[Main Screen] Hierarchy\Make Network: Yes, Yes

Result:

1. Hierarchy Network

[Main Screen] Network\Create Partition\Degree\All

Result:

1. Degree partition of Hierarchy Network

[Main Screen] Draw\Network + First Partition

The diagram appears in the Draw Screen. In this illustration, a partition has also been used to allocate decision area colors to the vertices, according to the legend given in Figure 1, thus formatting for aesthetic and analytical value (de Nooy et al., 2018: pp. 17-24).

Using the partition titled "Sub/Lab/Val fragments", available from the procedure in Figure 4, the following Pajek command provides a frequency table (de Nooy et al., 2018: 39-41):

[Main Screen] Partition\Info, and in the dialog window enter 1, followed by the number of vertices (in this case, 11).

The table shown above has formatted the Pajek frequency results for presentational purposes.

**Fig. 6** Diagram showing the number of feasible solutions, as well as their constituent options, using data from Fig. 1

solutions; it can only occur in these feasible solutions if options "Gas" and "Open" are adopted. Option "North," meanwhile, participates in seven feasible solutions, and its only major constraint is its incompatibility with "Gas." The comparative flexibility of adopting one or the other of the options, from the same decision area, is rendered explicit by allowing a consideration of all the decision areas and options.

The above serve to illustrate how the procedure enables insights that would not have been available had a reduced problem focus been initially chosen. On the one hand, decision-makers have a map of the problematic situation as whole, one that stimulates informed judgments when considering which subset of feasible solutions is amenable to further analysis based on contextually-relevant criteria. On the other hand, decision-makers can gauge the role and flexibility available to any one option. Therefore, in providing all feasible solutions, in lists and diagrams, the generically applicable computation method described here empowers decision-makers to make informed judgments on the macro and micro scale. In the words of Friend (2014: 31) cited earlier, the procedure helps to trace the impacts of different, any, and/or all programmatic influences.

## 7 Discussion and Conclusion

The essence of the Analysis of Interconnected Decision Areas (AIDA) is a search for the complete set of combinations of elements under the following structural conditions: an element must belong to only one group; one group may contain multiple elements; elements within a group are mutually exclusive; and, elements across groups may be connected due to incompatibilities or compatibilities. The elements may be multiple types of entities (e.g. decisions, individuals, institutions, machines, etc.), and, in terms of quality and quantity, may be distributed differently between and within each group. The point is that the groups are each seen as relevant in some context, and the combination of their elements, one from each group, must be uncovered given the constraints of group clustering, mutual exclusivity of all elements within any one group, and incompatibilities between certain elements across groups. AIDA, therefore, addresses the non-trivial systemic challenge of finding combinations of related elements constrained by group clustering, intra-group mutual exclusivity, and inter-group (in)compatibilities.

This paper has provided the means through which the complete set of combinations may be uncovered. In doing so, it has emphasized the crucial role played by network modeling, the computational facility afforded by the innovative introduction to the analysis of a signed graph, and the consequent flexibility of initial inputs that this affords. The procedure described offers a solution approach that is generically applicable across contexts, and one that is unlimited in terms of scale. The procedure has been tested with data provided by Harary et al. (1965), Kammeier (1998), Weas and Campbell (2004), and Friend and Hickling (2005: pp. 34, 36). In all cases, the results conform to the respective scholars' claimed (but heretofore unconfirmed) findings.

For the practitioner faced with a situation whose structure conforms to the aforementioned characteristics, the procedure in this paper has alleviated the problem of computational and presentational tractability. Through the provision of lists as well as diagrams of the complete set of combinations, decision-makers can make informed judgments should they want to tackle the entire problem or some proper subset thereof. For the former case, decision-makers can work with complete information of the network structure of the situation, and, as illustrated, can even gauge the role played by, and the flexibility available to, any one option in the entire network. Such assessments can also be done when working on some subset of a problem, but the initial choice—and, if desired, comparison—between competing subsets can now be undertaken in full view of all feasible solutions and their constituent options. At the very least, decision-makers can appreciate, either in detail or in proportional terms, their intellectual grasp and ownership of a chosen subset formulation relative to the whole network problem. Furthermore, the macro and micro insights afforded by the procedure assist practitioners when deciding on the constitution of whatever problem focus they might choose: piecemeal or wholesale changes to the number of decision areas to be considered, along with their interrelations, can now be analyzed in full view of their contextual relevance. In brief, in alleviating computational and presentational tractability, greater material and procedural choice is made available to decision-makers.

In focusing on this provision, the paper has contributed to answering Friend's (2014) call to further enhance the Strategic Choice Approach (SCA) and, in particular, some of the tools that were developed during his time at the Institute of Operational Research. By responding to Friend's encouragement, the paper has not only provided the means for resolving the computational challenge at the heart of AIDA, which itself lies at the heart of SCA, but has also demonstrated the utility of linking the SCA software, STRAD, with network software. This, moreover, points to wider possibilities: the utility of network science for the family of problem structuring methods (PSMs) (Rosenhead and Mingers 2001) to which SCA belongs. For if there is one thing that many PSMs have in common, it is that they construct, and rely upon, network diagrams. The PSM literature, however, offers its practitioners little or no network analytical power.

For example, Soft Systems Methodology (SSM) can produce large conceptual systems (e.g. Wilson 2001), but no analyses are offered to identify the activities that sustain the internal coherence of such systems and, without which, any one such system disconnects into two or more independent subsystems. A graph theorist or network scientist would draw on the analysis of cut vertices (Aldous and Wilson 2000: 222), or the specific definitions of, and multiple analytical procedures for finding, brokerage that have emerged in the literature (de Nooy et al. 2018: 170–196).

As another example, consider the Decision Explorer® software that complements Strategic Options Development and Analysis (SODA). This software includes 41 analyses that help users navigate cognitive maps designed as directed graphs/networks (Banxia 2005: 156–158). Two of these analyses concern centrality, that is, the central influence a construct may enjoy in a map. On the one hand, the software allows centrality to be calculated according to immediate degree (the number of links around a construct, separable into indegree and outdegree); on the other, according to the distances of all ancestors and descendants of a construct across the entire map. There exist, however, at least 108 centrality measures for network models, all of them mathematically defined and open to translation into computer programs (Schoch 2015: 12; also see Schoch and Brandes 2016). Incorporating these and other analyses would serve to add to the power of SODA.

A means through which such an incorporation can happen has been opened by the introduction of the network software Pajek in this paper. This software is designed specifically as a network calculator that can handle billions of vertices, and their relations, irrespective of context. It is, therefore, useful for both, abstract and empirical analyses. Conforming to Ackoff's (1967: B153) maxim—that "no [software] should ever be [used] unless the [user] for whom it is intended [is] trained to evaluate and hence control it rather than be controlled by it"—Pajek requires the user to structure an analysis in a manner analogical to the mathematical operations on networks. This affords precise operational oversight with consequent demystification of the black box. Furthermore, Pajek has a long history of published algorithms which are open to evaluation (Batagelj 1991, 2003; Batagelj and Mrvar 1998, 2008, 2014; Batagelj and Zaversnik 2011; Batagelj et al. 2014; Batagelj and Cerinšek 2013). This, along with its user-friendly manageable operations, enables users to maintain control of their use of the software instead of being controlled by it. In addition, the software provides high resolution graphics of networks, with multiple means for manipulating their aesthetic presentation, thus allowing for sophisticated visual appreciation to complement analytical results. Finally, Pajek is constantly updating its interface capabilities with other software.

This last point indicates at least one desirable opportunity for PSM practitioners: since Pajek, STRAD and Decision Explorer® are programmed to work with networks, a respective computational interface, between the former and the latter two, would enhance the latter far beyond their current capabilities, thus offering decision-makers powerful analytical means for navigating the structural complexity of the situations in which they are used. Furthermore, interface advantages accrue both ways, for the analyses undertaken in this paper stimulated the design and inclusion of new procedures in Pajek itself. All indications, therefore, point to a fruitful dialog between the PSM and network science fields. The relevance of a such a dialog lies in the common aim between the two fields: the resolution of complex, organizational, socially-infused problems. The potential for a powerful partnership lies in the fact that each bring respectively different strengths. Network science is strong on the *structural* analysis of complexity. The forte of the PSM field lies in its sophisticated conceptualization and analysis of process, what may be termed *procedural* complexity. Both are required for supporting and facilitating interactions between decision-makers.

Finally, there seem to be at least two outstanding research opportunities concerning algorithmic design. Smith and Morrow (1999: 257) note that the option graph "suggests that pairs of solutions may be incompatible, but in reality it may be triples of solutions that are incompatible, although any pair within the triple would be satisfactory." This idea is echoed by Jones (1970/1992: 315) who, referring to Luckman (1967), notes that AIDA could be "extended to take account of incompatibilities of sub-solutions taken in threes, fours, etc."; he adds that "it would be useful if AIDA could be applied in the case of conditional [probabilistic] incompatibilities." Two paths for future research on computational processes in AIDA are thereby identified. First, as it stands, AIDA considers pairwise incompatibilities when computing the number of feasible solutions. The question raised is whether this dyadic approach might not be sufficient in some cases. Conceptually, at least, it is not without relevance to design a computational approach that finds feasible solutions that account for $n$-tuple incompatibilities, whilst simultaneously accounting for possible compatibilities between any $n - 1$ options within the $n$-tuple. Second, it would be useful if AIDA could be developed to incorporate stochastic processes whereby probabilities, and especially conditional probabilities, of the occurrence of decision areas, of options, and of incompatibilities between options might be systemically incorporated. Any one of these two developments and, especially, both together, would enhance the versatility of AIDA for group decision situations characterized by uncertainty and complexity.

## References

Ackoff R (1967) Management misinformation systems. Manag Sci 14(4):B147–B156

Aho AV, Hopcroft JE, Ullmann JD (1974) The design and analysis of computer algorithms. Addison-Wesley, New York

Aldous JM, Wilson RJ (2000) Graphs and applications: an introductory approach. Springer, London

Banxia (2005) Decision explorer online reference. Version 3.3. Banxia Software Limited, Kendal

Barnard JM (1993) Substructure searching methods: old and new. J Chem Inf Comput Sci 33(4):532–538

Batagelj V (1991) Some mathematics of network analysis. Network seminar, Department of Sociology, University of Pittsburgh

Batagelj V (2003) Efficient algorithms for citation network analysis. University of Ljubljana, Institute of Mathematics, Physics and Mechanics, Department of Theoretical Computer Science, Preprint Series, 41, 1–27 (arXiv:cs/0309023v1, 14 September)

Batagelj V, Cerinšek M (2013) On bibliographic networks. Scientometrics 96(3):845–864

Batagelj V, Mrvar A (1998) Pajek: a program for large network analysis. Connections 21(2):47–57

Batagelj V, Mrvar A (2008) Analysis of kinship relations with Pajek. Soc Sci Comput Rev 26(2):224–246

Batagelj V, Mrvar A (2014) Pajek. In: Alhajj R, Rokne J (eds) Encyclopedia of social network analysis and mining. New York: Springer, pp 1245–1256. [Also see the latest version online: Batagelj V, Mrvar A (2018) Pajek and PajekXXL. In: Alhajj R, Rokne J (eds) Encyclopedia of social network analysis and mining. Springer, New York. https://link.springer.com/referenceworkentry/10.1007/978-1-4939-7131-2_310. Accessed 14 Oct 2018

Batagelj V, Zaversnik M (2011) Fast algorithms for determining (generalized) core groups in social networks. Adv Data Anal Classif 59(2):129–145

Batagelj V, Doreian P, Ferligoj A, Kejžar N (2014) Understanding large temporal networks and spatial networks: exploration, pattern searching, visualization and network evolution. Wiley, Chichester

Bayazit N, Esin N, Ozsoy A (1981) Integrative approach to design techniques. Des Stud 2(4):215–223

Blandford S, Hope RP (1985) Systematic methods for the problem solving process with particular reference to design. IEE Proc A Phys Sci Meas Instrum Manag Educ Rev 132(4):199–212

Bonnici V, Giugno R, Pulvirenti A, Shasha D, Ferro A (2013) A subgraph isomorphism algorithm and its application to biochemical data. BMC Bioinform 14(7):S13

Brualdi RA (2018) Introductory combinatorics. Fifth edition [2018 reissue]. Pearson, New York

Burbidge JL (1973) AIDA and group technology. Int J Prod Res 11(4):315–324

Cartwright D, Harary F (1956) Structural balance: a generalization of Heider's theory. Psychol Rev 63(5):277–293

Chartrand G (1977) Introductory graph theory. Dover, New York

Cohen MN (1995) Lewis carroll: a biography. Alfred A. Knopf, New York

Conte D, Foggia P, Sansone C, Vento M (2004) Thirty years of graph matching in pattern recognition. Int J Pattern Recognit Artif Intell 18(03):265–298

Cook DJ, Holder LB (2007) Mining graph data. Wiley, Chichester

de Nooy W, Mrvar A, Batagelj V (2018) Exploratory social network analysis with Pajek. Revised and expanded edition for updated software: third edition. Cambridge University Press, Cambridge

Dorst K, Royakkers L (2006) The design analogy: a model for moral problem solving. Des Stud 27(6):633–656

e Costa CAB, Lourenço JC, Oliveira MD, e Costa JCB (2014) A socio-technical approach for group decision support in public strategic planning: the Pernambuco PPA case. Group Decis Negot 23(1):5–29

Eppstein D (1999) Subgraph isomorphism in planar graphs and related problems. J Graph Algorithms Appl 3(3):1–27

Fillmore J, Williamson S (1974) On backtracking: a combinatorial description of the algorithm. Soc Ind Appl Math J Comput 3(1):41–55

Foggia P, Percannella G, Vento M (2014) Graph matching and learning in pattern recognition in the last 10 years. Int J Pattern Recognit Artif Intell 28(01):1450001

Friend J (1989) The strategic choice approach. In: Rosenhead J (ed) Rational analysis for a problematic world: problem structuring methods for complexity, uncertainty and conflict. Wiley, Chichester, pp 121–157

Friend J (1992) New directions in software for strategic choice. Eur J Oper Res 61(1–2):154–164

Friend J (2001) The strategic choice approach. In: Rosenhead J, Mingers J (eds) Rational analysis for a problematic world revisited: problem structuring methods for complexity, uncertainty and conflict, 2nd edn. Wiley, Chichester, pp 115–149

Friend J (2014) Starting to build a useful science of public policy choice. Operational Research Society, Document Repository, IOR Legacy Section. Birmingham: West Midlands. Available at the following URL: http://www.theorsociety.com/DocumentRepository/Browse.aspx?CatID=1. Accessed 26 Nov 2018

Friend J, Hickling A (2005) Planning under pressure: the strategic choice approach, 3rd edn. Elsevier Butterworth Heinemann, Oxford

Friend J, Norris ME, Stringer J (1988) The institute for operational research: an initiative to extend the scope of OR. J Oper Res Soc 39(8):705–713

Friend D, Friend J, Cushman M (2002) STRAD: the STRategic ADviser, version 2.3 for Windows, release 2.30 July 2002. User's Manual. Stradspan Limited, Sheffield

Gat D, Gonen A (1981) Orientation map for planning and design methods. Des Stud 2(3):171–175

Golomb SW, Baumert LD (1965) Backtrack programming. J Assoc Comput Mach 12(4):516–524

Hage P, Harary F (1983) Structural models in anthropology. Cambridge University Press, Cambridge

Harary F (1957) Structural duality. Behav Sci 2(4):255–265

Harary F (1969) Graph theory. Addison Wesley, Reading

Harary F, Jessop N, Luckman J, Stringer J (1965) Analysis of interconnected decision areas: an algorithm for project development. Nature 206(4979):118

Hickling A (1978) AIDA and the levels of choice in structure plans. Town Plan Rev 49(4):459–475

Hope RP, Sharp JA (1989) The use of two planning decision support systems in combination for the redesign of an MBA information technology programme. Comput Oper Res 16(4):325–332

Hsiao S-W, Chou J-R (2004) A creativity-based design process for innovative product design. Int J Ind Ergon 34(5):421–443

Hsiao S-W, Chuang JC (2003) A reverse engineering based approach for product form design. Des Stud 24(2):155–171

Jones JC (1970/1992) Design methods, second edition. Wiley, Chichester

Jones JC (1979) Designing designing. Des Stud 1(1):31–35

Kammeier H (1998) A computer-aided strategic approach to decision making in urban planning: an exploratory case study in Thailand. Cities 15(2):105–119

Kirsh D (2000) A few thoughts on cognitive overload. Intellectica 30(1):19–51

Knuth D (1997) The art of computer programming: volume 1, fundamental algorithms, 3rd edn. Addison Wesley, Upper Saddle River

Knuth D (2015) The art of computer programming: volume 4, satisfiability, fascicle 6. Addison Wesley, Boston

Krissinel EB, Henrick K (2004) Common subgraph isomorphism detection by backtracking search. Softw: Pract Exp 34(6):591–607

Larrosa J, Valiente G (2002) Constraint satisfaction algorithms for graph pattern matching. Math Struct Comput Sci 12(4):403–422

Luckman J (1967) An approach to the management of design. OR 18(4):345–358

Norese MF, Rolando D, Fregonara E (2015) Integration of problem structuring methods: a methodological proposal for complex regional decision-making processes. Int J Decision Support Syst Technol 7(2):58–83

Phahlamohlaka J, Friend J (2004) Community planning for rural education in South Africa. Eur J Oper Res 152(3):684–695

Randic M (1978) Fragment search in acyclic structures. J Chem Inf Comput Sci 18(2):101–107

Randic M, Wilkins CL (1979) Graph-based fragment searches in polycyclic structures. J Chem Inf Comput Sci 19(1):23–31

Rosenhead J (1996) What's the problem? An introduction to problem structuring methods. Interfaces 26(6):117–131

Rosenhead J, Mingers J (2001) Rational analysis for a problematic world revisted: problem structuring methods for complexity, uncertainty and conflict. Wiley, Chichester

Saaty TL (1980) The analytic hierarchy process. McGraw-Hill, New York

Schoch D (2015) A positional approach for network centrality. Dissertation submitted for the degree of Doctor of Natural Sciences, Department of Computer and Information Science, Universität Konstanz. Available at the following URL: https://kops.uni-konstanz.de/handle/123456789/34821. Accessed 11 Oct 2018

Schoch D, Brandes U (2016) Re-conceptualizing centrality in social networks. Eur J Appl Math 27(6):971–985

Sharifzadegan MH, Fathi H, Zamanian R (2014) Using strategic choice approach in urban regeneration planning (Case study: Dolatkhah area in Tehran, Iran). Int J Archit Urban Dev 4(2):45–52

Smith RP, Morrow JA (1999) Product development process modeling. Des Stud 20(3):237–261

Stradspan Ltd. (2014) http://www.stradspan.com/about.htm and http://www.stradspan.com/products.htm. Retrieved 10 Feb 2018

Tarjan RE (1972) Depth-first search and linear graph algorithms. Soc Ind Appl Math J Comput 1(2):146–160

Todella E, Lami IM, Armando A (2018) Experimental use of strategic choice approach (SCA) by individuals as an architectural design tool. Group Decision and Negotiation https://doi.org/10.1007/s10726-018-9567-9 (online pending print publication)

Tsang E (1993) Foundations of constraint satisfaction. Academic Press, London

Vento M (2015) A long trip in the charming world of graphs for pattern recognition. Pattern Recognit 48(2):291–301

Weas A, Campbell M (2004) Rediscovering the analysis of interconnected decision areas. Artif Intell Eng Des Anal Manuf 18(3):227–243

Wells MB (1971) Elements of combinatorial computing. Pergamon, Oxford

Willett P, Barnard JM, Downs GM (1998) Chemical similarity searching. J Chem Inf Comput Sci 38(6):983–996

Wilson B (2001) Soft systems methodology: conceptual model building and its contribution. Wiley, Chichester

Wu F-G, Sun H-H, Lin Y-C (2015) Innovative aid design of moving kitchenware for elders. Procedia Manuf 3:6266–6273