Chapter 19
PyHasse Software for Partial Order Analysis: Scientific Background and Description of Selected Modules

Rainer Brüggemann, Lars Carlsen, Kristina Voigt, and Ralf Wieland

Abstract The software PyHasse is an elaborated "experimental" software for ordinal analysis of data matrices. PyHasse is based on the interpreter programming language Python. A brief introduction to the programming language Python is given and the general principles behind PyHasse are outlined. An actual overview about PyHasse (status, April 2013) is provided. Today PyHasse comprises 91 modules covering 9 different categories, such as basic Partial Order Analysis, i.e., the drawing Hasse diagrams and the calculation of some important quantities. A selection of newer or rarely used modules are discussed in detail in order to explain some principles of PyHasse. As a leading example the pollution by Lead, Cadmium, and Zinc of regions of south-western Germany is discussed.

An outlook is given, where future projects are discussed. Such projects comprise among others, Internet access to some of the more important modules, inclusion of the Formal Concept Analysis tools, and of tools derived from POSAC and the variance-based sensitivity.

R. Brüggemann (✉)
Department of Ecohydrology, Leibniz-Institute of Freshwater Ecology and Inland Fisheries, Mueggelseedamm 310, Berlin, Germany
e-mail: brg_home@web.de

L. Carlsen
Awareness Center, Linkøpingvej 35, Trekroner, DK-4000 Roskilde, Denmark

Center of Physical Chemical Methods of Research and Analysis, al-Farabi Kazakh, National University, 96A Tole Bi street, 050012 Almaty, Kazakhstan

K. Voigt
Institute of Computational Biology, Helmholtz Center, Munich, Neuherberg, Germany

R. Wieland
Leibniz Centre for Agricultural Landscape Research (ZALF), Institute of Landscape Systems Analysis, Muencheberg, Germany

19.1 Introduction

The analysis of multi-indicator systems (Brüggemann and Patil 2010, 2011), aiming at a ranking of multiple characterized objects is of increasing interest in many scientific fields, e.g., environmental health (Voigt et al. 2011, 2012) or sociology and politics (Annoni 2007; Carlsen and Brüggemann 2013a, b). In this context, partial order methodology appears to be increasingly applied (see Brüggemann and Carlsen 2012 in their response to Huang et al. 2011). The tools of partial order are not as ancient as those of general decision-making methods which started with the scientific work of Condorcet, Borda, at the end of the eighteenth century (cf. Munda 2008). Partial order as a mathematical discipline seems to go back to the late nineteenth century, where Dedekind was exploring the Diedergroups. Strong impacts on the theory of partially ordered sets can be related to Hasse (1927, 1952) and Birkhoff (1984), two mathematicians who, as Dedekind, were mainly interested in algebraic aspects. Within the context of data matrices, i.e., within a statistical point of view, main contributions can be traced back to Patil on the one side (within the context of biological diversity, see Patil and Taillie 1976), and, without knowing each other, to the team Halfon and Reggiani (Halfon and Reggiani 1986), on the other side. The work of Halfon and his coauthors gave the basis for the computerized Hasse diagram technique (HDT), which is specifically related to partial order and their application to the ranking of objects simultaneously described by several indicators, i.e., by data matrices. A third line of development of the analysis of data matrices can be identified, which is the field of Formal Concept Analysis (FCA), developed in the 1980s (Ganter and Wille 1996), which also finds increasing interest (see for instance Bartel and Brüggemann 1998; Davey 2004; Carlsen 2009; Brüggemann and Patil 2011).

The application of many of the tools of partial order theory on data matrices is a priori extremely simple, however, tedious if performed manually. Therefore, it is understandable that together with development of computer programming, an increasing and more and more detailed support in the ordinal analysis of data matrices is ongoing.

In this book some chapters describe the application of selected modules of the PyHasse package, whereas in Brüggemann and Patil (2011), a state-of-the-art overview (by 2010) of the software packages Rapid and PyHasse is given.

This chapter explains some background material on Python, the programming language on which PyHasse is based, and renders some more and general information about PyHasse.

19.2 HDT Software

An overview about software, a status by 2006, was given by Halfon (2006).

A complete overview about theory and applications of partial order on multi-indicator systems is outside the scope of this chapter, which instead aims at a description of PyHasse. For introductory texts we refer to papers by Brüggemann et al. (2001) and Brüggemann and Voigt (2008).

Table 19.1 Software aiming at partial order analysis of data matrices

Software	Authors	Remark	Reference
Hasse	Halfon	Drawing Hasse diagrams	Halfon et al. (1986)
WHASSE	Brüggemann	Drawing Hasse diagrams and first attempts to introduce tools beyond the drawing	Brüggemann et al. (1999)
conimp4	Burmeister	Analyzing data matrices on the basis of Formal Concepts	Burmeister, CONIMP4, Programm zur Formalen Begriffsanalyse (1997)
conexp	Yevtushenko	Analyzing data matrices on the basis of Formal Concepts	Yevtushenko (2003) http://www.comp.dit.ie/pbrowne/compfund2/UserGuide.pdf (assessed 7 Nov 2012) Download, see for instance: http://sourceforge.net/projects/conexp/files/conexp/1.3/ (accessed Aug 2013)
DART	Talente (Manganaro et al. 2008)	Drawing Hasse diagrams, utility functions	Manganaro et al. (2008)
ProRank	Pudenz	Drawing Hasse diagrams with emphasis on simple data management	Pudenz (2005)
Rapid	Joshi, Brüggemann and Patil	Drawing Hasse diagrams, some analysis tools	Brüggemann and Patil (2011)
PyHasse	Brüggemann and Patil	Analyzing partially ordered sets, derived from data matrices	Brüggemann and Patil (2011)
Parsec	Fattore	Analysis on the basis of R; see also Myers and Patil (2010, 2014)	Fattore and Arcagni, Chap. 16

In Table 19.1 a—certainly not complete—overview of software is given. The newest (so far the authors are aware), the PyHasse, will be explained in more detail in this chapter.

19.3 Python as Programming Language for Contemporary Software Generation

Clearly the first question often stated may be: Why Python, why not JAVA, PERL, or traditional languages such as C++, Fortran, or VisualBasic? The most honest answer is, simply because Python fulfills to a wide degree the personal taste of the programmer, in this case of Rainer Brüggemann. This very personal way to find a decision about the suitable programming language may be unsatisfactory for many readers. Hence, we discuss some objective points that in the view of the programmer favor Python (however, without arguing that other programming languages do not have these features).

19.3.1 General Remarks

In the present context the arguments are following those of Lutz and Ascher (2003) closely, although there is a book available, where specifically Python for scientific uses is explained and which is recommended for further reading (Langtangen 2009).

When Python was developed by Guido von Rossum (cf. Venners 2003) it was developed in one step. Hence, Python had a very homogenuous structure from the very beginning. Clearly, Python has been further developed and will be further developed in the future. Actually, currently Python is delivered in version 3, whereas PyHasse is developed on the basis of Python 2.6, available since around 2007.

Python is—in contrast to C languages—comfortably readable and coherent. Python supports consequently the object-oriented programming style.

It is of further interest that typically Python codes are “1/3 to 1/5 the size of equivalent C++ or Java code” (Lutz and Ascher, page 3, 2003).

Briefly speaking there is a Python slogan which says that “In the Python way of thinking, explicit is better than implicit, and simple is better than complex” (Lutz and Ascher 2003, page 5).

Python is an interpreter language. That means, there is no need to compile and link the software before it is applied. In the Web site python.org, we find “Python is a programming language that lets you work more quickly and integrate your systems more effectively.” Clearly, it is to be expected that the linear reading of the programming code may be time consuming. However, the personal experience is that even the combinatorial algorithms, which are typical for the application field of partially ordered sets do not need much time, i.e., even the impatient programmer may await the result, sitting before his machine!

Technically spoken, Python belongs to the Very High-Level Languages (VHLL) (Müller and Schwarzer 2007). For the PyHasse author, the fact that developing new modules and testing them does not need to first compile parts of the program makes Python a very efficient and quick programming tool.

19.3.2 Portability

The portability of Python programs is high. For example, PyHasse programs run without problems on different Windows operating systems as well as on different UNIX or Linux machines. Although there is not much experience with Macintosh operating systems, examples are known that PyHasse can be ported to the Macintosh without major difficulties.

19.3.3 Libraries

As most other modern programming languages Python provides many freely downloadable libraries. All possible applications of "modern life" can be handled by such libraries:

- NumPy and Matplotlib: powerful libraries for numerical calculations and visualization which can replace MATLAB in many applications.
- Statistical libraries, drawing libraries are available, as well as libraries designed to handle databases (MySQL, Oracle, …).
- The Internet programming libraries are of increasing importance and many web frameworks provide support for a quick construction of Web sites for example Plone (with parts of ZOPE) or Django.

Also more exotic applications are supported like:

- PIL: (PhotoImageLibrary) a library for manipulating electronically photos
- PyGame: a library facilitating game programming

19.3.4 Programming Support

One important point is that Python supports the development of own written libraries, which are specifically designed for the scientific purpose of any software package.

Another important point is that Python supports the development process by a set of tools: For example, Cython expands Python adding type information to a Python program to make Python modules faster. Cython may further be used to include C/C++ Code in a Python library. Alternatively SIP or SWIG can wrap existing C/C++ code to use it as a Python module. There are modules available that include the QT library which allows to implement modern graphical user interfaces or another module which includes the gnu scientific library (gsl).

For some applications like Monte-Carlo Simulations, Python as interpreter language is too slow. Techniques, discussed above, may be used to accelerate Python modules. It should further be noted that Python supports parallel programming. Thus, modules like PyPar connects Python programs to the powerful Message Passing Interface (MPI) and allow parallel processing even on a personal computer with four or eight cores. This underlines that Python is not restricted to fast prototyping but Python is a modern programming toolbox, which can be used for challenging projects.

Some useful links are:
python: http://www.python.org/
numpy, scipy: http://numpy.scipy.org/
matplotlib: http://matplotlib.org/

networkX: http://networkx.lanl.gov/
Cython: http://cython.org/
SWIG: http://www.swig.org/
QT: http://qt.digia.com/
PyQT: http://www.riverbankcomputing.co.uk/software/pyqt/intro
gsl: http://www.gnu.org/software/gsl/
PyPy: http://pypy.org/
Pypar: http://code.google.com/p/pypar/

19.4 A Practical Ranking Problem, i.e., a Test Set for Explaining PyHasse

In order to have an illustrative example at hand we look at nine regions in Germany. In order to measure the air quality, the concentrations of deposited Pb, Cd, and Zn are monitored in epiphytic mosses (Brüggemann et al. 1998). The question arises: Can we rank the regions simultaneously taking into account the concentrations of all three metals? In Table 19.2 the data matrix is shown.

Partial order theory provides an answer as displayed as a Hasse diagram, i.e., a transitively reduced acyclic, trianglefree digraph of order relations (as explained in several chapters of this volume) (Fig. 19.1). The first observation is that the

Table 19.2 Data matrix, nine regions, three metals, concentrations in mg/kg dry weight (rounded)

Region	Pb (Lead)	Cd (Cadmium)	Zn (Zinc)
6	11	0.2	31
8	20	0.4	55
7	14	0.3	41
17	13	0.3	63
9	17	0.3	45
16	13	0.4	51
14	12	0.6	41
5	14	0.4	45
29	9	0.4	29

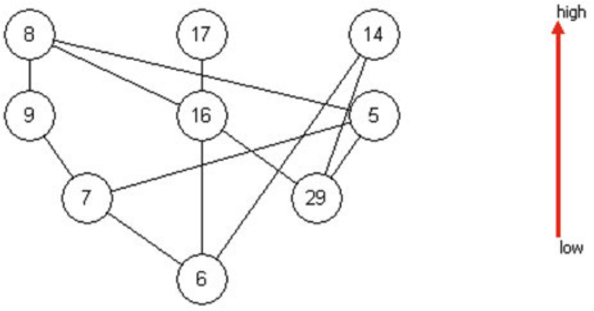


Fig. 19.1 Hasse diagram of nine regions with three attributes, namely the (rounded) metal concentrations in epiphytic mosses of Pb, Cd, and Zn

Hasse diagram is not slim, i.e., it obviously deviates remarkably from a linear order (Fig. 19.1).

Figure 19.1 shows the main characteristic of HDT: There are regions (generally "objects") which cannot be compared, as, e.g., region 8 and 17. The reason is that region 8, with respect to one metal, has a higher concentration than region 17, whereas region 17 on the other hand, has a higher concentration than region 8 with respect to another metal. Thus, the two regions are "in conflict with each other." Technically we describe an incomparability by the symbol ||, i.e., here as 8 || 17. A set of objects which are mutually incomparable is called an antichain, in contrast to a set of objects, which are mutually comparable, i.e., a chain. Hence, the set {9, 16, 5} is an antichain, whereas the set {6, 7, 9, 8} is a chain.

19.5 The PyHasse Software

19.5.1 Intention Behind the Software

19.5.1.1 Modules

PyHasse is a software consisting of a series of mutually independent programs. These programs are called "modules." When programming tools, as well as interfaces and all the partial order analysis tools are counted, the complete number of modules of PyHasse software is 91 (April 2013). However, this number is continuously changing, as new modules may replace a couple of older modules or new ideas to analyze partial orders derived from the ordinal analysis of data matrices eventually result in new modules.

In total, PyHasse is a software package with more than 50,000 lines of programming code (including comment lines, empty lines, which help to get a clear program code). Obviously, parts of program codes often appear several times, due to the intention that the modules should be mutually independent.

19.5.1.2 PyHasse as Experimental Software

PyHasse is intended to help solving daily problems applying partial order concepts on data matrices. It does not intend to provide either perfect statistical or graphical tools, especially it does not intend to include the vast number of applications, which more or less routinely are performed by applying spreadsheet software, such as Microsoft Excel®. The same kind of philosophy holds when a drawing of Hasse diagrams is considered. Thus, virtually all PyHasse modules offer the drawing of Hasse diagrams following the drawing convention, which has its origin in the work of Halfon (Halfon and Reggiani 1986). Nevertheless, these PyHasse-generated graphs are far from being perfect drawings. Hence in this context PyHasse cannot compete with the powerful freely downloadable program Graphviz, see Gansner and North (1999), which visualizes partially ordered sets in an almost perfect manner (see Sect. 19.5.4.3).

In sum PyHasse tries to fill the gap between highly specialized programs often developed in laboratories but not generally applicable and professionally written software, which usually may not reflect the state of the art of the theoretical development, even though updates are made available from time to time.

19.5.2 Basic Structure

19.5.2.1 Contextual Categories

PyHasse is structured in two ways: Contextually and from the programming point of view. In Table 19.3, nine contextual categories are explained.

In Fig. 19.2 a bar diagram displays the distribution of the PyHasse modules over the nine categories described in Table 19.3.

19.5.2.2 Programming Structure

The 91 modules are supported by four libraries (Table 19.4).

These four libraries are delivered together with the PyHasse modules (and some additional files) and the user has to put them into the folder, where Python is localized.

In order to facilitate the installation of PyHasse software, the programmer, Brüggemann, did not extensively use other comfortable libraries, such as MatplotLib or NumPy.

Together with the utility functions, the programming structure can be characterized by a scheme, as shown in Fig. 19.3.

19.5.2.3 Graphical User Interface

Most of the modules have similar graphical user interfaces (GUIs). In Python GUIs can be programmed, applying the standard library Tkinter, which is derived from Tcl/Tk. Thus, all user interfaces in the PyHasse package are built using Tkinter. The location of the typos: buttons (which govern the user activity) are vertically arranged following the most typical logical sequence of steps. A few modules are menu oriented, such as pyhassemenue8_3.py, DAHP.py, and modelHD9.py. In almost every user interface an "about" function is found, which informs briefly about the aim of the module and the programmer and (sometimes) about the leading idea out of the literature.

Table 19.3 Alphabetically sorted contextual categories of PyHasse, references are found in the Appendix

Group	Explanation	Modules	Described or applied in this book
AC	Antichain analysis in the broadest sense	antag2.py	Chap. 18
	Why incomparabilities of objects appear in antichains, but also, why different sets of vertices in a Hasse diagram are not or only loosely connected	antagscattplot2.py	Chap. 18
	In hdgt6.py some graph–theoretical concepts are programmed to explore distances in the Hasse graph	antichain20_4.py	Chap. 18
		checksepset5.py	
		findsomesepset4.py	
		hdgt6.py	
		recocgnizesepset3.py	
		sepanal15_3.py	
		sepanal16_coloured.py	
Comp-Ind	The intention is near to a concept having its origin in Patil's intention of "comparative knowledge discovery" (Patil and Joshi 2014), namely to explore the role of weights when composite indicators are known or are to be constructed	canonweight9_2.py	Chaps. 6 and 7
	In owa4.py the fuzzy concept after Yager (Yager 1988, 1993) is applied	canonweight_3D-grid2.py	Chap. 2
	CompInd is in pyhassemenue8 listed under MCDA	conflict7.py	
		gevol2.py	
		HDCI6.py	
		linagg10.py	
		owa4.py	
		stability9.py	
		weightbased modeling2.py	
Inter-face	Partial order may be expressed in different ways. When zeta or cover matrices are used, then some of the interface modules can read these matrices	covi_interactive3.py	
	Different PyHasse modules can communicate with each other applying a special format. Files with this internal format have the extension, i.e., *.pdt	covreader4.py	
	Finally an interface to apply graphviz is available	graphviz2.py	
		txtpdt_interface1.py	
		zetareader1.py	

(continued)

Table 19.3 (continued)

Group	Explanation	Modules	Described or applied in this book
LinExt	Beside the directed acyclic graph displaying partially ordered sets (Hasse diagram), partially ordered sets can be presented by a set of linear extensions	avrank5.py	Chap. 6
		avrkmut3.py	Chap. 6
	From linear extensions average ranks can be derived, as well as probabilities as prob($x>y$) or prob(x: rank=Rk) Winkler 1982; Brüggemann and Carlsen 2011; Wienand 2012	genlinext1.py	
		linext_play2.py	
		LPOMext4_2.py	
	This group contains modules, which are most crucial with respect to memory limitations, especially avrank5.py	LPOMstruct1.py	
		mutprobavrk.py	
		BubleyDyer8.py	
		CRF1.py	
MCDA	PyHasse contains some "classical" MCDA methods such as PROMETHEE (Brans and Vincke 1985), the concordance–discordance analysis of ELECTRE (Peters and Zelewski 2007; Opperhuizen and Hutzinger 1982), AHP (Saaty 1994), as well as TOPSIS and DEA (http://mat.gsia.cmu.edu/classes/QUANT/NOTES/chap12.pdf). However, typically these methods are included in extremely simplified versions. Beyond this, there are several variants available, where an outranking algorithm, for example, the derivation of the Copeland index is applied (Al-Sharrah 2010, 2011)	condor2_2.py	
		copel4.py	
		dahp4	
		deaMC1.py	
		Discordance3.py	
		genoutrk7.py	
		oreste6.py	
		outrkHD6.py	
		prom6_2.py	
		topsissimpl1.py	
METEOR	Method of evaluation by order theory. The basic idea is to analyze the role of weightings of the single descriptors by a stepwise procedure. METEOR could also be seen as a part of CompInd	cap7.py	
		HDweightMC2.py	
		meteorHD4.py	
		meteorparallel2.py	
Model	In the broadest sense, proximity (similarity) analysis as well as exploring partially ordered sets being most similar to a given one can be seen as a modeling technique. Furthermore, the influence of graph–theoretical structures in Hasse diagrams on average ranks is of interest and can be analyzed by "model posets"	combsimilarity7.py	Chap. 17
		concord2_2.py	
		modelHD9.py	
		similarity10_1.py	
		similarity_search7_2.py	

POT	Drawing Hasse diagrams in different configurations of inputs or of the graph, with different additional information, such as navigation tools or even a simple approach to estimate average ranks	mainHD20_5.py	Chap. 18
	A fuzzy concept of partial order after Kosko is included, as well as a MC simulation, fuzzydds7.py, after Wieland and Brüggemann (2013). A nonnumerical aggregation based on concepts developed by Carlsen (2008) is realized in hpor3.py. The module probranksThAC3.py follows an idea, published by Thiessen and Achari (2012)	mainHD20_5.py (without "psyco")	Chaps. 11 and 18
		mHDCl2_7.py	Chap. 10
		chain7_1.py	Chap. 2
		dds12.py	
		ExcelHD1.py	
		fuzzyHD13_2.py	
		fuzzydds7.py	
		graphvizHD1.py	
		HDalter6_1.py	
		HDedit3.py	
		HDscratch2.py	
		HDsimpl1.py	
		HD_msptree1.py	
		hpor3.py	
		interval8.py	
		levelheuristic2	
		majoriz1.py	
		palg4_2.py	
		pooc6.py	
		POTanalysis2.py	
		sensitivity18_3.py	
		SingleObjects_analysis4.py	
		pycluster1_2.py	
		orientation1.py	
		POcdn1.py	
		probranksThAC3.py	

(continued)

Table 19.3 (continued)

Group	Explanation	Modules	Described or applied in this book
Utility	This group of models is least relevant for the users. They are mainly helpful in documentation and programming. Some of those modules are internally called by pyhassemenue8.py. Another, pir3.py, can be used to get a complete tutorial text by means of all the help texts available for the PyHasse modules	discretiz3.py instruction1.py pir3.py pyhasseinfo1.py pyhassemenue8_4.py pyHasse_progr1.py randomdm2.py tool_libraryreader1.py utility_libraryreader1.py utility_PyHassecount1.py utility_PyHasseInfo1.py utility_PyHasse_newsince1.py (abbr. UPN) utility_PyHasse_interval1.py	

Modules described in this chapter are written in italic and bold characters

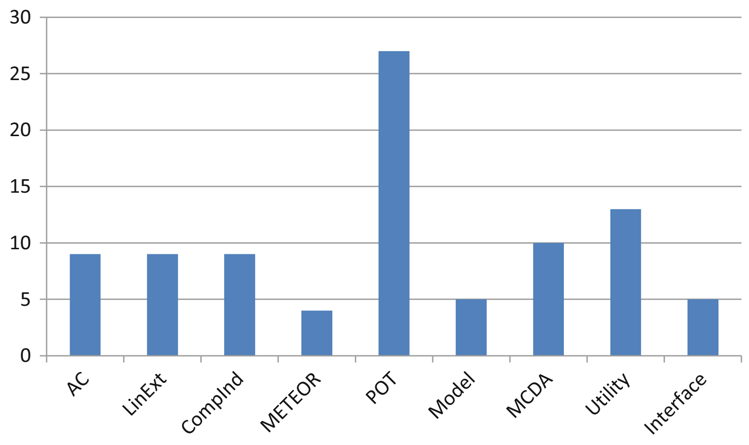


Fig. 19.2 Distribution of the 91 PyHasse modules within the nine contextual categories given in Table 19.3

Table 19.4 Libraries, supporting the PyHasse modules

Name of library	Description	Remark
raioop2.py	A library of classes, i.e., on procedures based on object oriented programming Mainly: user interfaces and graphics	Written by Brüggemann 4,500 lines of programming code
rmod2.py	Library of procedures, mainly of combinatorial character and manipulating matrices	Written by Brüggemann More than 6,800 lines of programming code
pstat.py	Statistics	Free downloadable from Internet. However, routinely delivered together with the other two libraries above as part of the PyHasse package
stats.py	Statistics	Free downloadable from Internet: however routinely delivered together with the other two libraries above as part of the PyHasse package

In addition, there is a “help” function, which has the following structure:

- Aim
- Prerequisites
- Usage or steps
- Results (not in all cases)
- Difficulties
- Literature
- Example data files

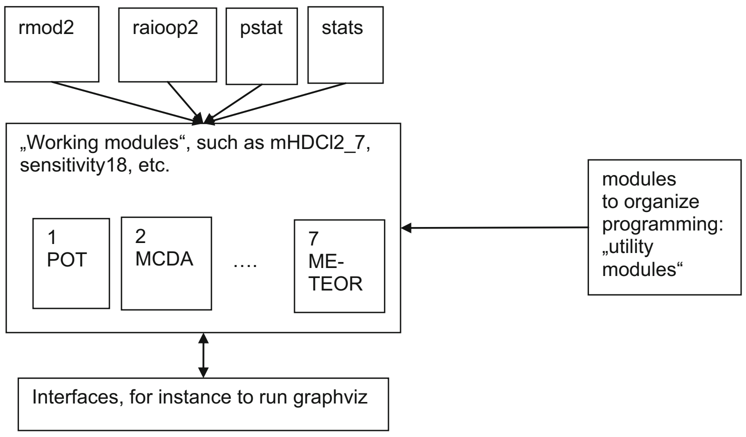

rmod2
raioop2
pstat
stats
„Working modules“, such as mHDCl2_7, sensitivity18, etc.
1 POT
2 MCDA
….
7 ME-TEOR
modules to organize programming: „utility modules“
Interfaces, for instance to run graphviz

Fig. 19.3 Programming structure of PyHasse

19.5.2.4 PyHasse Data Flow (Example: Windows® as Operating System)

Within the Windows® environment the majority of potential users will apply Microsoft Excel®.

In order to fulfill the input requirement for the PyHasse module, it is important that the rows as well as the columns have a short label (optimal are labels with up to three characters) and that the (0,0) position of the data matrix (in Excel the A,1) is not empty. Furthermore, none of the PyHasse modules accept data gaps. Hence, it is in the responsibility of the users to provide a data sheet with all labels and no data gaps. In contrast, software packages such as DART (see Manganaro et al. 2008) and WHASSE (Brüggemann et al. 1999) provide some facilities to handle missing data.

Typically the PyHasse modules require the Excel sheet stored as a tab-separated txt file. Only the module EXCELHD1.py can directly apply the data by copying the appropriate field in the Excel sheet. Once the data matrix is read in, one may perform calculations and results can be stored in the internal format pdt. Some more important modules therefore offer to read these intermediate results as *.pdt files.

19.5.3 Overview

19.5.3.1 Most Often Used Modules

The application of the following modules is well described (cf. Table 19.1 and the appendix at the end. Further, specific references are available within the single modules).

- mainHD20_5.py and mHDCl2.py, resp.: Beside the Hasse diagram, these module provide navigation tools and much structural information, as well a variety of other facilities. As “basic” modules these are the most important

- chain7_1.py: Search and analysis of chains
- dds12.py: Dominance and separability of disjoint subsets of objects on the basis of the order relations among their elements
- LPOMext4_2.py: Average ranks calculated after two different approximations based on the “local partial order concept”
- fuzzyHD13.py: Instead of analyzing the “<” relation directly a subsethood is defined (Kosko measure, cf. Van de Walle et al. 1995) and a fuzzy partial order defined
- sensitivity19_1.py: A partially ordered set has a structure. This structure is characterizable by chains and antichains. What is the impact of any single matrix column (representing the indicator values for all the objects)? i.e., what is the impact of any single indicator on the structure of a poset?
- similarity10_1.py: The same set of objects may be described by different multi-indicator systems. What is the proximity between the two resulting posets?

19.5.4 Description of Some Modules of PyHasse Software

19.5.4.1 Module mHDCl2_7: The “New Main”

This module is one of the newest and is completely written in an object oriented programming style. The reason, why mHDCl2_7.py was developed, was threefold:

1. The similar module mainHD20_5.py runs into memory error when the data matrices are too large
2. The GUI and the logical organization were no more adequat
3. After some years of practical applications some adaptions appeared appropriate

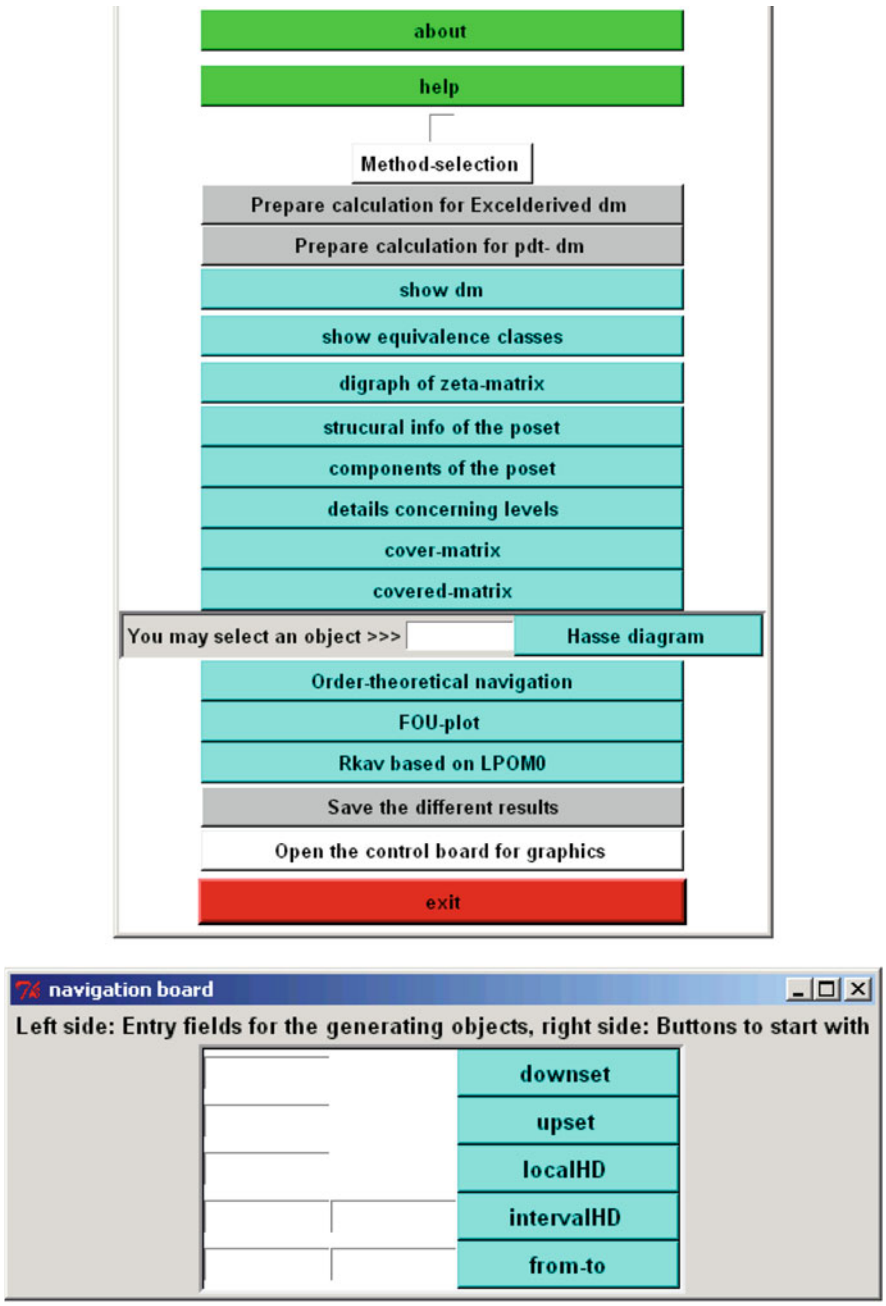
The purpose is, as with mainHD20_5.py, to provide a complete basical analysis of a partially ordered set as derived from a data matrix. This includes as results:

- Level structure
- Information of each object about its successors, predecessors, and incomparable objects in the Hasse diagram, in tabular form
- Hasse diagram
- Navigation tools: principal down- and upsets, interval graphs, local Hasse diagrams, the most simple approximation of average rank by the local partial order (LPOM0) (Brüggemann et al. 2004)

The GUI and its subsequent windows are shown in Figs. 19.4 and 19.5.

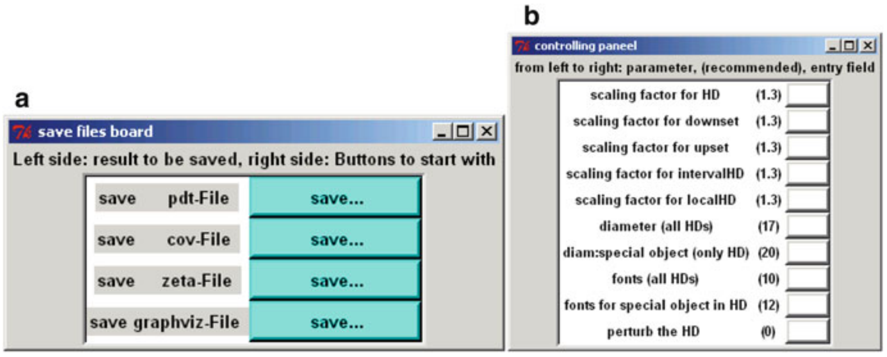
In the following we describe each button given in Fig. 19.5, starting from the top in Table 19.5.

In mHDCl2.py there are three other tools to overcome the difficulties of drawing Hasse diagrams: (a) by rendering information in a tabular form (Table 19.6) and (b) by the FOU plot, which is a realization of the concept of posetic coordinates (see Chap. 8), see Fig. 19.6.


about
help
Method-selection
Prepare calculation for Excelderived dm
Prepare calculation for pdt- dm
show dm
show equivalence classes
digraph of zeta-matrix
strucural info of the poset
components of the poset
details concerning levels
cover-matrix
covered-matrix
You may select an object >>>
Hasse diagram
Order-theoretical navigation
FOU-plot
Rkav based on LPOM0
Save the different results
Open the control board for graphics
exit
navigation board
downset
upset
localHD
intervalHD
from-to

Left side: Entry fields for the generating objects, right side: Buttons to start with

Fig. 19.4 GUI of mHDCl2_7.py and the window opening after pressing "Order theoretical navigation." Note that the first three navigation buttons need the input of one single object, whereas the buttons "intervalHD" and "from–to" need two objects as input



Left side: result to be saved, right side: Buttons to start with

save pdt-File
save cov-File
save zeta-File
save graphviz-File

from left to right: parameter, (recommended), entry field

scaling factor for HD	(1.3)	
scaling factor for downset	(1.3)	
scaling factor for upset	(1.3)	
scaling factor for intervalHD	(1.3)	
scaling factor for localHD	(1.3)	
diameter (all HDs)	(17)	
diam:special object (only HD)	(20)	
fonts (all HDs)	(10)	
fonts for special object in HD	(12)	
perturb the HD	(0)	

Fig. 19.5 Windows popping up after pressing "Save the different results" (**a**) and "Open the control board for graphics" (**b**)

In Fig. 19.6, a FOU plot is shown. Myers and Patil (2014) are focusing on possibilities to represent partially ordered sets by scatter plots in order to avoid too complex Hasse diagrams. Here our aim is similar. The basic idea is to describe partially ordered sets by "posetic coordinates," i.e., by numbers which are derived from partial order theory, e.g., the contents of principal down- and upsets and of $U(x)$. When equivalence relations are possible, the number of equivalent elements could be used too to obtain posetic coordinates. Here we characterize the poset by two order theoretical coordinates for each object x, i.e., by the difference of the contents of down ($O(x)$) and upsets ($F(x)$), OF and the content of the set of elements incomparable with x: U.

$$OF := \left(|O(x)| - |F(x)|\right) \text{ and } U := |U(x)|. \tag{19.1}$$

- In contrast to the coordinates, the original data matrix may render (Pb, Cd, Zn) now posetic coordinates, namely OF and U are used to characterize the objects.
- In contrast to the triangle coordinate representation (Brüggemann and Patil 2011), which is more detailled, the scatter plot, based on OF amd U is simple to be interpreted.

Generally, it is a promising new task in partial order theory to find best "posetic coordinates" allowing presentations of partial orders not so much depending on the clarity of the relational graph, such as the Hasse diagram.

Figure 19.6 shows that

- There are two regions selected (namely 8 and 14) being maximal elements, however, they differ in their values of their posetic coordinates.
- There is one region being at most incomparable $|U(x)| = 7$. object, this is region 17, which also is a maximal element.

Table 19.5 Explanations of the buttons of the GUI of mHDCl2_7.py

Button	Explanation	Remark
Method selection	0 as input A method to perform the transitive reduction is performed, which eliminates step by step the transivities 1 as input A method is used, following Simon (1992), which, however, leads to memory errors when the adjacency matrix is to be calculated and the number of objects is too large (>200) and at the same time the number of comparabilities is high	
Prepare calculation for Excel-derived dm	When this button is pressed, the module expects a data matrix following the principles explained in section "*PyHasse data flow (example Windows as operating system)*"	Internally all calculations are performed to get the Hasse diagrams and other combinatorial results
Prepare calculation for data matrices in the pdt Format	The module expects data matrices in the internal format pdt. This facility is not as often used as the Excel-derived dm	Internally all calculations are performed to get the Hasse diagrams and other combinatorial results
Show dm	Attributes (indicators) as well as the labels of the objects are shown. Furthermore, the complete data matrix is displayed	When the button "Hasse diagram" is pressed, the exact label of the object is needed
Show equivalence classes	If two rows, i.e., two objects have identical values for all indicators then the two objects are considered as equivalent, the alphabetically first object is retained A graphical as well as a tabular presentation of equivalence classes can be obtained	
Digraph of zeta matrix	The zeta matrix describes the order relations among the objects. In contrast to the representation in the Hasse diagram, which is based on the cover relations, the relations corresponding to transitivity of the order relation are shown too	
Structural info of the poset	Being aware that Hasse diagrams can be a complex system of lines (see Carlsen and Brüggemann 2013a) all needed information are provided in tabular form	See Table 19.6
Components of the poset	Graph theoretically the acyclic-directed graph may have vertices which are not connected (in former publications also called "hierarchies"). Here an information about the number of components and the distribution of objects over these components is available	

(continued)

Table 19.5 (continued)

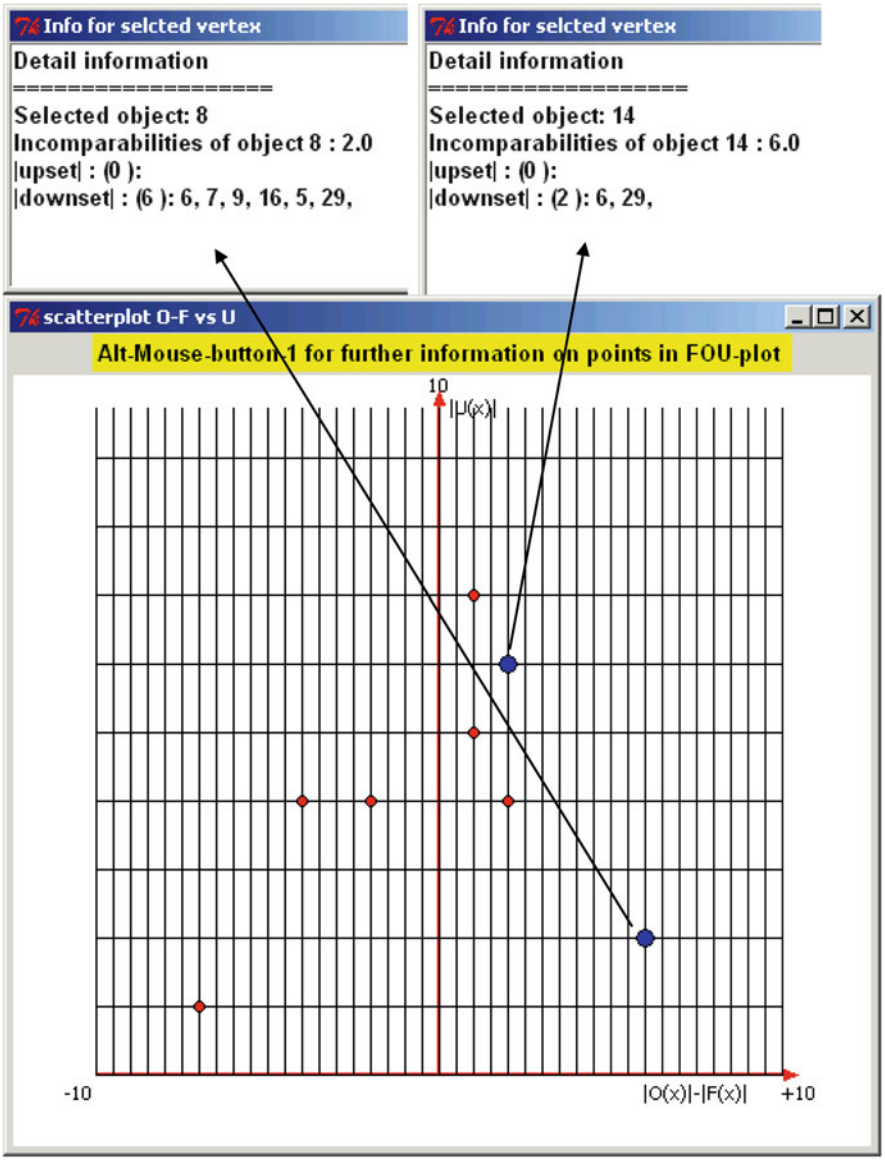
Button	Explanation	Remark
Details concerning levels	Levels are an important structure and a mean to get the set of objects weakly ordered (Brüggemann and Patil 2011) Therefore there is a multitude of more detailed information available	See below
Cover matrix	Even if the Hasse diagram is drawn, one may want to get a list of cover relations Here "Object *x* is covering …" is given	
Covered matrix	Similar as above. However, here "Object *x* is covered by…" is given	
Hasse diagram	A Hasse diagram is drawn. When the entry field at the left side is filled with the correct label of an object, this object will be marked in the graphic	See also Fig. 19.1
Order-theoretical navigation	It pops up an extra window, where it can be specified which navigation is wanted	See for instance Fig. 19.10
FOU plot	In order to analyze large data matrices, the Hasse diagram is often not suitable because of its complexity. Then other representations must be selected, as is pointed out in Myers et al., in several papers and in this book (Myers and Patil 2008; Myers et al. 2006) Here new coordinates are introduced for each object *x*: Abscissa: Difference of objects in downset and upsets of *x*: $(\|O(x)\|-\|F(x)\|)$ Ordinate, number of objects incomparable with *x*, $\|U(x)\|$. Because $F(x)$ is used as symbol for upset(x), $O(x)$ as symbol for downset x, and $U(x)$ for the set of incomparable objects with x, the name FOU plot was used	See also Fig. 19.11
Rkav based on LPOM0	The local partial order model LPOM0 will be applied to get an approximation the average ranks	In LPOMext4.py a more sophisticated approximation is available, due the extended LPOM In avrank5.py the exact average rank is available when certain conditions are fulfilled
Save the different results	See also Fig. 19.5	
Open the control board for graphics	See also Fig. 19.5 to get an impression about the multitude how graphics can be manipulated. In parentheses the default values are shown	
Exit	It is important to exit the program in order to avoid damages	

Table 19.6 Structural information of the data matrix of Table 19.2, related with the Hasse diagram of Fig. 19.1

One linear extension
6 < 29 < 7 < 5 < 16 < 9 < 14 < 17 < 8
Maximal elements
8, 14, 17
Minimal elements
29, 6
Isolated elements
Individual info:
First the object, then in parentheses: count of, then the list of elements
Sets of incomparable elements
6: (1): 29
8: (2): 14, 17
7: (4): 29, 14, 17, 16
17: (7): 14, 16, 29, 5, 7, 9, 8
9: (5): 29, 5, 14, 17, 16
16: (5): 9, 5, 17, 7, 14
14: (6): 17, 16, 5, 7, 9, 8
5: (4): 9, 14, 17, 16
29: (4): 9, 17, 7, 6
Downsets
6: (1): 6
8: (7): 16, 29, 5, 7, 6, 9, 8
7: (2): 7, 6
17: (2): 17, 6
9: (3): 9, 7, 6
16: (3): 16, 29, 6
14: (3): 14, 29, 6
5: (4): 5, 29, 7, 6
29: (1): 29
Upsets
6: (8): 14, 17, 16, 5, 7, 6, 9, 8
8: (1): 8
7: (4): 9, 8, 5, 7
17: (1): 17
9: (2): 9, 8
16: (2): 8, 16
14: (1): 14
5: (2): 8, 5
29: (5): 8, 5, 14, 29, 16

Checking the data matrix one can see that indeed regions 8 and 14 are pretty different with respect to their data profile (for the sake of clarity, the min, and max values over all regions for each of the three attributes are additionally given):

	Pb	Cd	Zn
Max:	20	0.6	63
8:	20	0.4	55
14:	12	0.6	41
Min:	9	0.2	29


Info for selcted vertex

```
Detail information
===================
Selected object: 8
Incomparabilities of object 8 : 2.0
|upset| : (0 ):
|downset| : (6 ): 6, 7, 9, 16, 5, 29,
```

Info for selcted vertex

```
Detail information
===================
Selected object: 14
Incomparabilities of object 14 : 6.0
|upset| : (0 ):
|downset| : (2 ): 6, 29,
```

scatterplot O-F vs U

Alt-Mouse-button-1 for further information on points in FOU-plot

Fig. 19.6 FOU plot (see text) based on Table 19.2, using posetic coordinates. The greater *blue circles* are obtained by clicking with the mouse on them. The abscissa counts from −10 to +10 with steps of 0.5, the ordinate, however, counts from 0 to 10 with steps of 1

Region 8 is dominantly polluted by Lead and Zinc, whereas the main contribution of pollution of region 14 is Cadmium. The maximal and minimal values of Pb, Zn, and Cd taken over all objects of the data matrix (Table 19.2) are added to facilitate the interpretation.

The FOU plot is mainly useful for an interactive analysis and can be further explored using the mouse. So the FOU plot fulfills similar tasks as those, explained by Myers in this book (Myers and Patil 2014) There is an abscissa which describes the relative position on a bad–good axis and the ordinate which quantifies the conflicts associated with each object.

Clicking with the left mouse button, pessing “ALT” a window pops up with more information (Fig. 19.6, top, left side, and right side). Basically, depending on the ranking aim, the points near the lines given by (19.2a) and (19.2b)

$$|U(x)| = n+1-OF,\ OF = (|O(x)|-|F(x)|),\ |F(x)| = 0 \tag{19.2a}$$

and

$$|U(x)| = n+1+OF,\ OF = (|O(x)|-|F(x)|),\ |O(x)| = 0 \tag{19.2b}$$

are of most interest, as they are the extremal points.

In contrast to mainHD20_5.py, the module mHDCl2.py does no more contain the Bubley–Dyer algorithm (Bubley and Dyer 1999) to get average ranks (see Patil and Joshi 2014) and the statistics concerning chain length. The BubleyDyer algorithm is now the central part of the module BubleyDyer8.py where also the algorithm, proposed by Patil and Taillie (2004), the Cumulatice Rank Frequency (CRF) iterative method is provided. The CRF algorithm can be applied to enrich the poset until a weak order is obtained. See for details Chap. 6.

19.5.4.2 The Module to Check the Role of Single Indicator Values: POOC6.py

As mentioned by Annoni et al. (2011, 2012) and explained in more detail by Brüggemann and Patil (2011), there are two types of sensitivity analysis:

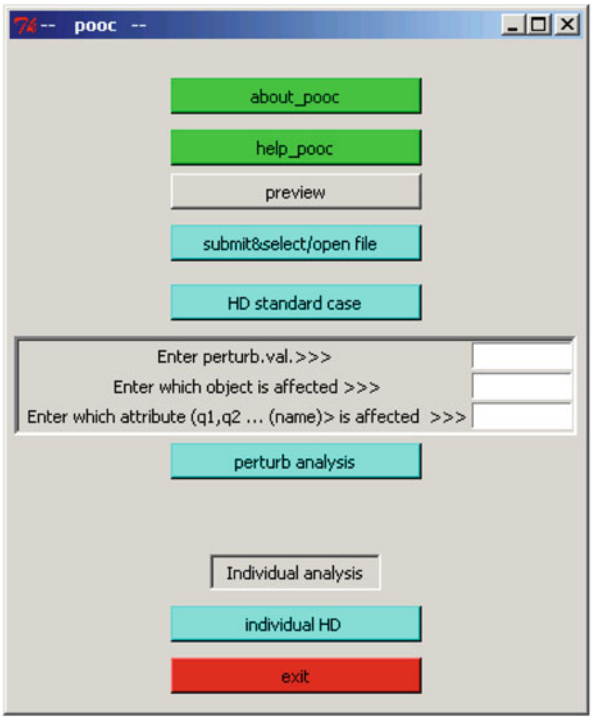
- Variation of the set of indicators, e.g., to elucidate the effect if one indicator is eliminated from the data matrix
- Variation of the values of indicators

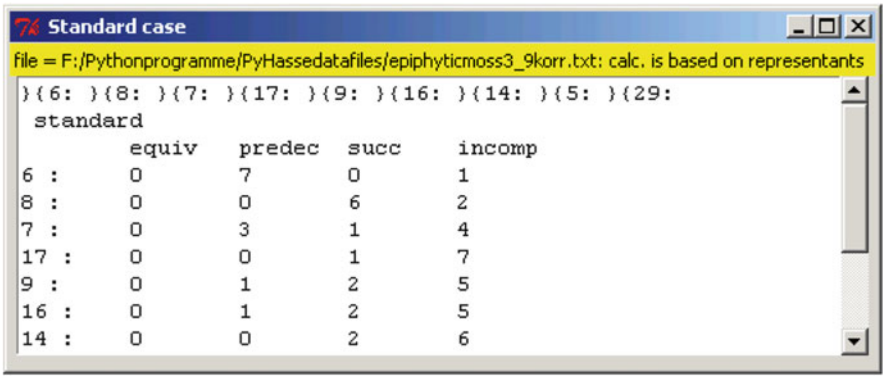
The first is referred to as attribute-related sensitivity (ARS), the second as attribute value-related sensitivity (AVRS). The ARS is the task of sensitivity18_3.py and is well described in the literature. Attribute value-related sensitivity is the task of POOC6.py (perturbation on order characteristics). With the new concept of variance-based sensitivity (Annoni et al. 2011, 2012; see also Chap. 13), the development concerning POOC6.py was slowed down. Nevertheless, this module appears mandatory, as long as the variance-based sensitivity is not programmed within PyHasse.

The GUI of POOC6.py is shown in Fig. 19.7.

After selecting the same data matrix as for Fig. 19.1, a posetic overview over the data matrix (Fig. 19.8) is first obtained, whereby now four coordinates are used.

Fig. 19.7 GUI of POOC6.py





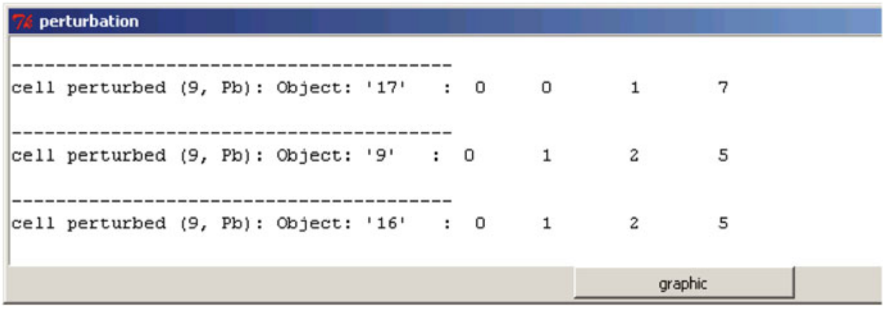
file = F:/Pythonprogramme/PyHassedatafiles/epiphyticmoss3_9korr.txt: calc. is based on representants

```
}{6: }{8: }{7: }{17: }{9: }{16: }{14: }{5: }{29:
 standard
        equiv    predec   succ     incomp
6 :      0        7        0        1
8 :      0        0        6        2
7 :      0        3        1        4
17 :     0        0        1        7
9 :      0        1        2        5
16 :     0        1        2        5
14 :     0        0        2        6
```

Fig. 19.8 Posetic "coordinates" (equiv, predec, succ, and incomp) of the data matrix, describing metal pollution of epiphytic mosses, in south-west of Germany (cf. Table 19.2)

The coordinates are:

- *equiv* (*eq*): number of equivalent elements with *x*
- *predec* (*pred*): number of elements above *x*, *pred* =$|O(x)\text{-}\{x\}|$
- *succ*: number of elements below *x*, *succ* = $|F(x)\text{-}\{x\}|$
- *incomp* (*ic*): number of elements incomparable with *x*, *ic*= $|U(x)|$



```
----------------------------------------
cell perturbed (9, Pb): Object: '17'   :  0     0       1       7

----------------------------------------
cell perturbed (9, Pb): Object: '9'   :  0      1       2       5

----------------------------------------
cell perturbed (9, Pb): Object: '16'   :  0     1       2       5
```

Fig. 19.9 Posetic coordinates, after perturbing the value of attribute Pb of region 9 by changing the attribute by adding the value of 1

Thus, as an example, select as element of interest region 9, one sees that there is no element, equivalent with region 9, there is one element above 9, two elements below region 9 and 5 regions which are not comparable to region 9.

The role of pooc6.py is now to check how a change in an attribute value will change the set of posetic coordinates.

We enter a perturbing value 1, select object 9 and attribute “Pb,” i.e., pollution of lead in the epiphytic moss. Note that clearly a perturbing value for one of the columns of the data matrix is specific, for instance, 1 for Pb concentration is a small value, 1 for Cd pollution would be larger than the whole span of Cd values! Here we perturb Pb by 1/20 of the maximal value.

Technically a perturbation by 1 means that we change the original entry *q1(9)* by adding 1, i.e., changing the value from 17 to 18, and thus observe the possible effects (Fig. 19.9).

To the most left side: the site of perturbation and the perturbed indicator are explained, then information is given how the different elements of the poset are reacting.

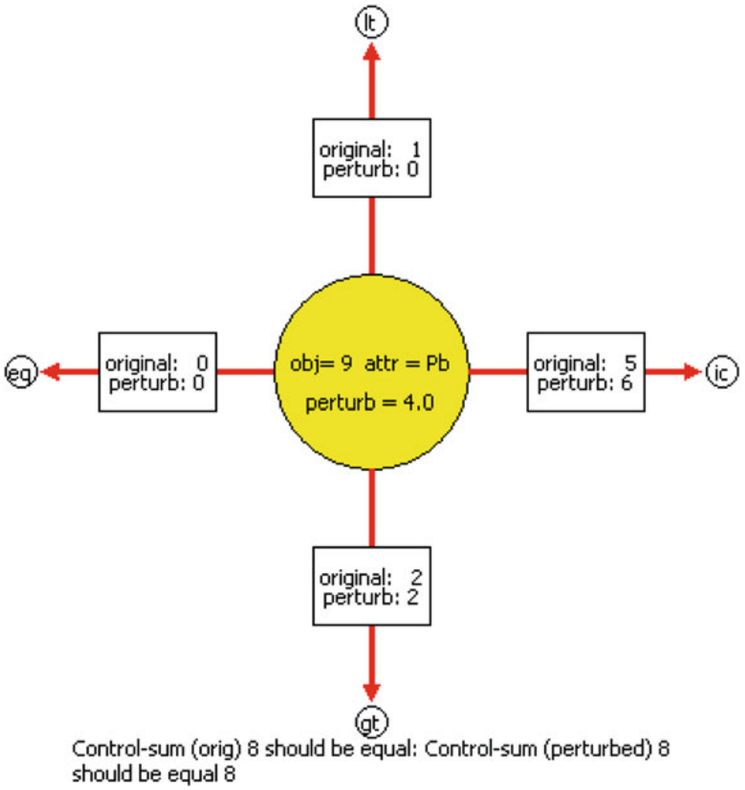
A perturbation by adding 4 to the original value, i.e., 20 % of the maximum of lead concentrations with the regions considered, changes the coordinates.

	eq	predec	succ	incomp
(Perturbed) (9, Pb): Obj: 9:	0	0	2	6

The number of predecessors of region 9 would in this case be reduced and the number of incomparable elements with region 9 increased.

We could conclude that the posetic information concerning region 9 is rather stable with respect to increasing the value of Pb. Clearly this procedure can be repeated for every element of interest, every attribute, and with every perturbing value.

In Fig. 19.10 a graphical display on what happens after perturbing the value of Pb for region 9 by 4 is given (in terms of region 9 less than (lt), greater then (gt), incomparable with (ic), and equivalent with (eq)).



Control-sum (orig) 8 should be equal: Control-sum (perturbed) 8
should be equal 8

Fig. 19.10 Schematic overview about the four posetic coordinates: eq: number of equivalences, ic: number of incomparabilities, gt: number of predecessors (greater), and lt: number of successors (less than) after changing the attribute Pb of region 9 by four units

Figure 19.10 deserves some further explanations: The large yellow circle informs about the perturbation itself.

The affected object (obj) is region 9, the attribute (attr) perturbed is Pb, and the amount of perturbation (perturb) is 4.0. In the little white circle the four posetic coordinates are indicated and in the rectangular box an information is given, what happens with respect to this specific coordinate: For example, the value of "lt" does not changed, i.e., it is not perturbed (perturbed value of lt:=0). In contrast, the coordinate "ic" changes by perturbation. The original value is 5, after perturbation this value changed to 6. (perturb (of ic)): 6.

Figures 19.8, 19.9, and 19.10 are the result of POOC6.py, which in turn is designed to help to find answers concerning the Hasse diagram in Sect. 19.4, namely the effect of data uncertainty. Additionally, however not shown, any perturbed data matrix can be visualized by a Hasse diagram.

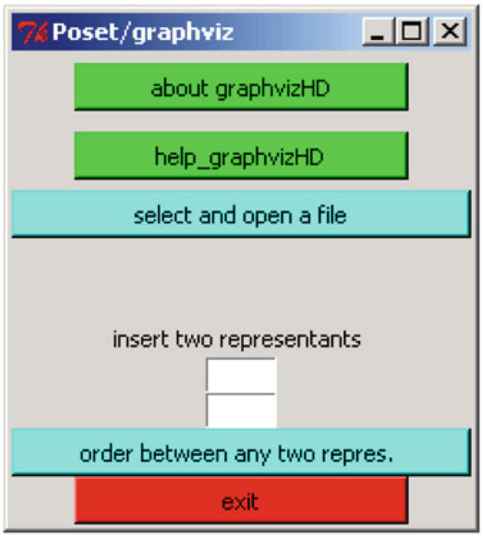


Fig. 19.11 GUI of graphvizHD1

19.5.4.3 Module graphvizHD1.py

Introduction

This module serves as a nice example for the general philosophy in the context of PyHasse, i.e., not to compete with professional software, if available. In the module graphvizHD1.py, some information on the partially ordered set is given. However, the graph drawing is a matter of the well-known graph–theoretical program Graphviz (Gansner and North 1999), which is explained below. In Fig. 19.11 the GUI is shown.

As for other modules, about and help functions are found. Behind the button "select and open a file," the facilities of the Tkinter library are applied.

After the selection of the data file, a window pops up with more information (see Table 19.7).

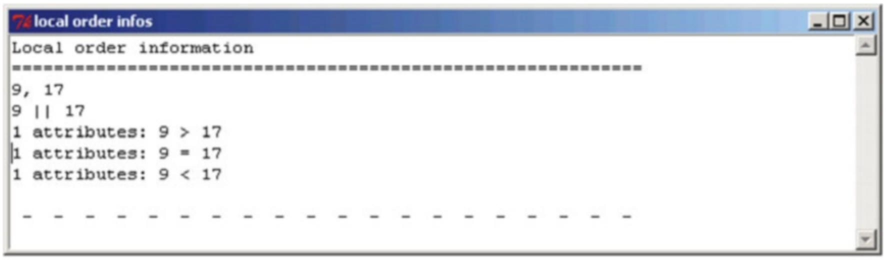
It is seen that local information (i.e., information not related to the complete object set, but to a user selected pair of objects) is available by inserting objects into the two open entry fields. Inserting for example "9" and "17," two objects of the data matrix of the selected example file, an information is obtained: a) comparable or not and b) in which orientation the two regions are comparable. Here it is found: 9 || 17, see also Fig. 19.12.

For a deeper analysis procedure, the concept of "distance due to incomparability" (Bartel and Mucha, Chap. 3) may be applied.

Further, a window is opened to select name and site of the file, subsequently to be analyzed by graphviz.

Table 19.7 Content of the window, popping up after selecting and opening the file (containing the data matrix about pollution in epiphytic mosses) (and doing subsequently all needed calculations to obtain the partial order)

Info about poset of F:/Pythonprogramme/PyHassedatafiles/epiphyticmoss3_9korr.txt
1. General info
Objects (representants)
6, 8, 7, 17, 9, 16, 14, 5, 29
Properties (indicators, attributes)
Pb, Cd, Zn
2. Posetic info
Number of levels (= length of maxim. chains) = 4
Number of elements in largest level = 3
Comparabilities = 17
Incomparabilities = 19.0
Count of maximal elements = 3
Maximal elements
8, 17, 14
Count of minimal elements = 2
Minimal elements
6, 29
Count of isolated objects = 0
Isolated objects



```
Local order information
==============================================================
9, 17
9 || 17
1 attributes: 9 > 17
1 attributes: 9 = 17
1 attributes: 9 < 17

 - - - - - - - - - - - - - - - - - - - -
```

Fig. 19.12 Local information about a pair of objects, here about a pair of regions

Graphviz

Graphviz is a professional program to draw graphs, i.e., visualize binary relations on a ground set. Graphviz draws binary relations of the object set, or more exactly, of the set of representatives. The software Graphviz is freely downloadable from the Internet and is described by Gansner et al. (1993) and Gansner and North (1999).

The version used here is 2.26.3, and among the programs available in Graphviz, the program Gvedit, v: 1.01 is used.

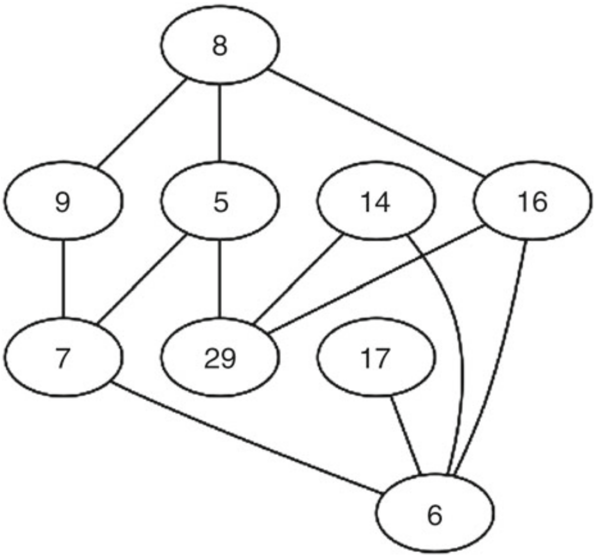


Fig. 19.13 Result after running Gvedit: a gif File

By successful running Gvedit, for instance, a gif File is obtained (see Fig. 19.13), many other formats are available too. It represents the same order relation as in Fig. 19.1. However, the drawing rules in Graphviz are dominated by minimizing the crossings of lines. Gvedit allows many controlling interactions by the user: However, for most purposes those specifications are not needed. A more detailed description of the graphviz option is outside the scope of the present chapter.

19.6 Summary and Conclusions

19.6.1 Summary

When a data matrix is to be analyzed with respect to some ranking or evaluation, then usually one has to select a software. Whereas the construction of a composite indicator is simple and can be done with spreadsheet facilities, like MS Excel, the analysis within partial order methodology can in general not be done by spreadsheet software.

With the example of a small real-life data matrix, where the regional pollution is measured in the special target of epiphytic the technical performance by some modules of PyHasse are demonstrated. There are PyHasse modules available, which unequivocallly are important and often used, for example, mainHD20_5.py, mHDCL2.py, similarity10_1, or sensitivity18_3, LPOM4ext.py, and dds12.py. Some others are described and follow crudely the logical line:

1. What information is provided by the Hasse diagram? (mHDCl2_7.py, Sect. 19.5.4.1)?
2. What happens when data entries are changed? (POOC6.py, Sect. 19.5.4.2)
3. Graphical display in the form of Hasse diagrams is appealing. However, the display of partial orders allows many freedoms. In PyHasse firstly a conservative point of view is taken, i.e., to locate the objects in the highest position, which is order theoretically possible. Secondly the objects are arranged in levels. These two principles often lead, in the graphical presentation of the results (the Hasse diagram), to crossing of lines, which may be rather confusing. Thus, an alternative is discussed, and the use of the freely downloadable software Graphviz is suggested (Sect. 19.5.4.3).

19.6.2 Conclusions

PyHasse is today applied by many teams around the world. It is clear that correspondingly many ideas are expressed how PyHasse can be improved

- In its technical handling
- Contextually, in its tools to ordinally analyze data matrices

PyHasse is not claimed as a user-friendly software with a good guidance of the users. However, it should be clear that PyHasse does not want to (and cannot) compete with, for instance, DART (Manganaro et al. 2008), which provides a very convenient tool to get Hasse diagrams as well as some basic information derived from a data matrix—even with missing data. The application of PyHasse needs some preparatory steps in data handling before it can be run. It also most often needs an a posteriori activity by the user. This is the consequence of the conception behind PyHasse, to help specifically in studies of partial ordering, i.e., in all consequences which arise from the ordinal analysis of multi indicator systems. PyHasse provides copy-and-paste texts to support the documentation of results.

When graphical representations are available (bar diagrams, Hasse diagrams, scatter plots, etc.), their purpose is to give the user a first impression, and when the user wants a professional graphic software, such as Excel, then some few steps of data handling are necessary.

PyHasse is rapidly developing as ideas from users as well as concepts from the literature relatively easily may be programmed leading to new modules. The price is that the total absence of bugs cannot be guaranteed, albeit most modules are tested rather carefully. Futher the user interfaces may not always be as comfortable as possibly desirable and philosophies how to guide the user are only rudimentarily realized. PyHasse is an "experimental" software under constant development and suggestions, comments, and wishes from users are always welcome and appreciated.

19.7 Outlook

For the time being, eight major objectives are still on the agenda. However, obviously time is required and the development further constantly compete with other more rapidly realizable ideas. The eight future objective can be summarized as:

1. PyHasse being made available in an Internet version. Some preliminary attempts have been made. However, the cooperation with web designers, etc., appears crucial.
2. Although the powerful conexp3, written in Java is available for analysis of Formal Concepts (Yevtushenko 2003; Ganter and Wille 1996, Burmeister 2003) a Formal-Concept-Analysis-module within PyHasse would facilitate many applications.
3. POSAC is a program performing a reduction of the attributes of the data matrix to two coordinates. The underlying idea is to maintain the typical outcome of partial order theory, i.e., the appearance of incomparabilities but at the same time simplifiying the analysis. POSAC is an approximation. Nevertheless a, possibly simplified version in PyHasse would be helpful (see Brüggemann and Patil 2011 and references therein).
4. When a reduction to two new attributes as in POSAC is intended, a calculation of the poset dimension would be useful. However, the calculation of the dimension of a poset is computationally extremely difficult. Nevertheless, it is important to get ideas about the dimension of posets.
5. The variance-based sensitivity analysis is most urgently needed as an implementation in PyHasse. So far, the needed calculations are performed using Matlab. Consequently, the extensive numerical part should be programmed in C++ or at least by including the library NumPy.
6. In multivariate statistics, cluster analysis plays an important role. A straightforward application of cluster analysis is suitable in order to get clear Hasse diagrams by reducing the number of vertices. This reduction can be done in the form of deriving a poset on cluster centers instead on the single objects. This reduction is the main feature of the PyHasse module pycluster1_2.py. However, an order theoretical approach would be helpful too: Instead of defining equivalence relations such as "belonging to the same cluster," one could appropriately define equivalence relations among the elements of a poset, also called "blocks" (Davey and Priestley 1990) and analyze the resulting posets based on the representative elements, which clearly is simpler than the original poset.
7. Finally, a project aiming at extending PyHasse by an additional fuzzy-poset analysis is in progress. A first variant is provided in fuzzydds7.py. Now an intensive testing phase is needed.
8. The further analysis of the two approximations of average ranks based on local partial order model is a task for the future. It is hoped to give improved statements about the accuracy of the LPOM model.

19.8 (a) List of Abbreviations (Alphabetically Sorted)

Abbreviation	Meaning
AC, ac	Antichain
ACM	Antichain matrix
ARS	Attribute-related sensitivity
AVRS	Attribute value-related sensitivity
CompInd	Composite indicators
DART	Decision analysis by ranking techniques
dm	Data matrix
equiv(eq)	Number of equivalent elements of a certain object
FCA	Formal concept analysis
FOU	Plot derived from $\|F(x)\|$, $\|O(x)\|$, $\|U(x)\|$ (contents of upset of x, downset of x, incomparables with x)
GUI	Graphical user interface
HD, hd	Hasse diagram
HDT	Hasse diagram technique
IB	Information base (set of attributes of a certain ranking study)
incomp(ic)	Number of incomparable elements of an object x
LinExt	Linear extensions
LPOM	Local partial order model
LPOM0	LPOM, based on the simplest approximation
LPOMext	LPOM, based on an extended method
MAC	Macintosh
MCDA	Multicriteria decision analysis
METEOR	Method of evaluation by order theory
OS	Operating system
Parsec	Partial orders in Socioeconomics
POSAC	Partial order scalogram with coordinates
poset	Partially ordered set
POT	Partial order theory
predec(pred)	Number of predecessors of a certain object
Rapid	Ranking and prioritization information delivery
Rkav	Average height, commonly called average rank
succ	Number of successors of a certain object
VHLL	Very high-level language

19.9 (b) Further Recommended References Within the Context of PyHasse and HDT

Brüggemann R, Pudenz S, Voigt K, Kaune A, Kreimes K (1999) An algebraic/graphical tool to compare ecosystems with respect to their pollution. IV: comparative regional analysis by Boolean arithmetics. Chemosphere 38:2263–2279

Brüggemann R, Voigt K, Restrepo G, Simon U (2008) The concept of stability fields and hot spots in ranking of environmental chemicals. Environ Model Softw 23:1000–1012

Brüggemann R, Kerber A, Restrepo G (2011) Ranking objects using fuzzy orders, with an application to refrigerants. Match Commun Math Comput Chem 66(2):581–603

Carlsen L, Brüggemann R (2009) Partial order ranking as a tool in environmental impact assessment. In: Halley GT, Fridian YT (eds) PAH and PCB pollution of the River Main as an illustrative example. . environmental impact assessment. Nova Science Publishers, pp 335–354

Carlsen L, Brüggemann R (2011) Risk assessment of chemicals in the River Main (Germany): application of selected partial order ranking tools. Statistica and Applicazioni, special issue 125–140

De Loof K, De Meyer H, De Baets B (2006) Exploiting the lattice of ideals representation of a poset. Fundamenta Informaticae 71:309–321

De Loof K, De Baets B, De Meyer H, Brüggemann R (2008) A Hitchhiker's guide to poset ranking. Comb Chem High Throughput Screen 11:734–744 (3-3)

De Loof K, De Baets B, De Meyer H (2011) Approximation of average ranks in posets. Match Commun Math Comput Chem 66:219–229

Restrepo G, Brüggemann R (2008) Dominance and separability in posets, their application to isoelectronic species with equal total charge. J Math Chem 44: 577–602

Restrepo G, Weckert M, Brüggemann R, Gerstmann S, Frank H (2008) Ranking of refrigerants. Environ Sci Technol 42:2925–2930 (3-8)

Sailaukhanuly Y, Zhakupbekova A, Amutova F, Carlsen L (2013) On the ranking of chemicals based on their PBT characteristics: comparison of different ranking methodologies using selected POPs as an illustrative example. Chemosphere 90:112–117

Simon U, Brüggemann R, Mey S, Pudenz S (2005) METEOR – application of a decision support tool based on discrete mathematics. Match Commun Math Comput Chem 54:623–642

Simon U, Brüggemann R, Behrendt H, Shulenberger E, Pudenz S (2006) METEOR: a step-by-step procedure to explore effects of indicator aggregation in multi criteria decision aiding – application to water management in Berlin, Germany. Acta Hydrochim Hydrobiol 34:126–136

Tsakovski S, Simeonov V (2011) Hasse diagram technique as exploratory tool in sediment pollution assessment. J Chemometrics. doi:10.1002/cem.1381:1-8

Tsakovski S, Kudlak B, Simeonov V, Wolska L, Garcia G, Namiesnik J (2012) Relationship between heavy metal distribution in sediment samples and their ecotoxicity by the use of the Hasse diagram technique. Analytica Chimica Acta. doi: http:///10.1016/j.aca.2011.12.052

Voigt K, Brüggemann R, Scherb H, Shen H, Schramm K-H (2010a) Evaluating the relationship between chemical exposure and cryptorchidism by discrete mathematical method using PyHasse software. Environ Model Softw 25:1801–1812

Voigt K, Brüggemann R, Kirchner M, Schramm K-W (2010b) Influence of altitude concerning the contamination of humus soils in the German Alps: a data evaluation approach using PyHasse. Environ Sci Pollut Res 17:429–440

References

Al-Sharrah G (2010) Ranking using the Copeland score: a comparsion with the Hasse diagram. J Chem Inf Model 50:785–791

Al-Sharrah G (2011) The Copeland method as a relative and categorized ranking tool. Stastidtica et Applicazioni, special issue, 81–95

Annoni P (2007) Different ranking methods: potentialities and pitfalls for the case of European opinion poll. Environ Ecol Stat 14:453–471

Annoni P, Brüggemann R, Saltelli A (2011) Partial order investigation of multiple indicator systems using variance – based sensitivity analysis. Environ Model Softw 26:950–958

Annoni P, Brüggemann R, Saltelli A (2012) Random and quasi-random designs in variance-based sensitivity analysis for partially ordered sets. Reliab Eng Syst Saf 107:184–189

Bartel H-G, Brüggemann R (1998) Application of formal concept analysis to structure-activity relationships. Fresenius J Anal Chem 361:23–28

Birkhoff G (1984) Lattice theory, vol XXV. American Mathematical Society, Providence, RI

Brans JP, Vincke PH (1985) A preference ranking organisation method (The PROMETHEE method for multiple criteria decision – making). Manag Sci 31:647–656

Brüggemann R, Carlsen L (2011) An improved estimation of averaged ranks of partial orders. Match Commun Math Comput Chem 65(2):383–414

Brüggemann R, Carlsen L (2012) Multi-criteria decision analyses. Viewing MCDA in terms of both process and aggregation methods: Some thoughts, motivated by the paper of Huang, Keisler and Linkov. Sci Total Environ 425:293–295

Brüggemann R, Patil GP (2010) Multicriteria prioritization and partial order in environmental sciences. Environ Ecol Stat 17:383–410

Brüggemann R, Patil GP (2011) Ranking and prioritization for multi-indicator systems – introduction to partial order applications. Springer, New York, NY

Brüggemann R, Voigt K (2008) Basic principles of Hasse diagram technique in chemistry. Comb Chem High Throughput Screen 11:756–769

Brüggemann R, Sørensen PB, Lerche D, Carlsen L (2004) Estimation of averaged ranks by a local partial order model. J Chem Inf Comput Sci 44:618–625

Brüggemann R, Voigt K, Kaune A, Pudenz S, Komossa D, Friedrich J (1998) Vergleichende ökologische Bewertung von Regionen in Baden- Württemberg GSF-Bericht 20/98. GSF, Neuherberg

Brüggemann R, Bücherl C, Pudenz S, Steinberg C (1999) Application of the concept of partial order on comparative evaluation of environmental chemicals. Acta Hydrochim Hydrobiol 27:170–178

Brüggemann R, Halfon E, Welzl G, Voigt K, Steinberg C (2001) Applying the concept of partially ordered sets on the ranking of near-shore sediments by a battery of tests. J Chem Inf Comput Sci 41:918–925

Bubley R, Dyer M (1999) Faster random generation of linear extensions. Discrete Math 201: 81–88

Burmeister P, CONIMP4, Programm zur Formalen Begriffsanalyse (1997) Technische Hochschule Darmstadt, Arbeitsgruppe 1, Fachbereich 4 (Mathematik) WWW-Adresse: http://www.mathematik.tu-darmstadt.de/~burmeister/ConImpIntro.pdf (last access August, 2013)

Carlsen L (2008) Hierarchical partial order ranking. Environ Pollut 155:247–253

Carlsen L (2009) The interplay between QSAR/QSPR studies and partial order ranking and formal concept analyses. Int J Mol Sci 10:1628–1657

Carlsen L, Brüggemann R (2013a) Indicator analyses, what is important: and for what? In: Brüggemann R, Carlsen L, Wittmann J (eds) Multi-indicator systems and modelling in partial order, Chap 18. Springer, New York, NY

Carlsen L, Brüggemann R (2013b) The 'Failed Nations Index' offers more than just a simple Ranking. Soc Indic Res. doi:10.1007/s11205-012-9999-6

Davey BA (2004) Formal concept analysis. In: Eklund P (ed) ICFCA 2004, LNAI 2961. Springer, Berlin, pp 55–56

Davey BA, Priestley HA (1990) Introduction to lattices and order. Cambridge University Press, Cambridge

DEA. http://mat.gsia.cmu.edu/classes/QUANT/NOTES/chap12.pdf. Assessed 16 Oct 2012

Gansner ER, North SC (1999) An open graph visualization system and its applications to software engineering. Softw Pract Exp 30(11):1203–1233

Gansner ER, Koutsofios E, North SC, Vo K-P (1993) A technique for drawing directed graphs. IEEE Trans Softw Eng 19:214–230

Ganter B, Wille R (1996) Formale Begriffsanalyse Mathematische Grundlagen. Springer, Berlin

Halfon E (2006) Hasse diagrams and software development. In: Brüggemann R, Carlsen L (eds) Partial order in environmental sciences and chemistry. Springer, Berlin, pp 385–392

Halfon E, Reggiani MG (1986) On ranking chemicals for environmental hazard. Environ Sci Technol 20:1173–1179

Halfon E, Hodson J, Miles K (1986) An algorithm to plot Hasse diagrams on microcomputers and Calcomp plotters. Ecol Model 47:189–197

Hans Petter Langtangen (2009) A primer on scientific programming with Python (Texts in Computational Science and Engineering). Springer, Auflage: 1 (4 Aug 2009)

Hasse H (1927) Höhere Algebra II Gleichungen höheren Grades. Walter De Gruyter, vormals G.J. Göschen'sche Vetrlagshandlung, Berlin und Leipzig

Hasse H (1952) Über die Klassenzahl abelscher Zahlkörper. Akademie Verlag, Berlin

Huang IB, Keisler J, Linkov I (2011) Multi-criteria decision analysis in environmental sciences: ten years of applications and trends. Sci Total Environ 409:3578–3594

Lutz M, Ascher D (2003) Learning Python. O'Reilly, Beijing

Manganaro A, Ballabio D, Consonni V, Mauri A, Pavan M, Todeschini R (2008) The DART (Decision Analysis by Ranking Techniques) software. In: Pavan M, Todeschini R (eds) Scientific data ranking methods: theory and applications. Elsevier, Amsterdam, pp 193–207

Müller M, St. Schwarzer (2007) Python im deutschsprachigen Raum, Tagungsband zum Workshop am 8 September 2006 in Leipzig. Lehmanns Media, Berlin

Munda G (2008) Social multi-criteria evaluation for a sustainable economy. Springer, Berlin

Myers WL, Patil GP (2008) Semi-subordination sequences in multi-measure prioritization problems. In: Todeschini R, Pavan M (eds) Data handling in science and technology, vol 27. Elsevier, New York, NY, pp 161–170

Myers WL, Patil GP (2010) Preliminary prioritization based on partial order theory and R software for compositional complexes in landscape ecology, with applications to restoration, remediation, and enhancement. Environ Ecol Stat 17:411–436

Myers WL, Patil GP (2014) Higher-order indicator with rank-related clustering in multi-indicator systems. In: Brüggemann R, Carlsen L, Wittmann J (eds) Multi-indicator systems and modelling in partial order. Springer, New York, NY

Myers WL, Patil GP, Cai Y (2006) Exploring patterns of habitat diversity across landscapes using partial ordering. In: Brüggemann R, Carlsen L (eds) Partial order in environmental sciences and chemistry. Springer, Berlin, pp 309–325

Opperhuizen A, Hutzinger O (1982) Multi-criteria analysis and risk assessment. Chemosphere 11:675–678

Patil GP, Joshi S (2014) Comparative knowledge discovery with partial order and composite indicator. In: Brüggemann R, Carlsen L, Wittmann J (eds) Multi-indicator systems and modelling in partial order. Springer, New York, NY

Patil GP, Taillie C (1976) Ecological diversity: concepts, indices and applications. In: The Biometric Society (ed) Proceedings of the 9th biometric conference, vol II, Boston, The Biometric Society, Boston, MA, pp 383–411, 22–27 Aug 1976

Patil GP, Taillie C (2004) Multiple indicators, partially ordered sets, and linear extensions: multi-criterion ranking and prioritization. Environ Ecol Stat 11:199–228

Peters ML, Zelewski S (2007) TOPSIS als Technik zur Effizienzanalyse. WiSt January 2007:9–15

Pudenz S (2005) ProRank – software for partial order ranking. Match Commun Math Comput Chem 54:611–622

Saaty TL (1994) How to make a decision: the analytical hierarchy process. Interfaces 24:19–43

Thiessen RJ, Achari G (2012) Can the national classification system for contaminated sites be used to rank sites. Can J Civ Eng 39:415–431

Van de Walle B, De Baets B, Kersebaum KC (1995) Fuzzy multi-criteria analysis of cutting techniques in a nuclear dismantling project. Fuzzy Set Syst 74:115–126

Venners B (2003) The making of, Python, a conversation with Guido van Rossum, Part I. http://www.artima.com/intv/python.html. Assessed 20 Sept 2012

Voigt K, Scherb H, Brüggemann R, Schramm K-W (2011) Application of the PyHasse program features: sensitivity, similarity, and separability for environmental health data. Statistica and Applicazioni, special issue 155–168

Voigt K, Brüggemann R, Scherb H, Cok I, Mazmanci B, Mazmanci MA, Turgut C, Schramm K-W (2012) Evaluation of organochlorine pesticides in breast milk samples in Turkey applying features of the partial order technique. Int J Environ Health Res. doi:10.1080/09603123.2012.71915

Wieland R, Brüggemann R (2013) Hasse Diagram technique and Monte Carlo simulations. Match Commun Math Comput Chem 70:45–59

Wienand O (2012) http://bio.math.berkeley.edu/ranktests/lcell/. Assessed 6 Nov 6

Winkler P (1982) Average height in a partially ordered set. Discrete Math 39:337–341

Yager RR (1988) On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Trans Syst Man Cybern 18:183–190

Yager RR (1993) Families of OWA operators. Fuzzy Set Syst 59:125–148

Yevtushenko SA (2003) Concept explorer the user guide: system of data analysis. Concept Explorer 127–134