

University of Passau
Department of Informatics and Mathematics



Bachelor Thesis

Visualizing General Morphological Analysis via Multidimensional Scaling

Author:

Julius Kempf

September 21, 2015

Advisors:

Prof. Dr.-Ing. Sven Apel
Chair of Software Product Lines

Dr. Norbert Siegmund
Chair of Software Product Lines

Kempf, Julius:

Visualizing General Morphological Analysis via Multidimensional Scaling

Bachelor Thesis, University of Passau, 2015.

Abstract

The goal of this thesis is to give a thorough overview of the field of General Morphological Analysis (GMA) and of possibilities for GMA computer support. The state of the art of the conceptual approaches is presented and some suggestions to improve the procedure are developed. Then the thesis turns on GMA computer support. Two already existing approaches are shortly presented and then the functions of two GMA-supporting programs are outlined that have been developed for the purpose of this thesis. The focus is on GMAvisual, a program that makes use of Multidimensional Scaling (MDS) to visualize the GMA solution space. The goal is to give the user an impression of the structure of the solution space and possibly of the inherent clusters. To this purpose, the mathematical backgrounds of classical MDS and of similarity measures for categorical data are shortly outlined and included.

Contents

List of Figures	viii
List of Tables	ix
1 Introduction	1
2 General Morphological Analysis (GMA)	3
2.1 The GMA procedure	4
2.2 Consistency assessment - known approaches	7
2.2.1 Binary pairwise consistency assessment	7
2.2.2 Scaled pairwise consistency assessment	9
3 Possible improvements of the consistency assessment	11
3.1 Binary pairwise consistency assessment - effectiveness	11
3.2 Higher order consistency assessment	12
3.3 Selective triple consistency assessment (TCA)	15
3.4 Scaled consistency assessment	18
3.4.1 Calculating the consistency value of a configuration	19
3.4.2 Equidistant consistency calculation	20
3.4.3 Weighted consistency calculation	23
3.4.4 Reflections on the weighting function	23
3.5 Scaled consistency assessment and selective TCA	25
4 Computer support for GMA - known approaches	29
4.1 MA/Carma	29
4.2 Parmenides EIDOS	33
4.2.1 Classical functions and sorted configurations	33
4.2.2 Cluster visualization of the configuration space	37
5 SIM - an inference model with scaled consistency data	39
6 Multidimensional Scaling (MDS) for GMA - theoretical backgrounds	45
6.1 Classical Multidimensional Scaling	46
6.1.1 Calculating distances from coordinates	46

6.1.2	Calculating coordinates from distances	47
6.2	Similarity measures for categorical data	49
7	GMAvisual - an MDS prototype for GMA	53
7.1	Functions of GMAvisual by control device areas	55
7.1.1	Option area	56
7.1.2	Parameter area	58
7.1.3	Visualization area	62
7.1.4	Explore area	64
7.2	MDS with different similarity measures	69
7.3	Other scaling algorithms - comparing MDS to t-SNE	72
8	Conclusion	75
	Bibliography	77

List of Figures

2.1	Morphological Field of the Holiday Trip problem	6
2.2	Cross Consistency Matrix of the Holiday Trip problem (binary)	8
3.1	Percentage of reduced configurations in dependence of the percentage of inconsistent value pairs	12
3.2	Number of x -tuples depending on the respective tuple size x	14
3.3	Number of x -tuples depending on the respective tuple size x (incl. numbers of reduction by inconsistent tuples) - example cases	15
3.4	Selective triple consistency assessment for the Holiday Trip problem	17
3.5	Cross Consistency Matrix of the Holiday Trip problem (scaled)	20
3.6	Calculating consistency values of configurations - frequency distributions of the Holiday Trip problem (equidistant vs. weighted approach)	22
3.7	Frequency distributions of configurations containing at least one inconsistent triple (equidistant vs. weighted approach, Holiday Trip problem)	26
3.8	Consistency calculation with triple correction factor (Holiday Trip problem)	27
4.1	Morphological field in MA/Carma (adapted from [Rit03])	30
4.2	Cross consistency matrix in MA/Carma (adapted from [Rit03])	31
4.3	Display Field in MA/Carma (adapted from [Rit03])	32
4.4	Morphological field in Parmenides EIDOS	35
4.5	Cross consistency matrix in Parmenides EIDOS	35
4.6	Sorted configuration space in Parmenides EIDOS (1)	36
4.7	Sorted configuration space in Parmenides EIDOS (2)	36
4.8	Cluster View in Parmenides EIDOS (1)	38

4.9	Cluster View in Parmenides EIDOS (2)	38
5.1	Scaled Inference Model on the full configuration space (1)	41
5.2	Scaled Inference Model on the full configuration space (2)	41
5.3	Scaled Inference Model on the full configuration space (3)	42
5.4	Scaled Inference Model on the reduced configuration space	42
5.5	Scaled Inference Model with the equidistant weighting function	43
5.6	Scaled Inference Model with relative value frequencies	43
7.1	Cross consistency assessment in GMAvisual	54
7.2	List of available projects in GMAvisual	54
7.3	Visual frame in GMAvisual	55
7.4	GMAvisual: Adjusting the number of shown configurations	57
7.5	GMAvisual: Adjusting the size of the configuration bubbles	58
7.6	GMAvisual: Fixing parameter values	59
7.7	GMAvisual: Coloring and shaping of parameter values	60
7.8	GMAvisual: Combination of coloring and shaping	61
7.9	GMAvisual: Exploring three dimensions	62
7.10	GMAvisual: Tooltips and color modes	63
7.11	GMAvisual: Zooming	63
7.12	GMAvisual: Explore area - selection of a single configuration	64
7.13	GMAvisual: Explore area - selection of a group of configurations	65
7.14	GMAvisual: Explore area - caching selections	66
7.15	GMAvisual: Open selection in new window	67
7.16	GMAvisual: Automatic clustering support	68
7.17	GMAvisual: MDS with different similarity measures (1)	71
7.18	GMAvisual: MDS with different similarity measures (2)	72
7.19	GMAvisual: Alternative scaling algorithms - t-SNE	73

List of Tables

2.1	Cuttings of the formal configuration space (Holiday Trip problem)	6
3.1	Selected examples of inconsistent triples (Holiday Trip problem)	18
3.2	Sorted configuration space, best 15 configurations (Holiday Trip problem)	28

1. Introduction

General Morphological Analysis is a popular creativity technique and a problem solving method. The concept is very simple, but nonetheless pretty powerful. Few scientists work on GMA, its use and possible enhancements, and so there are few scientific publications about it. Tom Ritchey of the Swedish Morphological Society is the most active in the field.

This thesis summarizes and presents the GMA procedure and the known approaches for the consistency assessment part. Then it examines some of the mathematical backgrounds of the consistency assessment and makes suggestions on how to improve it.

As GMA itself is rather simple but implies combinatorial calculations that are most of the time too time-consuming to be done by hand, it's all about computer support: computer support for the GMA-inherent calculations - but also computer support as additional step in the GMA process. The output of GMA is a (possibly very large) set of categorical vectors. This set, the so-called solution space, is to be structured and presented to the user of GMA software in a way that he can grasp as much information as possible. Two programs that implement GMA and additional GMA computer support (MA/Carma and Parmenides EIDOS) are presented and their functions briefly outlined.

The motivation for this thesis was, not only to thoroughly present the state of the art, but to develop new approaches for additional GMA computer support. Two small programs have been implemented to this purpose. One, the Scaled Inference Model (SIM), just enhances the concept of MA/Carma with scaled consistency data. The other one, GMAvisual, is the heart of this thesis. It brings together GMA and Multidimensional Scaling (MDS), an algorithm that calculates (two-dimensional) coordinates from distances. Applying MDS to the solution space allows to visualize the categorical vectors of the solution space as points on a two-dimensional plane. That way a visual clustering is effected: similar elements of the solution space will be placed closer to each other than dissimilar elements. The arising clusters give a notion into which solution categories the solution space could maybe be partitioned. Such a partitioning would be a great enhancement to the GMA procedure.

GMAvisual offers various functions to interactively explore the MDS visualizations (and so the GMA solution space) - these are thoroughly explained using screenshots of the program. Before that, the mathematical backgrounds of MDS and similarity measures for categorical data are briefly presented.

With its comprehensive summary of the GMA procedure (incl. suggestions for improvements) and of the existing computer support, this thesis will be of use for any reader who has never heard of GMA - as well as for somebody working in the field but not being familiar with the possibilities for computer support. The presentation of GMAvisual gives the prove that it does make sense to combine GMA with MDS (for at least the problem analyzed in the presentation). Besides many instruments are included to make the visualization more explorable and more conveniently explorable. All this can be of great use for anybody who wants to include cluster visualizations into his GMA software.

2. General Morphological Analysis (GMA)

General Morphological Analysis (GMA) is a generalization of the concept of Morphological Analysis which is used in various scientific domains - always in a slightly different, yet of course similar meaning. The following background information is adapted from [Rit98] and [Rit06b]:

The term *morphology* itself comes from the antique Greek word *morphê* which means *shape* or *form*. With *logos* meaning *word* and in a broader sense *the study and the teaching of something*, morphology is the *study of form and shape*, the *study of pattern*. This means the shape of an object and of its parts, and how these parts are put together to form the whole object.

As the considered objects can principally be from any domain and can be both physical objects (e.g. body parts, rock formations) as well as mental objects (e.g. sentences, ideas), the basic idea of studying objects via decomposing them into their parts has been adapted by several disciplines (e.g. anatomy, geology, linguistics, philosophy). In [Rit98] Ritchey furthermore highlights that “[m]orphology is used in disciplines where *formal structure*, and not necessarily quantity, is a central issue”.

It was Fritz Zwicky (1898 - 1974), a Swiss-American astrophysicist and aerospace scientist, who first detached the concept of morphological analysis from any specific domain and generalized it. To speak with his own words, he further developed the procedure so that it should include the “study [of] the more abstract structural interrelations among phenomena, concepts, and ideas, whatever their character might be”. [Rit98] This generalized concept of morphological analysis nowadays is a popular creativity technique and problem solving method. It is a method to understand and structure problems or systems that is especially suited for multidimensional, non-quantifiable problems to which mathematical or causal modelling cannot be applied.

2.1 The GMA procedure

The basic GMA procedure is described well in most of Tom Ritchey’s publications (for example in [Rit06b]), the following outline however is mostly taken from [ZDSM14]:

1. Definition of the problem

A problem that is to be analyzed (and hopefully solved) with GMA first needs to be well defined and precisely formulated.

2. Structural analysis of the problem (Decomposing the problem into parameters)

The structure (morphê) of the problem needs to be thoroughly examined. Then the *morphological* analysis is executed: the problem is decomposed into subproblems called *parameters* (sometimes also *issues*). These parameters together should comprise the whole problem (that is they should cover all decisive aspects of the problem) while overlapping as little as possible. They shall be “mutually exclusive and collectively exhaustive”[ZDSM14].

Let’s denote the n parameters of the considered problem with $P_i, i \in \{1, \dots, n\}$.

3. Setting up the Morphological Field (Determining the value ranges)

Then a finite range of possible values is determined for each of the obtained parameters. Mathematically modelled: For each parameter P_i a value set $V_i = \{v_1, \dots, v_{m_i}\}$ is defined. The cardinality m_i of the value set V_i is the number of different values that the parameter P_i can take. The values of the parameters can be of qualitative or quantitative nature. The degree of abstraction is determined by the specific problem and the specific goal of each analysis. However it is always important to determine the value range of each parameter independently.

The aggregation of the parameters and their corresponding values is called *Morphological Field*. It is mostly depicted as a table.

4. Setting up the solution space

So the problem has been decomposed into parameters and a range of possible values has been determined for each of the parameters (the Morphological Field has been set up).

A possible solution of the considered problem corresponds to a tuple containing one of the possible values for each of the parameters. The next step of the GMA process is to set up *all* these possible value combinations. Mathematically speaking: the Cartesian product of the value sets of the parameters is calculated. The combinations (the tuples of possible problem solutions) are called *configurations*. The whole set of them (the Cartesian product of the value sets) is called (*formal*) *solution space* or (*formal*) *configuration space*. Let’s denote the formal configuration space with \mathcal{CS} . Then $\mathcal{CS} = V_1 \times \dots \times V_i$ and $k = |\mathcal{CS}| = m_1 \cdot \dots \cdot m_n$ is the number of structurally possible solutions of the considered problem.

The table of the Morphological Field can also be read as a compact representation of the formal solution space.

Example: Planning a holiday trip

As a simple example let's look at the problem of planning a holiday trip. Maybe a group of people want to go on holidays together, that's what they know for sure. But they did not yet come to a decision about what *kind* of trip it should be and want to thoroughly analyze their reasonable options before taking a decision.

1. Definition of the problem

During the remaining time of the year 2015 four friends want to go on holidays together. That's all they know.

What options do they have? What kind of trips can they reasonably undertake together?

2. Structural analysis of the problem (Decomposing the problem into parameters)

The problem can obviously be decomposed into subproblems: How long should the trip be? Should there be one single destination or should it rather be kind of a roadtrip? Should the destination be far away or rather near? What kind of transportation will be used to get to the destination? What about the accommodation? And how much money will be spent?

The problem could be decomposed into the following six main subproblems (parameters) that need to be solved/answered to obtain a solution to the whole problem:

$P_1 = \textit{Duration}$, $P_2 = \textit{Single destination vs. roadtrip}$, $P_3 = \textit{Distance to destination}$,
 $P_4 = \textit{Transportation}$,
 $P_5 = \textit{Accommodation}$, $P_6 = \textit{Costs}$

3. Setting up the Morphological Field (Determining the value ranges)

For each of the parameters P_i it needs to be considered which values the parameter can reasonably take. In the example the sets of possible parameter values V_i are determined as follows:

$V_1 = \{[0,1), [1,3), [3,7), [7,14), [14,30]\}$
 $V_2 = \{\textit{Single destination}, \textit{Main destination} + \textit{roadtrip}, \textit{Only roadtrip}\}$
 $V_3 = \{[0,50), [50,150), [150,300), [300,500), [500,1000], >1000\}$
 $V_4 = \{\textit{none (walking)}, \textit{bicycle}, \textit{train}, \textit{bus}, \textit{car}, \textit{airplane}\}$
 $V_5 = \{\textit{none}, \textit{tent}, \textit{hostel}, \textit{AirBnB}, \textit{hotel}, \textit{all-inclusive}\}$
 $V_6 = \{[0,100), [100,250), [250,500), [500,1000], >1000\}$

Figure 2.1 shows the parameters and their possible value sets in the form of a table, the *Morphological Field*.

4. Setting up the solution space

Every combination of values (one taken from each column of the Morphological Field) is a principally possible solution of the Holiday Trip problem and represents one specific kind of holiday.

For example $([0,1] ; \textit{Single dest.} ; [0,50] ; \textit{train} ; \textit{none} ; [0,100])$ represents a short and cheap day trip by train, $([3,7] ; \textit{Single dest.} ; [500,1000] ; \textit{airplane} ; \textit{hostel} ; [500,1000])$ could stand for a long weekend in a rather remote place.

For this specific problem and with the specific analysis, that has been done, there are $k = 5 \cdot 3 \cdot 6 \cdot 6 \cdot 6 \cdot 5 = 16200$ possible configurations in the formal configuration space, ergo 16200 possible holiday trips. Table 2.1 illustrates the concept of the formal configuration space by showing some cuttings.

Parameters:					
Duration [d]	Single dest. vs. roadtrip	Distance to dest. [km]	Transportation	Accommodation	Costs [€/person]
Parameter values:					
[0,1)	Single destination	[0,50)	none (walking)	none	[0,100)
[1,3)	Main dest. + roadtrip	[50,150)	bicycle	tent	[100,250)
[3,7)	Only roadtrip	[150,300)	train	hostel	[250,500)
[7,14)		[300,500)	bus	Air BnB	[500,1000]
[14,30]		[500,1000]	car	hotel	>1000
		>1000	airplane	all-inclusive	

Figure 2.1: Morphological Field of the Holiday Trip problem

<ul style="list-style-type: none"> • ([0,1), Single destination, [0,50), none (walking), none, [0,100)) • ([0,1), Single destination, [0,50), none (walking), none, [100,250)) • ([0,1), Single destination, [0,50), none (walking), none, [250,500)) • ... • ([0,1), Only roadtrip, [500,1000], car, none, [0,100)) • ([0,1), Only roadtrip, [500,1000], car, none, [100,250)) • ([0,1), Only roadtrip, [500,1000], car, none, [250,500)) • ... • ([1,3), Only roadtrip, [300,500), train, none, [0,100)) • ([1,3), Only roadtrip, [300,500), train, none, [100,250)) • ([1,3), Only roadtrip, [300,500), train, none, [250,500)) • ... • ([3,7), Only roadtrip, [150,300), none (walking), none, [0,100)) • ([3,7), Only roadtrip, [150,300), none (walking), none, [100,250)) • ([3,7), Only roadtrip, [150,300), none (walking), none, [250,500)) • ... • ([7,14), Only roadtrip, [0,50), car, none, [0,100)) • ([7,14), Only roadtrip, [0,50), car, none, [100,250)) • ([7,14), Only roadtrip, [0,50), car, none, [250,500)) • ...
--

Table 2.1: Cuttings of the whole formal configuration space of the Holiday Trip problem.

2.2 Consistency assessment - known approaches

If a considered problem is complex and the possible solutions are not examined systematically, it is easy to miss one particular solution (and maybe this is just one of the best solutions!). But via morphological analysis the space of possible solutions of a problem is systematically opened up: The formal configuration space \mathcal{CS} is a set that contains really every possible solution for the analyzed problem. Therefore it is just not possible to forget anything.

But then, on the other hand, with a growing number of parameters and values, the formal configuration space can become just too large to oversee it. Without further structuring or reduction of the configuration space, the technique of GMA would be rather useless for the majority of complex problems.¹

The most common first step for making the configuration space more accessible is the *consistency assessment*.² It aims at reducing the configuration space by finding and erasing the *inconsistent* configurations: Some of the generated configurations just don't make sense, because some of the values in the configuration-tuple contradict each other. Therefore the configuration is only a formal solution, but not really a possible solution to the problem.

In the Holiday Trip example $c_1 = ([\mathbf{0},\mathbf{1}], \textit{Single dest.}, [0,50], \mathbf{airplane}, \textit{none}, [100,250])$ and $c_2 = ([\mathbf{7},\mathbf{14}], \textit{Single dest.}, [300,500], \textit{airplane}, \mathbf{all-inclusive}, [\mathbf{250},\mathbf{500}])$ are rather inconsistent configurations. In c_1 one would probably judge it contradictory to go for a one-day trip by plane, c_2 describes a week-long stay in an all-inclusive hotel for less than 500 €.

2.2.1 Binary pairwise consistency assessment

Scanning the configuration space for inconsistent configurations is a time-consuming task, as there are $k = m_1 \cdot \dots \cdot m_i$ configurations to be checked, with $k = 16200$ in the example. As this task cannot really be executed by a computer (unless maybe with some future semantic technologies) there needs to be found another way than to look at each and every configuration separately. The most common approach (and probably the only one that has been applied so far) is to do *pairwise* Cross Consistency Assessment (CCA). The assumption is, that most of the inconsistent configurations have *two* of their values being mutually inconsistent.

In pairwise CCA each pair of values (v, w) taken from two different value sets is assessed for mutual consistency ($v \in V_i, w \in V_j, i \neq j$). If a configuration contains such a pair of values that is inconsistent, the whole configuration is inconsistent (like above in the example-configuration c_1). The great advantage: As the number of configurations grows exponentially with the number of parameters, the number of pairwise relationships between value sets only grows as a quadratic polynomial, or more precisely in relation to the triangular number series.³ While $k = m_1 \cdot m_2 \cdot \dots \cdot m_n$ is the number of configurations, the

¹Remark: A deeper insight into the problem would still be won because of the reflections about decompositions into parameters and their possible value ranges. But there would be little further use.

²Confer again [Rit98], [Rit06b] and [ZDSM14].

³Verifiable mathematical fact, but also mentioned by Tom Ritchey in [Rit98] and [Rit06b].

each other, they are judged *inconsistent*. The respective box in the matrix is marked, the others are left empty. In this thesis this approach of assessing the pairs via the two judgements **consistent** - **inconsistent** is referred to as *binary*.

This binary approach, as well as the rest of the procedure as it has been described so far, is all adopted from the work of Dr. Tom Ritchey, a former Research Director for the *Institution for Technology Foresight and Assessment* at the Swedish National Defence Research Agency (FOI) in Stockholm. As stated in [Rit06a], “he is the founder of the Swedish Morphological Society (www.swemorph.com), Director of Ritchey Consulting Inc. and a founding partner of the U.K. based Strategy Foresight Partnership.” The described procedures of GMA and binary pairwise CCA can be found in most of his publications. Here mainly [Rit98], [Rit06a] and [Rit06b] have been summarized.

It shall not be left unmentioned that Tom Ritchey also introduces more than binary consistency assessments in his work. In the application case presented in [Rit06a] for example, there appears a cross consistency matrix with entries “-” for **consistent**, “X” for **inconsistent** and “K” for two values that are not inconsistent, but are “highly unlikely or uninteresting”. In [Rit09] it is explained, that configurations containing one of these “K”-pairs can optionally be deleted from the configuration space. In the supporting software the affected configurations can thus be toggled on and off.

2.2.2 Scaled pairwise consistency assessment

There is an alternative approach, very obvious and simple yet powerful, that has already been applied. Unfortunately there are no publications about it, but it can be found within the Parmenides EIDOS “software suite for complex decision-making” - that has generously been provided by the Parmenides Foundation for the purpose of this thesis (see Section 4.2). There the cross consistency matrix is not filled with designators for **consistent** and **inconsistent**, but with integral values ranging from -3 to 3 . The higher the value with which a value pair is assessed, the better the two values fit together. This approach will be referred to as *scaled* pairwise consistency assessment. Figure 3.5 shows the cross consistency matrix of the Holiday Trip problem, as it would appear in the EIDOS software suite.

With the numerical consistency values of the value pairs, consistency values of the whole configurations can be calculated (in EIDOS just via the arithmetic mean of the involved pairs) and then the configurations can be sorted by consistency. This great advantage will be examined more thoroughly in Section 3.4.

3. Possible improvements of the consistency assessment

3.1 Binary pairwise consistency assessment - effectiveness

With the binary pairwise CCA, as just described in Section 2.2.1, the configuration space can be reduced by all configurations that contain at least one of the value pairs that are marked inconsistent. And as each of the pairs appears in a lot of configurations, this very basic approach can already yield a good reduction of the configuration space: With the assessment of Figure 2.2 the configuration space of the Holiday Trip example is reduced by 71%, from 16200 to 4641 configurations. And this is not even a very good score. Depending on the percentage of pairs that are judged inconsistent, the percentage of inconsistent configurations rises significantly. While the first inconsistent pairs affect the full formal configuration space (so only the one inconsistent pair ($[0,1]$; *airplane*) from the example configuration c_1 would reduce \mathcal{CS} by 540 configurations!), later pairs can only eliminate the configurations that are still contained in the current \mathcal{CS}' . Therefore the curve of reduced configurations first rises steeply and then flattens.

Figure 3.1 shows the percentage of reduced configurations in dependence of the percentage of inconsistent value pairs. The diagram on the right depicts the data points for some real example problems (each having six parameters).¹ The diagram on the left shows data for a dummy problem ($n = m = 6$). Here, for each measurement a certain percentage of pairs has randomly been assessed inconsistent. Each data point represents the average of five measurements.

¹Alongside the Holiday Trip problem, this are the Multihazard Disaster problem of MA/Carma (found in [Rit06a]) and six example problems coming with the Parmenides EIDOS software suite.

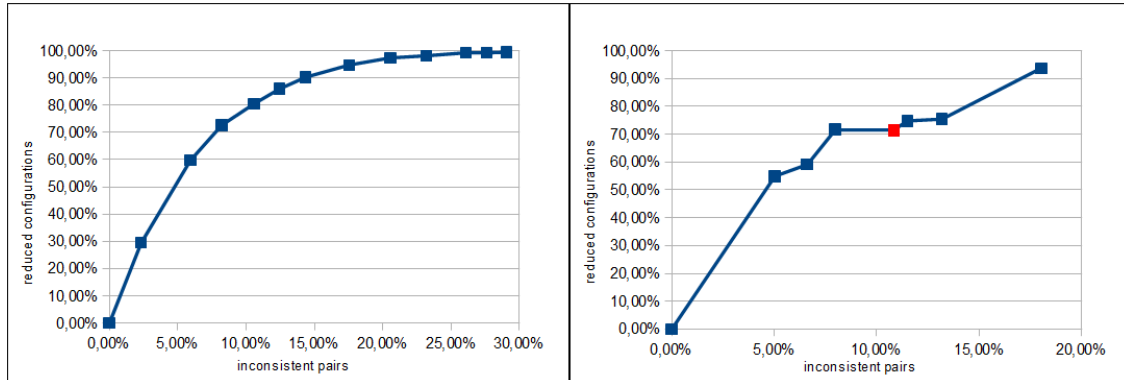


Figure 3.1: Percentage of reduced configurations in dependance of the percentage of inconsistent value pairs. *Left:* Averages of five measurements (each) with randomly assessed pairs for a 6x6 problem. *Right:* Data of some real example problems with six parameters each. Red is the data point for the Holiday Trip problem.

3.2 Higher order consistency assessment

The goal of the consistency assessment step should of course principally be to eliminate *all* inconsistent configurations from the formal configuration space and thereby receive the actual solution space. This is obviously *not* achieved by only doing pairwise CCA: The example configuration $c_2 = ([7,14], \text{Single dest.}, [300,500], \text{airplane}, \text{all-inclusive}, [250,500])$ from above does not contain any inconsistent value pair and will therefore survive the pairwise CCA. However the configuration c_2 would be judged inconsistent and we would actually want to filter it out of the configuration space.

To get optimal filtering results, further consistency assessment would be needed. In his works about future studies that make use of the GMA idea (amongst others in [Rhy95]), Russell Rhyne suggests - as subsequent step - to directly look at each of the remaining configurations and decide if it makes sense or not. So he goes from pairwise CCA directly to the consistency assessment of the whole n -tuples that remain (n being the number of parameters of the considered problem). In many scenarios, and especially in those that Rhyne is dealing with, that seems like a reasonable approach - as the configuration spaces are usually being drastically reduced by pairwise CCA. In [Rhy74] he mentions an example where the configuration space is being reduced from 189,000 to fewer than 100. But as the concept of GMA as presented here shall be most generally applicable, this approach is at first not really satisfying. In our Holiday Trip example case we would have to consider all 4641 separate configurations that are remaining, to find the remaining inconsistent ones.

But there might be another approach: If we at first look at the value pairs, why don't we look at the value *triples* after that? In the example configuration c_2 it is the value triple $([7,14], \text{all-inclusive}, [250,500])$ that makes the configuration inconsistent (as more than seven days in an all-inclusive hotel would cost more than 500 €). Filtering out all configurations containing a value triple of this kind would reduce the original configuration space by 108 configurations. In our case, after reducing the space via pairwise CCA, there are still 18 configurations left that contain this specific triple. That means, in this case,

we can further reduce the configuration space by 18 configurations - with only one triple being judged inconsistent.

So why not generally apply triplewise CCA after the pairwise CCA? After all there is the possibility that additional inconsistent configurations are filtered out! - And after the triplewise CCA the value quadruples could be assessed, and then the value quintuples, and so on - until finally, at the very end, the whole n -tuples (the configurations themselves) would be assessed.² Every step might improve the consistency assessment and find some new inconsistent configurations. - And yet a general application of the higher-order CCA is not to be recommended, as the following considerations show:

Large configuration spaces

Of course, every step would not only reduce the number of configurations in the configuration space, but also the number of x -tuples that have to be assessed in the respectively next step. But still there is one problem: If the considered configuration space is big and/or there are just very few inconsistent configurations, then the configuration space just cannot be reduced very much after all. And then, additionally going through all the triples, quadruples, and so on, would do nothing but multiply the effort. For example: Let's assume that all of the 4641 remaining holiday configurations are valid. A user (who does not know about this), who wants to optimize the consistency assessment and therefore wants to go through all the x -tuples successively, would have to assess 1958 triples, 5645 quadruples, 8152 quintuples and then, finally, the 4641 configurations. This does obviously not make any sense. Even going only through the triples and then considering the whole configurations (for the really optimal consistency assessment) is a questionable procedure in this case. If perfect consistency assessment is wanted (theoretically) and therefore the whole configurations have to be examined anyway, it's most probably of no use to look at the triples before. Indeed the configuration space would have to be reduced by more than 1958 additional configurations (via the triplewise CCA) to make this step efficient.

Binomial coefficient

As it already appeared in the numbers of x -tuples of the Holiday Trip problem: there is one more principle objection against successively assessing all the (remaining) x -tuples. And that is the characteristics of the binomial coefficients, namely their distribution. Let's look at a decision problem with n parameters, each having a fix number of values ($\forall i : m_i = m$). The number of x -tuples ($x \in \{0, \dots, n\}$) is then calculated by $\binom{n}{x} \cdot m^x$. And while the second part grows exponentially, the binomial coefficient is obviously binomially distributed and reaches its maximum at $\frac{n}{2}$. Therefore, as long as $m < n$, there are *more* $(n-1)$ -tuples than n -tuples. In an example case of $n = 6$ that means that there are more quintuples than sextuples - which are the actual configurations. With a low m (relative to n) this might go down to situations where there even are more triples than actual configurations (even before doing the pairwise CCA reduction!). Figure 3.2 shows

²This obvious idea is a.o. mentioned in [CCS94].

the growth of the number of x -tuples with the growing of x (for the Holiday Trip problem and some dummy problems having six parameters and a fix number of parameter values). In cases where there are more x -tuples than actual configurations it obviously does make even less sense to sit down and assess the x -tuples manually.

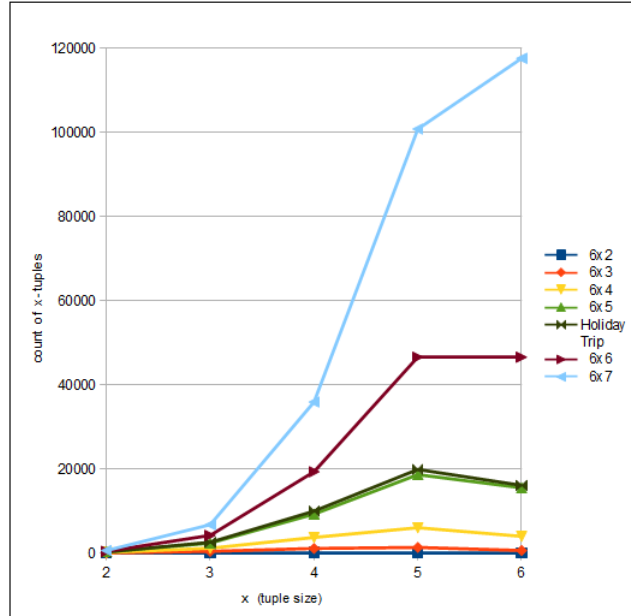


Figure 3.2: Number of x -tuples depending on the respective tuple size x for several problems. $\nu \times \mu$ means a problem of $n = \nu$ parameters, all having $m = \mu$ possible parameter values.

And then - furthermore - when the x -tuples are reduced using the inconsistencies from the pairwise CCA³, the described situation may become even more prominent. The numbers of x -tuples and reduced x -tuples of the Holiday Trip example are depicted in Figure 3.3. As one can see, the number of the whole configurations (sextuples) is not only smaller than the number of quintuples from the beginning, it also decreases the most (in relation to the original number). That way - after the reduction by the pairwise consistency - there are even less sextuples left than quadruples. This development is a characteristic phenomenon, as one can see by the other example problems in Figure 3.3. No matter how the counts of the x -tuples are originally distributed, when you reduce them via the value pair inconsistencies, the percentage of eliminated x -tuples will be directly proportional to x . The higher the considered tuple size x , the more the x -tuples will be reduced percentually.

As already touched above, the relation between the number of parameters n and the average size of the value ranges m plays an important role when it comes to reduction rates. The higher m in relation to n , the lower the percentage of reduced x -tuples will be. The

³From the set X containing all possible x -tuples, remove all x -tuples that contain at least one pair of values that has been judged inconsistent in the pairwise CCA step and receive the reduced set \bar{X}_2 .

different appearance of the diagrams of the Ammonia Accident problem and the Multihazard Disaster problem is mainly due to this effect: The first has an average value set size of 4.0 (while having 8 parameters), the latter of 9.7 (while having only 6 parameters).

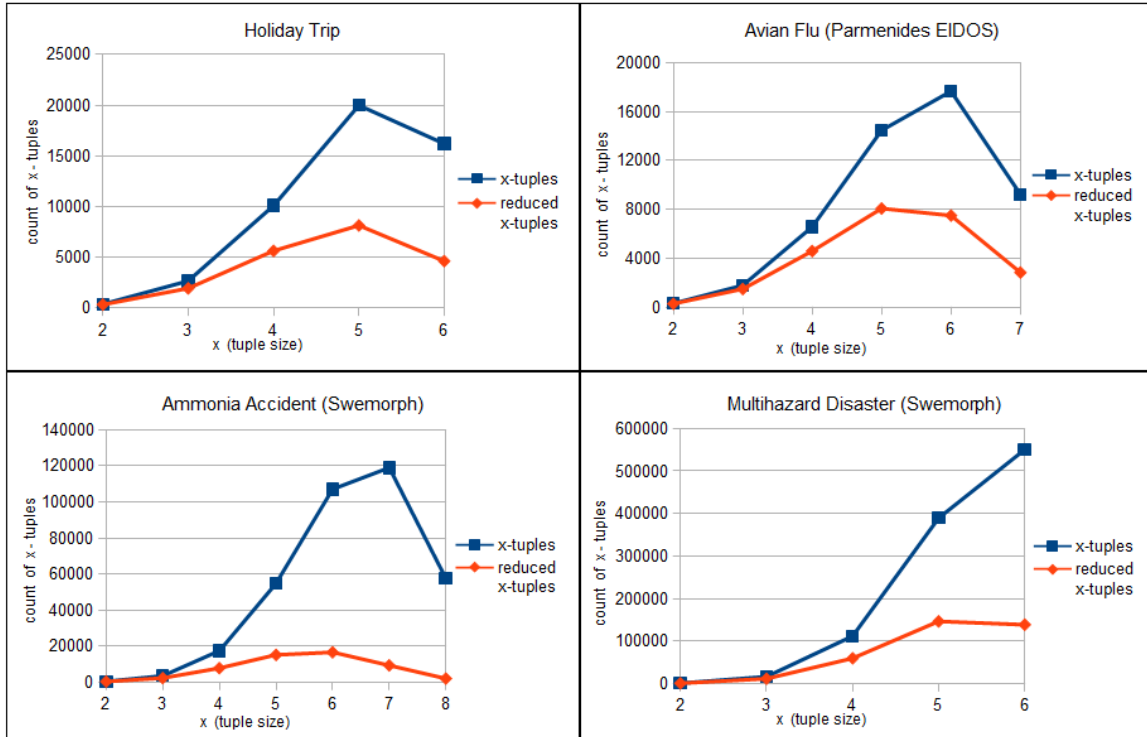


Figure 3.3: Number of x -tuples depending on the respective tuple size x for four example problems. *Blue*: without any reduction. *Red*: the x -tuples reduced via the consistency assessment data from the pairwise CCA.⁴

In conclusion, generally applying higher-order consistency assessment is not a good idea. If the configuration space and the sets of the other x -tuples are not too big anyway, it is not improbable that there are less overall configurations to be assessed than even value triples.

Another point is, that the analyzed problem should have been decomposed into parameters that are mutually exclusive.⁵ Ideally they don't overlap and are independent from each other. In this ideal case there are no higher-order inconsistencies to be expected. In a less ideal case at least only a few. In this case going through thousands of tuples would still mean cracking a walnut with a sledgehammer.

3.3 Selective triple consistency assessment (TCA)

However there might be cases where the modelling could not be done so successful. Where the obvious decomposition into parameters leaves them with some interdependencies. Or

⁴The two problems from the Swedish Morphological Society are adapted from [RSE02] and [Rit06a].

⁵See Section 2.1.

where the decomposition could not reasonably have been done otherwise. In these cases (and with the intention to optimize the consistency assessment of course) one should still not directly take on all existing value triples, but rather first assess the trios of *parameters* and ask the question: Is it likely that there are inconsistent value triples inherent in this specific combination of parameters? With this question answered, one can then assess only the *relevant* value triples for consistency.

To illustrate this idea: In the Holiday Trip problem it is obvious that the duration of the trip, the accommodation and the costs are interwoven. Paying 500 € for a stay in an all-inclusive hotel (maybe for a weekend) is not impossible, having a ten-day vacation for 500 € is principally possible, as well as a ten-day vacation in an all-inclusive hotel. The triple of the three features would however be judged inconsistent. Within this *parameter trio* (of duration, accommodation and costs) one would maybe assume more inconsistent value triples than just this one. Other parameter trios might be judged less promising because its parameters are mutually rather independent (like maybe duration, transportation and accommodation). In this case the inconsistent triples would already have been filtered out by pairwise CCA.

Therefore the suggestion is to introduce *selective triple consistency assessment* (selective TCA). After the pairwise CCA the promising parameter trios are chosen and then the value triples of the chosen parameter trios are assessed for consistency (of course only the value triples that do not contain any inconsistent pair). If no parameter trio seems promising (because the parameters are sufficiently independent) or if the triple assessment would be unreasonably time-consuming, this step can just be skipped. An important remark about the costs of this suggestion: The number of parameter trios is given by the binomial coefficient $b = \binom{n}{3}$ which is given in the following table (up to a parameter count of $n = 9$). There may be thousands of value triples - the number of parameter trios however remains well manageable:

parameters n	3	4	5	6	7	8	9
parameter trios b	1	4	10	20	35	56	84

The actual assessment of the selected value triples can obviously not be executed via a clear graphic representation like the pairwise cross consistency matrix. The easiest approach is to present the triples successively (as depicted on the right of Figure 3.4). The great advantage of a matrix representation is that all the “neighboring” tuples are visible. This allows the assessing person to calibrate his judgements (“*If I judged the other pair inconsistent, I should also judge this one inconsistent. Or maybe I have to reconsider the other judgement.*”). Assessing the triples, this advantage is lost when the triples are shown successively and separately. Even if it is made possible to go back and revise the previous assessments, a real overview will not be given. The alternative of showing all the selected triples in a list would improve this, but the three-dimensional value matrix would still be depicted in one dimension and the real overview of the neighboring entries would therefore still not be given. But there is another reason that speaks against the list version: Assessing the consistency of a triple is significantly more complex and time-consuming

than assessing a tuple, as one has to hold one more value in one's mind, combine it with the others and think of possible worlds for the combination. While the actual decision holds more information and might therefore be indeed easier and more accurate, the simulation of the combination tends to need more intellectual work. Therefore it is preferable to have a surface that prominently highlights only the currently assessed triple. That makes it easier to concentrate on this special case. - Both representations, the successive and the list representation, are not optimal solutions. Elaborate GMA software could combine these two, or try some clever and/or graphically enhanced matrix depictions.

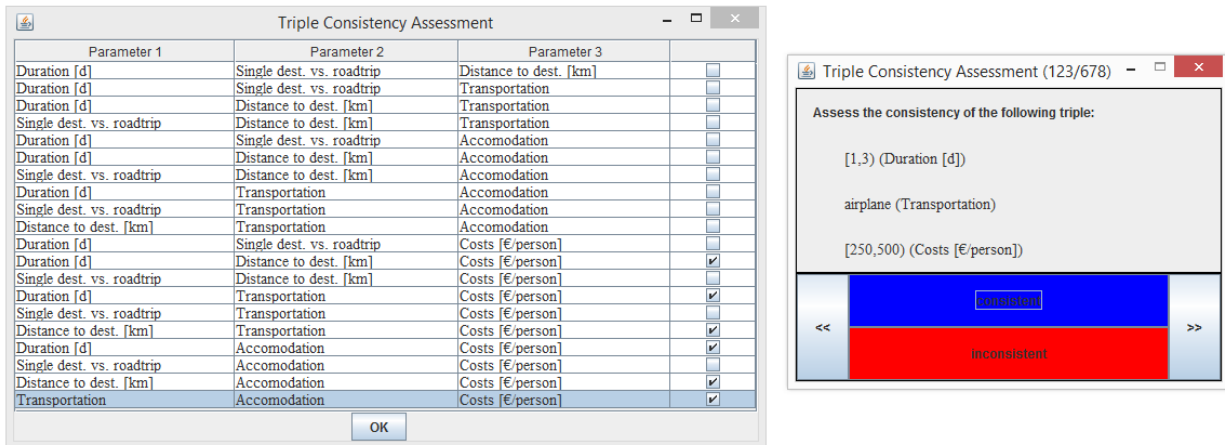


Figure 3.4: Selective triple consistency assessment for the Holiday Trip problem. *Left*: selection of the promising parameter trios. *Right*: successive and separate assessment of the value triples.

Holiday Trip In the case of the Holiday Trip example we were left with 4641 out of 16200 configurations and the feeling that the pairwise CCA did not eliminate all inconsistent configurations. It especially seemed like it's the *costs* that often depend on the combinations of the other values (e.g. of the pair duration-accommodation or the pair duration-transportation). With the selection of parameter trios that is shown on the left of Figure 3.4, one has to assess 678 out of the 1958 remaining value triples. Confronted with the actual triples, there were surprisingly many that seemed so improbable that they were judged inconsistent. With 144 inconsistent value triples the configuration space was again reduced from 4641 to 1786, so by another 61.5%. Table 3.1 shows examples of value triples that have been judged inconsistent.

Conclusion: With 397 tuple assessments (out of which 43 inconsistent) the configuration space of the Holiday Trip problem was reduced from 16200 to 4641. As next step, to improve the consistency assessment, 678 selected triples were assessed (out of which 144 inconsistent). Thereby the configuration space was again reduced to 1786.

Although this specific example problem is obviously rather suited for selective TCA (for the parameters are not perfectly independent), the additional step is less efficient than the pairwise CCA. In many other cases selective TCA will not be useful at all. Yet it would mean missing out on opportunities to principally ignore it.

Duration	Transportation	Costs [€/person]
[3,7)	bus	>1000
[3,7)	airplane	[100,250)
[0,1)	none (walking)	[100,250)

Distance to dest. [km]	Accommodation	Costs [€/person]
>1000	all-inclusive	[100,250)
[150,300)	hotel	>1000
[0,50)	tent	[250,500)

Transportation	Accommodation	Costs [€/person]
bicycle	none	[100,250)
airplane	all-inclusive	[100,250)
bus	tent	>1000

Duration [d]	Accommodation	Costs [€/person]
[7,14)	all-inclusive	[250,500)
[3,7)	hotel	[0,100)
[3,7)	tent	[500,1000]

Duration [d]	Distance to dest. [km]	Costs [€/person]
[3,7)	[150,300)	>1000
[0,1)	[0,50)	[100,250)
[7,14)	>1000	[100,250)

Distance to dest. [km]	Transportation	Costs [€/person]
[150,300)	car	>1000
[0,50)	none (walking)	[250,500)
[500,1000]	bicycle	>1000

Table 3.1: Selective triple consistency assessment for the Holiday Trip problem. Selected *examples* of triples that have been judged inconsistent.

3.4 Scaled consistency assessment

Until now we have set up the formal configuration space of a problem (containing really every solution to the problem) and then reduced it as far as reasonably possible. The result is a set of configurations of which we have good reasons to believe that it contains a high percentage of valid configurations. And if we are lucky and this reduced set is pretty small and maybe contains only some dozens of valid solutions, we can go through these remaining configurations, evaluate them, categorize them and select the ones that

are supposedly best suited for our problem. But what if there are several hundreds or even thousands of remaining configurations? - Without any additional steps, such a huge, unstructured set of configurations will be of little practical use. We will see in Section 4.1 that it is well possible to offer computer support for the exploration of the unstructured configuration space, but this section will first come back to the very simple yet considerable enhancement to the consistency assessment, that has already been applied in the Parmenides EIDOS software suite: *scaled* consistency assessment (confer Section 2.2.2). The idea is indeed simple: When assessing the value tuples for consistency, why only make the binary assessment **consistent/inconsistent**? Why not grade the value tuples on a larger scale? The suggestion is to introduce Likert scaling and to treat every tuple assessment as a Likert item with five or seven levels (cf. [Wik15]). (Three levels should be too little to collect all information, more than seven however would not bring any additional benefit and rather distract the assessing person.) Opting for seven levels, the question for each value pair would be: “How good do these two values fit together?” The possible answers could be: **worst, very bad, bad, maybe, good, very good, best**. With this information collected, consistency values can be calculated for all the configurations in the configuration space. And then the configurations in the configuration space can be ordered according to their consistency values. This is a decisive advantage: now the output of the whole process is not an unstructured set of configurations, but an ordered list. Without any further steps or computer support, one can now at least examine only the first, let’s say 100, *most consistent* configurations!

3.4.1 Calculating the consistency value of a configuration

But how to calculate the consistency value of a configuration out of the Likert-scaled consistency assessment? The easiest way is to define a weighting function \bar{w} that assigns numerical values to the Likert-levels and then calculate the average of the consistency values of the involved tuples (just as EIDOS does it). To formalize this: for a configuration $c \in \mathcal{CS}$ comprising the tuple set T_c the consistency value $v(c)$ is calculated as follows:

$$v(c) = \frac{1}{|T_c|} \cdot \sum_{t \in T_c} w(t) , \text{ with } |T_c| = \binom{n}{2}. \quad (3.1)$$

Thereby $w(t)$ is the consistency value of the tuple t which corresponds to the numerical value that has been assigned to the consistency level of t by the weighting function \bar{w} . Figure 3.5 shows the cross consistency matrix of the Holiday Trip problem with the scaled assessment. For reasons of convenient depiction the textual descriptions of the consistency levels have been replaced with integral values from -3 to 3 . That way the appearance of the matrix is equal to the EIDOS approach. The obvious coloring has also been adapted from there.

		Single dest.	Single dest. vs. roadtrip																												
		Main dest. + roadtrip	Only roadtrip	Distance to dest. [km]										Transportation																	
				[0,50]	[50,150]	[150,300]	[300,500]	[500,1000]	>1000	none (walking)	bicycle	train	bus	car	airplane	none	tent	hostel	AirBnB	hotel	all-inclusive	Gests [€/person]									
				[0,100]	[100,250]	[250,500]	[500,1000]	>1000																							
Duration [d]	[0,1]	3	2	2	3	2	1	-1	-2	-3	3	3	3	0	3	-3	3	3	3	-1	2	-3	3	-1	-2	-3	-3				
	[1,3]	3	1	1	2	3	3	2	0	-1	1	2	3	2	3	-1	-3	2	2	-1	1	-1	-1	2	2	0	-2				
	[3,7]	3	1	1	1	2	2	2	1	0	0	2	2	2	2	2	-3	1	1	2	2	1	0	2	1	1	-1				
	[7,14]	2	1	2	-1	0	1	2	2	2	0	2	1	1	2	2	-3	1	1	2	1	2	-2	1	1	1	0				
	[14,30]	2	2	2	-1	0	1	2	2	2	-1	1	1	1	2	3	-3	1	1	2	1	3	-3	-2	-1	2	2				
Single dest. vs. roadtrip	Single dest.			2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	1	1	1	1	1			
	Main dest. + roadtrip			-1	0	1	2	2	2	0	1	1	1	2	2	0	2	2	-1	1	-2	0	1	2	2	2	2				
	Only roadtrip			1	1	1	2	2	2	2	3	1	1	3	-3	1	2	1	-2	1	-3	1	1	1	1	1	1				
Distance to dest. [km]	[0,50]			3	2	2	-1	2	-3	1	2	-1	-2	-1	-3	2	1	-1	-2	-3	2	1	-1	-2	-3						
	[50,150]			0	2	2	0	2	-3	1	1	0	-2	0	-1	1	1	0	-1	1	1	0	-1	-2							
	[150,300]			0	1	2	2	2	-2	0	2	1	-2	1	0	1	1	1	1	0	1	1	1	1	0						
	[300,500]			-2	1	2	2	2	1	-1	1	2	0	1	0	0	1	2	1	1	0	0	1	2	1	1					
	[500,1000]			-3	1	2	2	2	3	-3	-1	2	2	2	2	-2	0	0	1	2	2	-3	-1	0	1	2					
	>1000			-3	1	1	1	2	3	-3	0	2	2	2	2	-3	-1	0	1	2	2	-3	-1	0	1	2					
Transportation	none (walking)			3	0	-1	-3	1	-3	2	-1	1	0	0	3	0	-1	-3	1	-3	2	-1	1	0	0						
	bicycle			2	2	2	-3	1	-2	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	1					
	train			1	1	2	1	1	-1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	0					
	bus			-1	0	1	-1	1	-1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	0					
	car			2	2	1	2	1	1	1	1	2	2	1	1	2	2	1	1	1	1	1	2	2	1	1					
Accomodation	none			2	0	-1	-2	-3	2	0	-1	-3	1	-3	2	0	-1	-2	-3	2	0	-1	-2	-3							
	tent			2	2	1	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2						
	hostel			1	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						
	AirBnB			-1	-1	0	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						
	hotel			0	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2						
	all-inclusive			-1	-1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2						

Figure 3.5: Cross Consistency Matrix of the Holiday Trip problem (consistency assessment with rating scale) - graphic appearance (numbers and colors) adapted from the Parmenides EIDOS software suite.

3.4.2 Equidistant consistency calculation

As mentioned above, the Parmenides EIDOS software suite for complex decision-making does indeed directly implement the pairwise CCA as depicted in Equation 3.1. The consistency of tuples is measured not with textual representations, but directly with numerical values from -3 to 3 . Then the consistency value of a configuration is calculated by directly adding up these values and dividing the sum by the number of tuples contained

in one configuration. So the configurations themselves also get consistency values between -3.0 and 3.0 . Applied to our context, this would mean that the weighting function \bar{w}_1 is given by:

$$\begin{aligned} \bar{w}_1(\text{best}) &= 3.0 & \bar{w}_1(\text{bad}) &= -1.0 \\ \bar{w}_1(\text{very good}) &= 2.0 & \bar{w}_1(\text{very bad}) &= -2.0 \\ \bar{w}_1(\text{good}) &= 1.0 & \bar{w}_1(\text{worst}) &= -3.0 \\ \bar{w}_1(\text{maybe}) &= 0.0 & & \end{aligned}$$

This approach does yield a ranking of all configurations by their internal consistency, however it is kind of a questionable one: After all the whole point of the classic pairwise CCA (as Coyle, Rhyne and Ritchey apply it) was to eliminate the configurations that don't make sense. A configuration that contains one inconsistent pair (judged **worst**) is itself completely inconsistent and can be completely discarded. If it's not directly deleted from the configuration space, it should at least come very late on the ranking of configurations. However with this equidistant approach, one **best** tuple is already sufficient to compensate for such an inconsistent tuple. In a problem with six parameters, each configuration comprises 15 tuples. If one of them is inconsistent and the others happen to get **best** assessments, the consistency value of the configuration would be 2.7, so very, very good - while the configuration actually doesn't make sense. Such an extreme coincidence is of course improbable, but an occurrence of an actually inconsistent configuration among the best in the ranking can not be ruled out.

The frequency distributions of some example problems suggests that the equidistant calculation of the consistency values is not fatal: the very best configurations will usually not contain **worst** tuples. The left side of Figure 3.6 shows the frequencies of configurations (by consistency value) together with the frequencies of configurations that comprise at least one negative tuple (**1a**: at least one **very bad** or one **worst** tuple; **1b**: at least one **worst** tuple; **1c**: at least one **very bad** tuple; **1d**: at least one **bad** tuple). The figure shows the data for the Holiday Trip example problem which are quite representative, also for the about ten example problems coming with the Parmenides EIDOS suite (which have cross consistency matrices with entries from -3 to 3 as described above and can therefore be examined in this context). As one can see, the lower rated half of the configuration space contains at least one **very bad** or one **worst** tuple - and, on the other hand, there are quite some configurations rated best, that contain neither. However configurations with one of the two low consistency assessments begin to mix into the others early. Only the first 22 configurations (0.14%) do contain neither of the two, only the first 631 configurations (3.90%) contain no **worst** rated tuple.

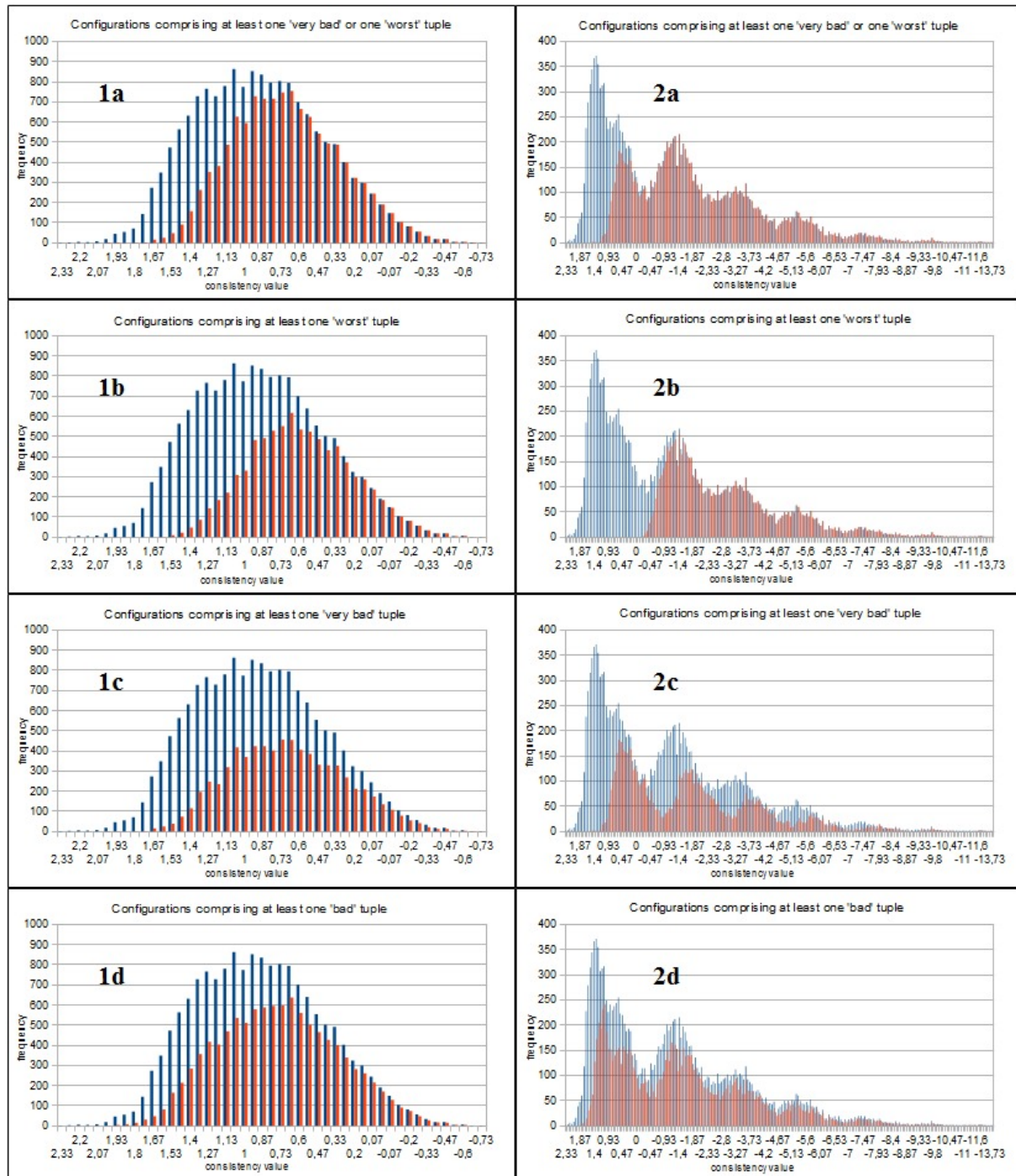


Figure 3.6: Calculating consistency values of configurations: Frequency distributions of the Holiday Trip problem. *Left*: the equidistant approach. *Right*: the weighted approach. *Blue*: the count of configurations having a certain consistency value. *Red*: the count of configurations having a certain consistency value while comprising at least one negative tuple (**a**: at least one *very bad* or one *worst* tuple; **b**: at least one *worst* tuple; **c**: at least one *very bad* tuple; **d**: at least one *bad* tuple).

3.4.3 Weighted consistency calculation

The equidistant approach is not fatal, but definitely unsatisfying. The goal of the CCA step is to reduce the configuration space, to identify and eliminate all inconsistent configurations. And one inconsistent pair makes a whole configuration inconsistent. While one **best** pair does not make a configuration fundamentally better than others, one **worst** pair principally disqualifies it. In the same way a **very good** pair and a **very bad** pair do not have the same conceptual weights. And these weights should go into the calculation of the consistency values of the configurations.

The actual definition of the weighting function depends on the field of application: How strong shall a configuration be punished for a bad tuple? Should configurations with a **worst** tuple have no chance at all? Is a **very bad** tuple also actually completely inconsistent? Shall a configuration be especially rewarded for a **best** tuple? -

As a working basis, the following weighting function \bar{w}_2 is suggested:

$$\begin{array}{ll} \bar{w}_2(\mathbf{best}) = 3.0 & \bar{w}_2(\mathbf{bad}) = -3.0 \\ \bar{w}_2(\mathbf{very\ good}) = 2.0 & \bar{w}_2(\mathbf{very\ bad}) = -10.0 \\ \bar{w}_2(\mathbf{good}) = 1.0 & \bar{w}_2(\mathbf{worst}) = -30.0 \\ \bar{w}_2(\mathbf{maybe}) = 0.0 & \end{array}$$

Like this a configuration *can* make up for a **worst** tuple, but will therefore need ten **best** tuples in return, and one **very bad** tuple needs five **very good** or three **best** tuples to be balanced. The positive designators are left with their linear weights. That way rather the overall best configurations will rate first (and not the ones with the most **best** tuples). The right side of Figure 3.6 shows the data of the Holiday Trip problem (which are again pretty representative) for this assymmetric weighting function \bar{w}_2 . Again there are shown the frequencies of configurations (by consistency value) together with the frequencies of configurations that comprise at least one negative tuple (**2a**: at least one **very bad** *or* one **worst** tuple; **2b**: at least one **worst** tuple; **2c**: at least one **very bad** tuple; **2d**: at least one **bad** tuple). As one can see, the configurations with bad tuples are rather pushed down. Now, with \bar{w}_2 , the first 1144 configurations (7.06%) don't contain any **very bad** or **worst** tuple, while the first 7017 configurations (43.31%) don't contain any **worst** tuple.

3.4.4 Reflections on the weighting function

One systematic approach for defining the weighting function could be to make it definitely impossible for a configuration with one **worst** tuple to get a positive consistency value. In that case (for a problem with n parameters and the linear positive weights from above), the weight of the **worst** assessment would have to be $\bar{w}_3(\mathbf{worst}) = (b - 1) \cdot (-3)$, with b being the number of tuples per configuration: $b = |T_c| = \binom{n}{2}$. If one tuple is rated **worst** and all others are rated **best**, the configuration will that way just get a consistency value of 0.0. The following table shows the count b of tuples per configuration and what the weight $\bar{w}_3(\mathbf{worst})$ would have to be, if one follows this approach:

parameters n	3	4	5	6	7	8	9
tuples per configuration b	3	6	10	15	21	28	36
$\bar{w}_3(\mathbf{worst})$	-6.0	-15.0	-27.0	-42.0	-60.0	-81.0	-105.0

With $\bar{w}_3(\mathbf{worst})$ being fixed, the range for the weight of the other two negative designators is also determined (to obviously be between $\bar{w}_3(\mathbf{worst})$ and 0.0). Then there are several options: One is of course, to anyhow set the other weights to fix values (independent from n and $\bar{w}_3(\mathbf{worst})$). Another option is to choose them to in each case fit a certain polynomial shape, for example by calculating them *proportionally*.

To give an example for the last suggestion: The designator **very bad** could be seen as “half as bad” and the designator **bad** as “one tenth as bad” as **worst**. (Let $p_{\mathbf{very\ bad}}$ and $p_{\mathbf{bad}}$ denote the proportion parameters, with $p_{\mathbf{very\ bad}} = \frac{1}{2}$ and $p_{\mathbf{bad}} = \frac{1}{10}$ in this case.) Then:

$$\bar{w}_3(\mathbf{very\ bad}) = p_{\mathbf{very\ bad}} \cdot \bar{w}_3(\mathbf{worst}) = \frac{1}{2} \cdot \bar{w}_3(\mathbf{worst})$$

$$\bar{w}_3(\mathbf{bad}) = p_{\mathbf{bad}} \cdot \bar{w}_3(\mathbf{worst}) = \frac{1}{10} \cdot \bar{w}_3(\mathbf{worst}) = \frac{1}{5} \cdot \bar{w}_3(\mathbf{very\ bad}).$$

With these arbitrarily fixed proportion parameters, one gets a weighting function \bar{w}_3 that adjusts to the size of the examined problem (namely the parameter count n). The following table shows \bar{w}_3 , again for problem sizes $n = 3$ to $n = 9$. (The values that are not shown are left linear, like before with \bar{w}_1 and \bar{w}_2 .)

parameters n	3	4	5	6	7	8	9
tuples per configuration b	3	6	10	15	21	28	36
$\bar{w}_3(\mathbf{bad})$	[-1.0]	-1.5	-2.7	-4.2	-6.0	-8.1	-10.5
$\bar{w}_3(\mathbf{very\ bad})$	-3.0	-7.5	-13.5	-21.0	-30.0	-40.5	-52.5
$\bar{w}_3(\mathbf{worst})$	-6.0	-15.0	-27.0	-42.0	-60.0	-81.0	-105.0

The main argument for a weighting function that is adjusting to n is, that with a growing n it is easier for a configuration with one **worst** or **very bad** tuple to get a high rating. To safely keep the pool of the first ranking configurations “clean” of such configurations, the weights need to rise (as just shown with the proportion parameter example). However, applying fix weights for all sizes of problems will most of the time also yield satisfying results. After all the goal is a ranking of all configurations by consistency, and the problem with the equidistant approach is mainly that some inconsistent configurations may rank pretty high. Already with rather low and fix weights (fix over n) for the negative designators, this problem will often be well addressed. If we look at the weighting function \bar{w}_2 , it does not satisfy the criteria of \bar{w}_3 (the Holiday Trip problem has six parameters), and still it almost perfectly sorts out the configurations with **worst** and also the ones with **very bad** tuples. Here statistics come into play. It is obviously just not probable to have one **worst** and ten **best** tuples in a configuration of six parameters. (This depends of course on the distribution of the levels in the consistency assessment.) For these characteristics \bar{w}_2 would most probably also work for problems with up to nine parameters.

Conclusion: When it comes to calculating consistency values of configurations out of their member tuples, there are very many adjusting screws. You can have fix weights for the negative designators as for the positive ones, or dynamic weights that change with the size of the problem (for both sides of the range or for just one of the two). In any way the determination of the approach and the calibration of the actual weights (Which exact distances for fix weights? Which proportions for dynamic weights? Which kind of polynomial function should the weights resemble?) should be subject to further research, that especially takes into account psychological moments. (E.g.: How much stronger as a judgement is “very bad” in comparison to just “bad”? How can the tendency to moderate judgements be addressed? Should maybe the moderate judgements “good” and “bad” get relatively strong weights?)

3.5 Scaled consistency assessment and selective TCA

The conclusion of Section 3.2 and Section 3.3 was that general consistency assessment of higher order is no good idea, but that selective triple consistency assessment is a step that in case can yield a good improvement with reasonable costs.

However this improvement (only shown with the Holiday Trip problem) was achieved on the background of binary pairwise CCA. With scaled pairwise CCA the situation is different. A certain hope could be, that configurations containing higher-order inconsistencies (which are not at all detected via binary pairwise CCA) would now at least appear very late in the ranking. The diffuse assumption behind that is a hope for certain semantic interferences that will (for example) make it improbable that a configuration containing only **best** pairs will contain a higher-order inconsistency. The prove that scaled pairwise CCA would usually rank configurations with higher-order inconsistencies very low would not only underline its utility, but would also allow to in good conscience leave out any higher-order CCA.

Unfortunately there are no data available to empirically test the assumption (e.g. with several example problems from various domains). The only reference problem will once more be the Holiday Trip problem. This however does not at all support the assumption. Figure 3.7 shows the frequency distributions of configurations containing at least one inconsistent triple. As one can see, they are pretty equally distributed over the ranking. The average consistency value is 0.43 for \bar{w}_1 and -1.00 for \bar{w}_2 .

To have a look at a specific example: The example configuration $c_2 = ([7,14), \textit{Single dest.}, [300,500), \textit{airplane}, \textit{all-inclusive}, [250,500))$ from above gets a consistency value of 1.66 (with both weighting functions), while containing the two inconsistent triples $t_1 = ([7,14), \textit{airplane}, [250,500))$ and $t_2 = ([7,14), \textit{all-inclusive}, [250,500))$.

So, doing the pairwise CCA with scaled assessments does not (or at least not in all cases) influence the utility of the selective TCA. Which leads to the question, how the information of the selective TCA would best be integrated into the ranking. The easy way is to leave the TCA binary (just assessing the triples by **consistent/inconsistent**) and to delete all configurations comprising at least one inconsistent triple out of the ranked

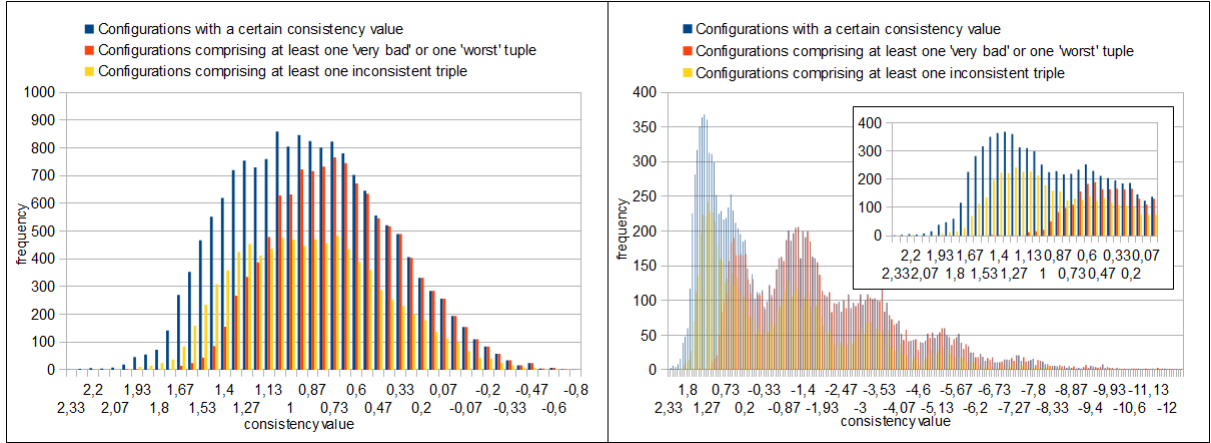


Figure 3.7: Frequency distributions of configurations containing at least one inconsistent triple (for the Holiday Trip problem). *Left*: the equidistant approach \bar{w}_1 . *Right*: the weighted approach \bar{w}_2 . *Blue*: the count of configurations having a certain consistency value. *Red*: the count of configurations having a certain consistency value while comprising at least one **very bad** or one **worst** tuple. *Yellow*: the count of configurations having a certain consistency value while comprising at least one inconsistent triple.

configuration space. The drawback of this approach is losing one of the advantages of ranking the configurations instead of deleting the “bad” ones: namely that *all* configurations remain available for further examination.

The alternative is to enhance the concept of the weighting function. Why not just calculate the values of the triple assessments into the consistency value?

For the moment let the weighting function \bar{w} that assigns the numerical values to the Likert-levels be the same for the triples. The triples that have not been selected for consistency assessment would all get the assessment “**maybe**” and therefore in our cases the consistency weight 0.0. As very many of the triples might not have been assessed, calculating the average of the consistency values of the involved tuples *and* the involved triples would compress the consistency values of the configurations around zero. Instead one approach could be to add a correction value to the primary consistency value. For a configuration $c \in \mathcal{CS}$ comprising the tuple set T_c and the triple set R_c the enhanced consistency value $v^*(c)$ could this way for example be calculated as follows:

$$v^*(c) = \frac{1}{|T_c|} \cdot \sum_{t \in T_c} w(t) + \frac{1}{|R_c|} \cdot \sum_{r \in R_c} w(r), \text{ with } |T_c| = \binom{n}{2} \text{ and } |R_c| = \binom{n}{3}. \quad (3.2)$$

Thereby $w(t)$ and $w(r)$ are the consistency values of the tuple t or the triple r which correspond to the numerical values that have been assigned to the consistency level of t or r by the weighting function \bar{w} .

With this approach it would obviously also be possible to integrate *scaled* triple assessment. For a certain configuration the non-assessed triples would make no contribution,

very good triples would increase the correction factor and very bad triples would reduce it - corresponding to the weights that have been assigned. The arising problem is the calibration of the weights that gets even more complicated than it was with the tuples-only calculation: Should the weighting function be the same for tuples and triples? Can a configuration have a bad *tuple* score, but a good *triple* score (correction factor), so that it still ranks rather high? Is the tuple score already a sufficient indicator for the overall consistency and should the triple assessments maybe only correct it into the negative (so that the user could only assess triples with grades of negative)? In which relation are the importance of the tuple assessment and the importance of the triple assessment? Do these calibrations depend on the semantic structure of the specific problems?

Obviously the goal should be to push back the configurations comprising an **inconsistent** (or **worst**) triple on the ranking of configurations. How far they should be pushed back in relation to the ones only containing **very bad** or **worst** tuples is a question that needs further inquiry.

Using Equation 3.2 with the weighting function \bar{w}_2 for the tuples as for the triples and recurring to the binary assessment from Section 3.3 (with “consistent” being translated to “maybe” and “inconsistent” to “worst”) the frequency distributions from Figure 3.8 emerge.

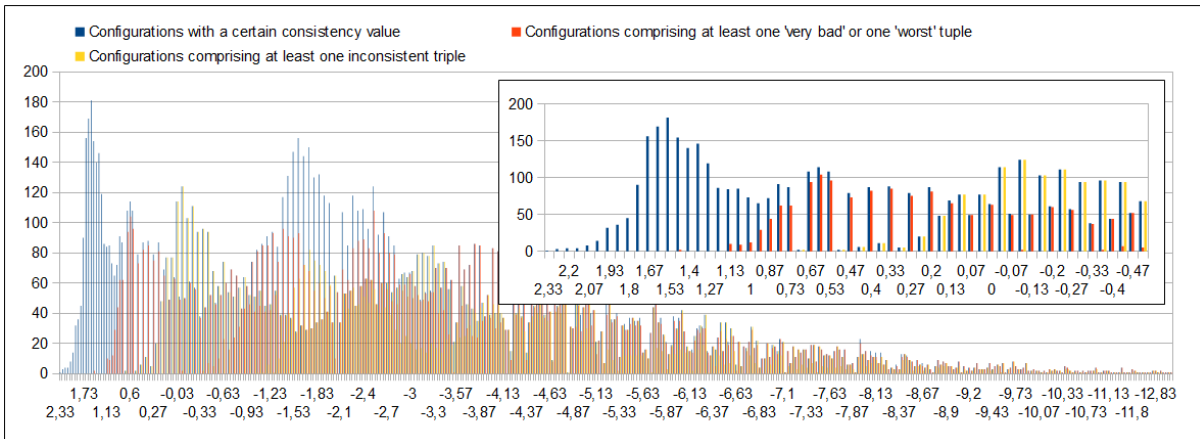


Figure 3.8: Consistency calculation with triple correction factor (Holiday Trip problem, weighting function \bar{w}_2). *Blue*: the count of configurations having a certain consistency value. *Red*: the count of configurations having a certain consistency value while comprising at least one **very bad** or one **worst** tuple. *Yellow*: the count of configurations having a certain consistency value while comprising at least one inconsistent triple.

As the Holiday Trip problem has six parameters and therefore one configuration comprises 20 value triples, the correction factor for a configuration with one inconsistent triple is $\frac{1 \cdot (-30) + 19 \cdot 0}{20} = -1.5$ (with \bar{w}_2) which (judging from Figure 3.8) seems quite adequate for the distributions of this specific problem.

There are two working basis that are chosen for the rest of this thesis. One is keeping all configurations in the ordered configuration space and calculating the consistency values via Equation 3.2 with the triple correction factor. The other one is deleting configurations

with inconsistent tuples or triples from the ordered configuration space. In both cases the triple assessment is left binary.⁶ In the second case, configurations containing a **very bad** or **worst** tuple or an **inconsistent** triple are directly removed from the configuration space. This second approach gets its justification by the fact that the calibration question is not really solved and therefore there could be confusion about the real meaning of the ordering of the configurations. The examination of the whole configuration space will not play an important role and after all the now deleted configurations are *inconsistent* and are no viable solutions to the considered problem. The remaining configurations are put in order via the original consistency value function (without triple correction factor).

Quite some time has now been spent to usefully reduce and order the formal configuration space. The output is meanwhile an ordered list with the most promising configurations at the top (see Table 3.2). But this list can still contain a huge number of configurations. Without deleting configurations it's still the full formal configuration space, but also when deleting the "inconsistent" configurations there can be many remaining. In the case of the Holiday Trip problem the reduced configuration space contains 1786 ordered configurations.⁷ Manually examining these 1786 configurations as a list and doing that thoroughly, so that no information is lost, would still be quite a challenge.

- | |
|--|
| <ol style="list-style-type: none"> 1. ([0,1), Single destination, [0,50), none (walking), none, [0,100)) 2. ([14,30], Single destination, >1000, airplane, all-inclusive, >1000) 3. ([14,30], Single destination, [500,1000], airplane, all-inclusive, >1000) 4. ([0,1), Single destination, [0,50), bicycle, tent, [0,100)) 5. ([14,30], Single destination, >1000, airplane, all-inclusive, [500,1000]) 6. ([14,30], Single destination, [500,1000], airplane, all-inclusive, [500,1000]) 7. ([0,1), Only roadtrip, [0,50), bicycle, tent, [0,100)) 8. ([0,1), Single destination, [0,50), car, tent, [0,100)) 9. ([0,1), Single destination, [0,50), bicycle, none, [0,100)) 10. ([0,1), Single destination, [0,50), none (walking), tent, [0,100)) 11. ([0,1), Only roadtrip, [0,50), car, tent, [0,100)) 12. ([0,1), Only roadtrip, [0,50), none (walking), none, [0,100)) 13. ([0,1), Single destination, [0,50), car, none, [0,100)) 14. ([0,1), Single destination, [0,50), train, tent, [0,100)) 15. ([14,30], Single destination, >1000, airplane, Air BnB, >1000) 16. ... |
|--|

Table 3.2: The 15 configurations ranked best in the sorted configuration space of the Holiday Trip problem.

⁶Assessing triples is already a more complex task than assessing tuples and will only be made even more costly with scaled assessment.

⁷It's the same number as at the end of sSection 3.3 because the **very bad** *and* the **worst** tuples have been translated to being **inconsistent** there (confer Figure 2.2 and Figure 3.5).

4. Computer support for GMA - known approaches

Naturally the idea comes to mind to use computer support to further develop the GMA procedure, to structure the configuration space, to gain more insights into the problem - to make more information accessible. After all, the Cartesian product which is the configuration space is just too big to be handled by a human mind without any help. So obviously the calculation of the configuration spaces, of tuples and triples, of the consistency values etc. has already been done with the computer. But with computer support there might be ways to further process the (reduced) configuration space or to present it to the user in a way that he can grasp more of the contained information. The rest of this thesis will describe a prototype (and its theoretical backgrounds) that has been developed for this purpose. But first there shall be presented two already existing approaches which were accessible for the writing of this thesis.

4.1 MA/Carma

The MA/Carma software suite of Dr. Tom Ritchey is well described in [Rit03], but screenshots of the graphical surfaces of morphological fields and cross consistency matrices appear in most of his publications (so also in all of the other publications of his that are cited in this thesis). MA/Carma is based on Ritchey's theoretical work, so it works on the background of the standard GMA procedure with binary pairwise CCA (as described in Section 2.2.1). The formal configuration space is reduced by all configurations that contain at least one inconsistent value pair, and then this reduced (but apart from that unstructured) configuration space is being made explorable with the support of MA/Carma. To this purpose, a depiction of the morphological field serves as an *inference model*.

MA/Carma and its use shall be illustrated with the example problem and the corresponding figures from [Rit03]. The problem is derived from a study that Tom Ritchey and his

colleagues did for the Swedish National Rescue Services Agency with which Sweden's preparedness for chemical accidents and chemical terrorist attacks has been examined. The morphological field is conceptually divided into a five-parameter "resource"-field and a three-parameter "response"-field. (As Ritchey states, "the scenario is in fact based on an actual accident in Sweden involving the release of ammonia caused by a railroad accident.")

Figure 4.1 shows the whole morphological field of this Ammonia Accident problem in MA/Carma (the so-called *Edit Field*), Figure 4.2 respectively shows the filled cross consistency matrix in MA/Carma (the so-called *Cross Consistency Field*). In the Edit Field the morphological field can be formatted (e.g. re-sized or color-coded). All cells - in the Edit Field as well as in the Cross Consistency Field - are backed up with a corresponding text field where descriptions (e.g. about the parameters or values), illustrating examples, comments, or justifications for the consistency judgements can be stored. The little red dot in the upper right corner of the cell shows if the corresponding text field has some content.

The screenshot shows a software window titled "Carma 2.02r - [PTGas2.scn - Ammonia accident (Sweden)]". The window contains a table with 8 columns and 6 rows. The columns are: PLANNING/ PLANS, TRAINING AND EDUCATION, PERSONNEL AVAILABLE, EQUIPMENT AVAILABLE, COMMAND LEVEL, RESPONSE to chemical release, RESPONSE: Information to public, and RESPONSE: Affected people. The rows represent different levels of preparedness: Full municipal preparedness plan, Response plan for specific case, Standard routine for specific case, Standard routine for general case, and Only alert plan. Each cell contains specific details about the parameters and response times for that scenario.

PLANNING/ PLANS	TRAINING AND EDUCATION	PERSONNEL AVAILABLE	EQUIPMENT AVAILABLE	COMMAND LEVEL	RESPONSE to chemical release	RESPONSE: Information to public	RESPONSE: Affected people
Full municipal preparedness plan	Broad co-op. training	11 or more	Special equipment for specific case	Command level 4	Reduce by at least 80% within 15 min	Warn involved within 5 min	Help many within 30 min
Response plan for specific case	Training for specific case	8-10	Base equipment for specific case	Command level 3	Reduce by at least 80% within 30 min	Warn involved within 30 min	Help some individuals within 15 min
Standard routine for specific case	Base education + regular training	5-7	Less than base equipment	Command level 2	Reduce by less than 50% within 15 min	No warning within 30 min	Help some individuals within 30 min
Standard routine for general case	Base education only	4 or less		Command level 1	Reduce by less than 50% within 30 min		No help within 30 min
Only alert plan					No measures within 30 min		

Figure 4.1: Morphological field of the Ammonia Accident problem in MA/Carma (adapted from [Rit03]).

Figure 4.2: Cross consistency matrix of the Ammonia Accident problem in MA/Carma (adapted from [Rit03]).

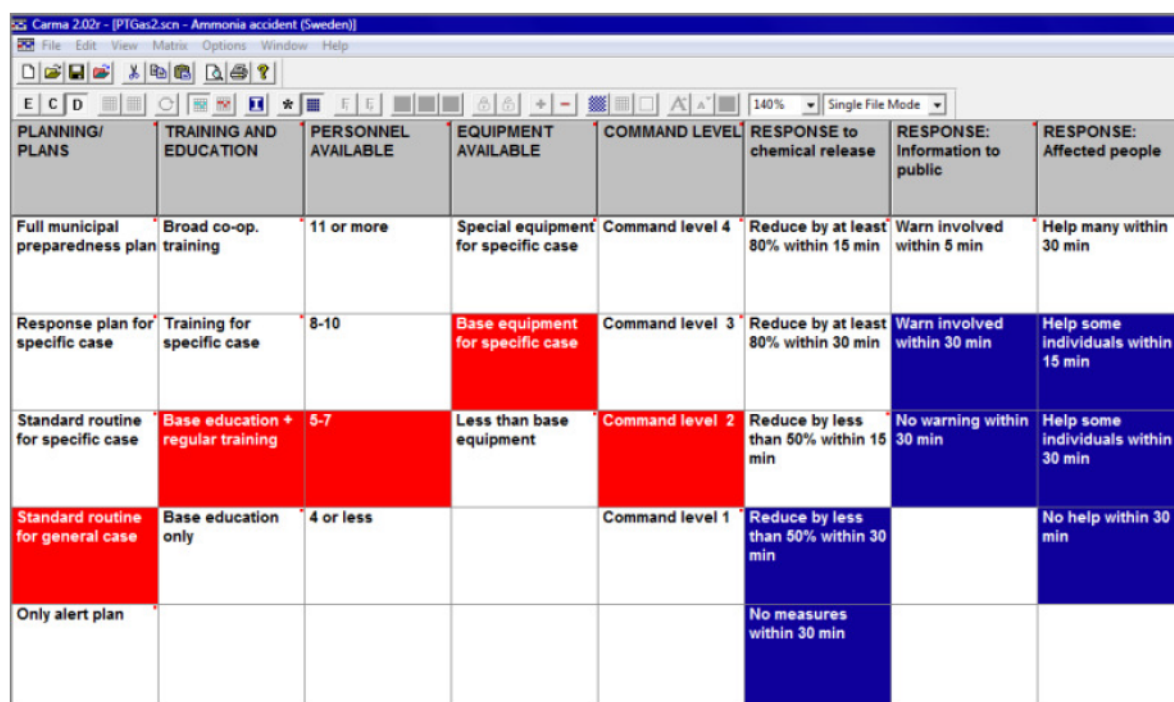
Figure 4.3 shows the so-called *Display Field*, the third kind of field in MA/Carma, which makes the reduced configuration space interactively explorable.

The Display Field shows the morphological field of the problem, just like in the Edit Field, but now - with a click on a value of a parameter - this parameter value can be fixed (or unfixed) and turns from blue to red (or of course the other way round). Behind the graphical surface lies the configuration space that has been reduced via the pairwise inconsistencies from the Cross Consistency Field.

Initially all parameter values are colored blue. If a parameter value is fixed, all values of all other parameters that are inconsistent with this fixed parameter turn from blue to white. (Also all other values of the same parameter turn to white.) That way the user can easily see which values of the other parameters are compatible with his selection. In Figure 4.3 for each of the “resource”-parameters of the problem, one value has been fixed. With this selection the values of the “response”-parameters that are still colored blue are still compatible. That means that there are configurations in the reduced configuration

space that comprise the five fixed parameter values, as well as one of the blue values for each of the “response”-parameters. The white values of the “response”-parameters are inconsistent with at least one of the fixed values. As Tom Ritchey puts it, “the field displays a given input (in red) concerning a rescue service’s preparedness resources for a particular chemical accident, and output (in blue) showing the level of response judged to be associated with this preparedness level.”[Rit03]

The benefit of this basic computer support should not be underestimated. With MA/Carma the reduced configuration space can be explored interactively. The user can freely fix one or more values of one or more parameters and see which other values are compatible with his selection. He can try various selections one after another, freeze a certain selection and compare it with another one. Anything can be input, anything can be output. In the Ammonia Accident problem for example, one can also give the wished responses as input and see which resources are necessary to reach this response level. With this interactive “inference model” as Ritchey calls it, the user can ask any kind of “what-if”-question and get an immediate graphical feedback as an answer. Not only are all kinds of “what-if”-questions comfortably answered without manually having to go through all configurations of the reduced configuration space; working for some time with the Display Field will furthermore give the user some feeling for the structure of the reduced configuration space and thereby more understanding of the problem as it has been modelled in the GMA process.



PLANNING/ PLANS	TRAINING AND EDUCATION	PERSONNEL AVAILABLE	EQUIPMENT AVAILABLE	COMMAND LEVEL	RESPONSE to chemical release	RESPONSE: Information to public	RESPONSE: Affected people
Full municipal preparedness plan	Broad co-op. training	11 or more	Special equipment for specific case	Command level 4	Reduce by at least 80% within 15 min	Warn involved within 5 min	Help many within 30 min
Response plan for specific case	Training for specific case	8-10	Base equipment for specific case	Command level 3	Reduce by at least 80% within 30 min	Warn involved within 30 min	Help some individuals within 15 min
Standard routine for specific case	Base education + regular training	5-7	Less than base equipment	Command level 2	Reduce by less than 50% within 15 min	No warning within 30 min	Help some individuals within 30 min
Standard routine for general case	Base education only	4 or less		Command level 1	Reduce by less than 50% within 30 min		No help within 30 min
Only alert plan					No measures within 30 min		

Figure 4.3: Display Field showing fixed parameter values (red) and the parameter values that are compatible with this combination of fixed parameter values (blue) (adapted from [Rit03]).

4.2 Parmenides EIDOS

The Parmenides EIDOS suite of the Parmenides Foundation (successor of the former Think Tools AG) has already been mentioned above - as being a software suite that implements GMA with scaled binary consistency assessment. Indeed GMA is a central component of EIDOS, but still only one out of several components that all arise from the greater idea behind EIDOS and the Parmenides Foundation: to develop and implement tools that support human thinking processes.

While human reasoning skills are rather constant, it is fair to say that our world gets more and more complex - and so do the decisions that have to be taken in this world (as for example by strategy planners in business or politics). With the motivation of researching complex human thinking, the Parmenides Foundation that works in close cooperation with the Ludwig-Maximilians-Universität (LMU) Munich has brought together around 40 scientists from the most various disciplines - all being concerned with human cognition and thinking.

Some of the findings have been translated to elements of a computer application (Parmenides EIDOS) that aims at improving the thinking and decision making of executives both in the private as well as in the public sector. An integral element of EIDOS is however an implementation of GMA with additional computer support and this shall be outlined in the following.

The meta-information stated here (and some more of it) can be found on the website of the foundation ([Fou15b]) or in a background information brochure they published ([Fou15a]). While being careful with publications about the EIDOS software suite, the Parmenides Foundation generously provided a temporary license of EIDOS for the purpose of this thesis. Therefore the rest of this section can give no theoretical background information, but will describe the main elements of GMA part of the tool.

4.2.1 Classical functions and sorted configurations

Embedded in a larger, guided process of planning a strategy (defining goals, examining scenarios, finding strategies, evaluating risks, ...) to address a certain problem, Parmenides EIDOS also implements classical GMA (as described so far in this thesis), but graphically enhanced and with further computer support for the exploration of the configuration space. In the EIDOS suite this step is used to screen for possible scenarios and to develop suitable strategies. To give a first impression of the looks of the program, Figure 4.4 shows a morphological field of an example problem and Figure 4.5 shows the corresponding cross consistency matrix as they appear in EIDOS. The example problem is analyzing the possible future scenarios of the EU petrochemicals market.

As described in Section 2.2.2 and Section 3.4, EIDOS uses the integral consistency assessment data to calculate the average consistency value of each configuration. Then the configuration space can be sorted by consistency. Figure 4.6 shows how EIDOS presents the sorted configuration space to the user: there is no list of the whole configurations (like in Table 3.2), but a list of their corresponding positions in the sorted configuration space and their average consistency value (on the left of the window). In Figure 4.6

the configuration at position 11 and with the average consistency value of 1.33 has been selected. As one can see on the right (highlighted in light blue), this is the configuration $c_{11} = (OPEC\ dominance, Going\ with\ the\ supply\ of\ feedstock, Massive\ import, More\ plastic\ used, Cash\ cows, Serious\ challenge)$. The advantage of this approach is that the user always overlooks the whole morphological field and thereby also all the alternative parameter values.

Parmenides EIDOS allows to examine the first x configurations (with x being adjustable by the user). It is possible to lock a parameter value (only one per parameter). Then - in the view with the sorted configuration space (like Figure 4.6) - there are only shown configurations containing this specific value. That way a value can be fixed, like in MA/-Carma. Then the top x configurations containing this value can be examined in a newly calculated window - with going through the separate configurations in the sorted list.

Figure 4.7 shows the EIDOS frame with two parameter values locked. Of the first two columns the value “Surplus of feedstocks” and “Going with the supply of feedstock” have been locked (and are marked with an orange point at the bottom left of the value panels). So in the list on the left there are only the first 100 of the in total 108 configurations that contain these two values.

An important remark: As repeatedly mentioned, EIDOS calculates the consistency values of configurations with the equidistant weighting function. But EIDOS does not neglect the risk of having configurations with inconsistent tuples among the best-rating. Whenever a configuration is selected, small bars in the parameter panels show if there are inconsistent tuples comprised.

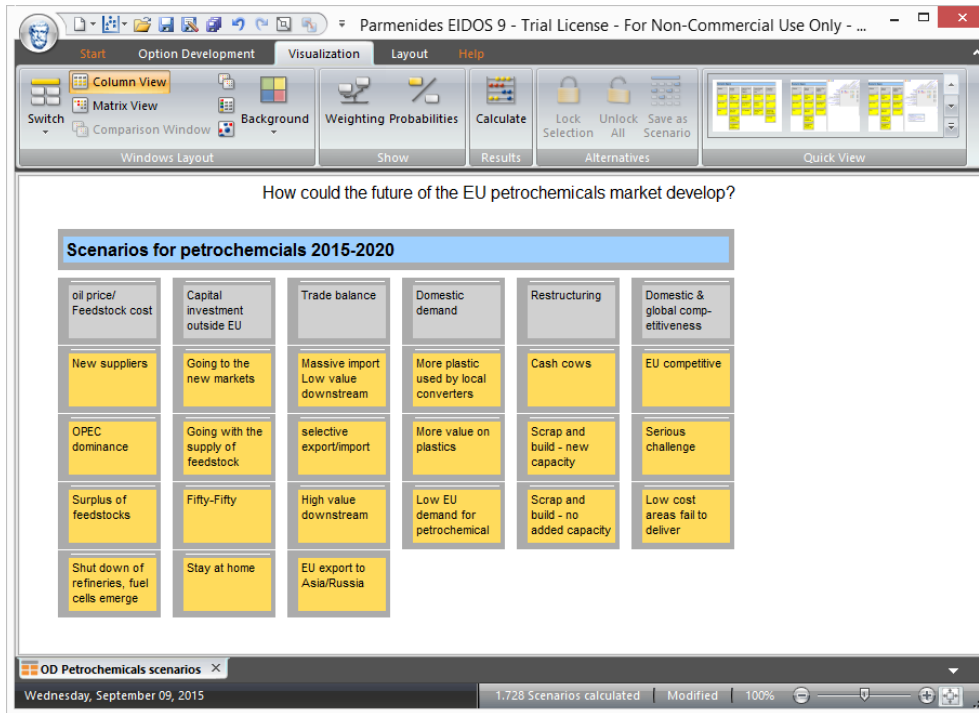


Figure 4.4: Morphological field of the Petrochemicals Scenarios problem in Parmenides EIDOS.

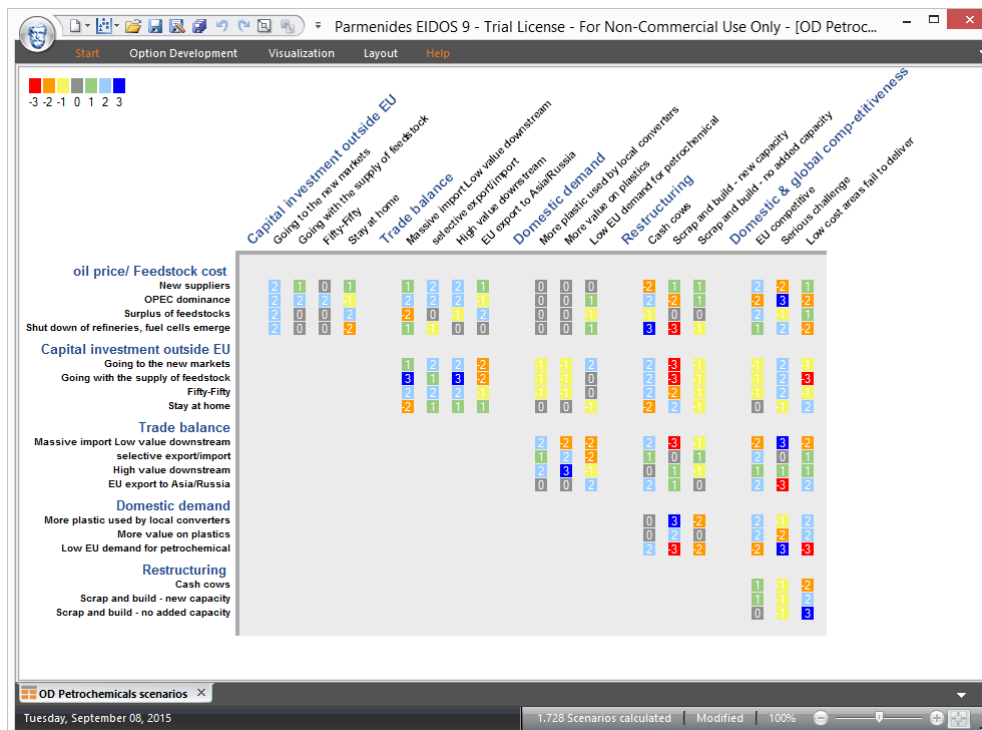


Figure 4.5: Cross consistency matrix of the Petrochemicals Scenarios problem in Parmenides EIDOS.

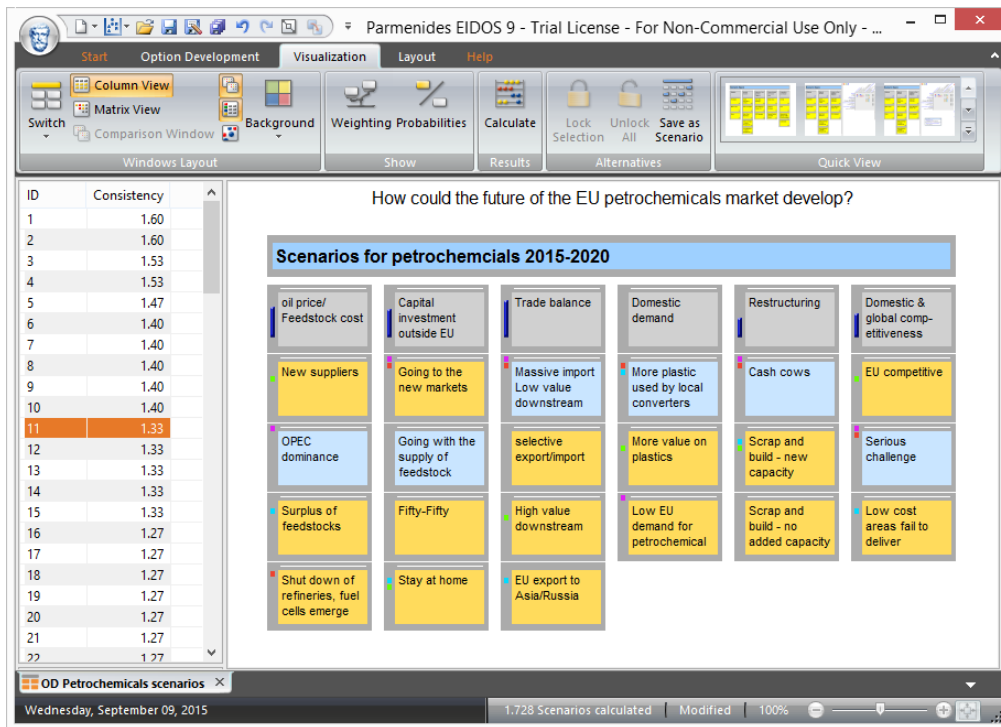


Figure 4.6: Presentation of the sorted configuration space in Parmenides EIDOS.

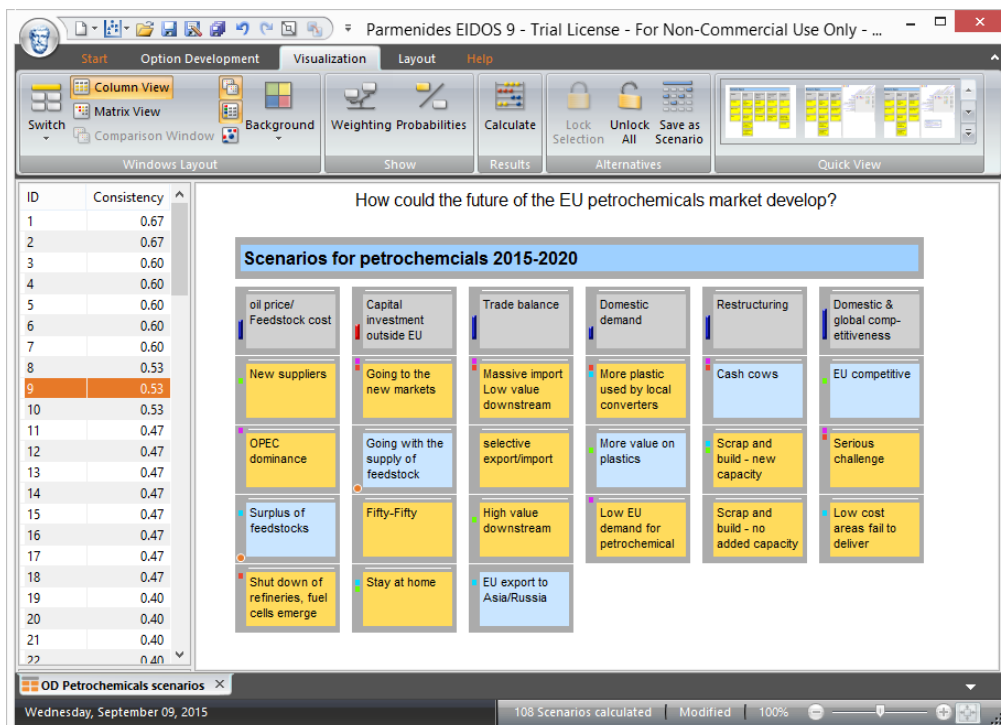


Figure 4.7: Presentation of the sorted configuration space in Parmenides EIDOS (two parameter values locked).

4.2.2 Cluster visualization of the configuration space

The aim of GMA as a step in the EIDOS process is to identify the consistent future scenarios and, respectively, consistent strategies that can be taken. Therefore the configurations with the highest consistency values are interesting to have a look at - but among these especially the configurations that are dissimilar from one another. To explain that with an example: if we talk about future scenarios, it is decisive to find out all different scenarios that represent the different rough directions into which the present situation could consistently develop.¹ To this purpose Parmenides EIDOS introduces a great enhancement to the classical GMA: a so-called *Cluster View*. Figure 4.8 shows this Cluster View as it is integrated into the EIDOS surface.

In the Cluster View each of the first 100 configurations is represented as a small blue bubble. The higher the consistency value of the configuration, the bigger the diameter of the bubble. In Figure 4.8 the mouse had just entered the bubble with the number 14 and therefore the configuration c_{14} is highlighted.

Now the central concept of the Cluster View: as it is stated in the legend, the representations of similar configurations lie close to each other, the representations of dissimilar configurations appear apart from each other. That way the user can not only go through the list of sorted configurations, but can have a look at different configurations from different parts of the map (bearing in mind that all of the depicted configurations are among the most consistent) and that way examine the different *kinds* of valuable solutions.

In Figure 4.8 four different relevant and representative configurations have been identified and saved by the user. And this group of distinctive configurations will then play a role later on in the EIDOS Strategizing Framework. Figure 4.9 shows how one of the saved configurations can be selected and how it is then highlighted in the morphological field. Visualizing the configuration space is definitely a very promising idea. After all, the configuration space is reduced and/or sorted by the consistency assessment, and then it's all about structuring and presenting - so that the user can grasp as much information as possible about the properties of the reduced configuration space. If - as EIDOS explains - similar configurations lie closer to each other, the clusters that are visible in the Cluster View are groups of similar configurations that can be detected in the configuration space. In Figure 4.8 for example, the user can principally see two large groups, one at the left and one at the right. If the Cluster View makes sense and the reduced configuration space is really divided into two pretty distinct halves, then this visually easily accessible information would rarely have been found by a user only going through a list of configurations. So, visualizing the configuration space and doing that with two-dimensional clusters of configurations seems to be a promising approach and in case a significant and great enhancement to GMA.

¹For example could a small farm develop into a bigger farm, a hobby farm, a holiday farm - or be sold. All of them are possible scenarios and will therefore have a rather high consistency value. But maybe the first twenty configurations in the sorted space all represent slight variations of the holiday farm concept. It is important not to miss the others.

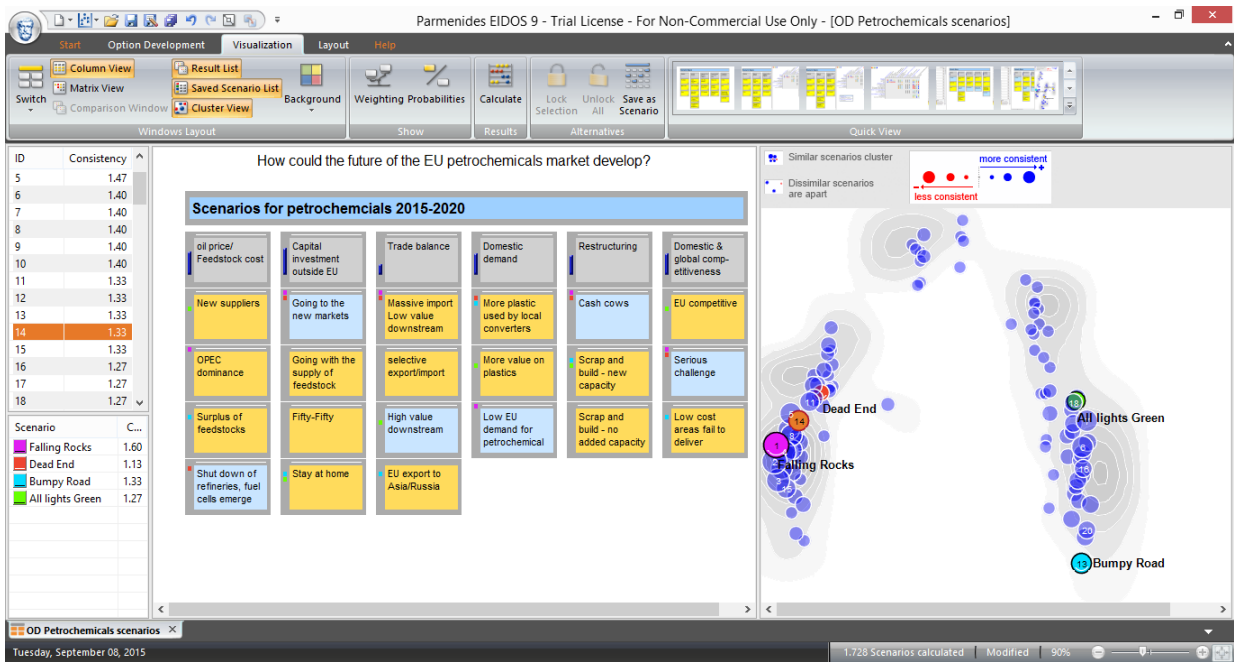


Figure 4.8: Parmenides EIDOS with the Cluster View of the Petrochemicals Scenarios problem (mouse just entered bubble number 14).

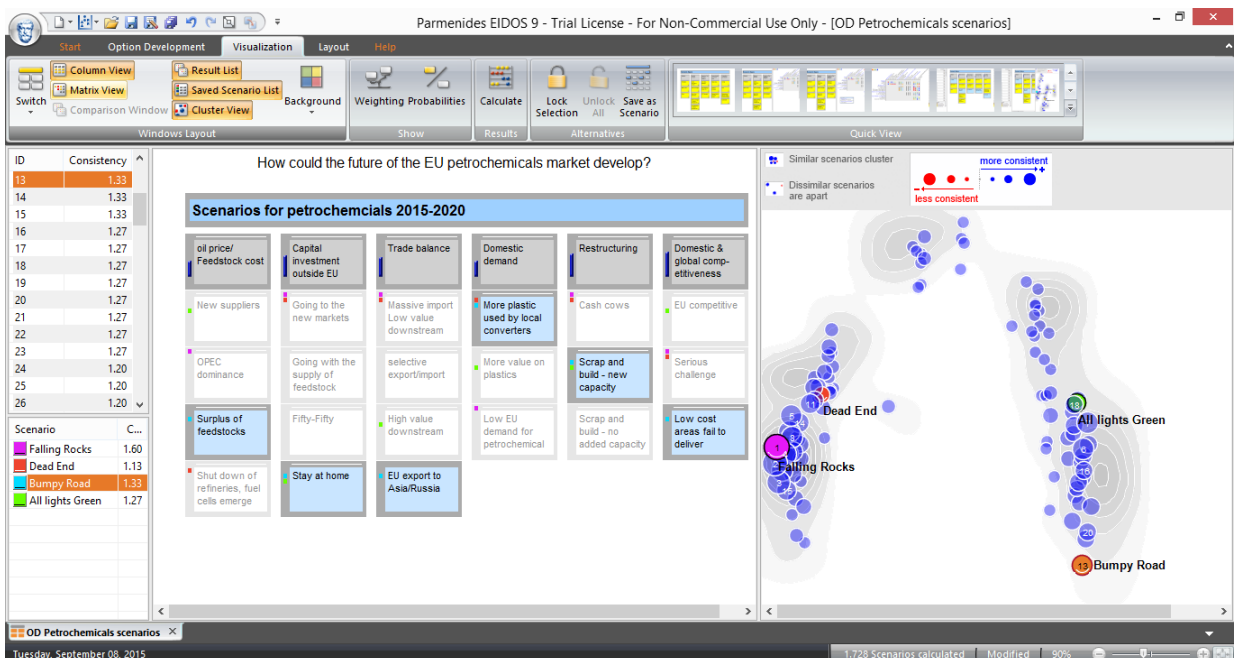


Figure 4.9: Parmenides EIDOS with the Cluster View of the Petrochemicals Scenarios problem (the previously saved “Bumpy Road” scenario has been selected).

5. SIM - an inference model with scaled consistency data

When it comes to thinking about additional computer support for GMA, a pretty obvious idea is, to enhance Ritchey's inference model with scaled consistency data. MA/Carma already is a powerful exploring tool on the basis of binary consistency data. Showing which values are consistent with the current selection and just uncoloring the others gives good impressions of the structure of the reduced configuration space. Enhancing this idea with scaled consistency data does not really change the experience and does not bring any fundamental enhancements. However, it can indeed bring some small improvements. In the following a prototype shall be presented and explained, that implements the described enhancement. It will be referred to as Scaled Inference Model (SIM).

When sticking to the MA/Carma design of showing the morphological field as an interface, the obvious idea is to color the value panels respective to their consistency value. To calculate the consistency value of a parameter value (depending on which values have been fixed), there are two options: Either display the average consistency value of the *configurations* that are still in the selection-reduced configuration space (Option 1). Or display the average consistency value of the *tuples* that the considered value forms with the selected/fixed values (Option 2). To formalize that:

Let there be n parameters P_i with value sets V_i . With a click on the panel of a certain value, that value is fixed. At a certain moment, let V_f be the set of fixed values. The configuration space \mathcal{CS} can then be reduced to \mathcal{CS}^* by all configurations that don't contain one of the fixed values (on the respectively considered parameter). The consistency value $\bar{v}_1(u)$ of some unfixed value u (from a parameter with no fixed values) is then calculated as follows (via Option 1):

$$\bar{v}_1(u) = \frac{1}{|C|} \cdot \sum_{c \in C} v(c) , \text{ with } C \subseteq \mathcal{CS}^* \text{ with } \forall c \in C : u \in c. \quad (5.1)$$

Via Option 2 the consistency value $\bar{v}_2(u)$ of a parameter value in the field would be calculated as follows ($\mathbf{t}(x, y)$ is a tuple built from value x and value y , w is the weighting function from Equation 3.1):

$$\bar{v}_2(u) = \frac{1}{|V_f|} \cdot \sum_{v_f \in V_f} w(\mathbf{t}(u, v_f)). \quad (5.2)$$

An advantage of Option 1 is, that the averages can already be calculated with no parameter value fixed - as Figure 5.1 shows. As one can see in the example, the configurations containing “none” as accommodation and the configurations containing “airplane” as transportation score the lowest average consistency values. (Admittedly this is not a too meaningful information.)

Figure 5.2 shows the same field with three fixed values. The left subfigure is calculated via Option 1, the right subfigure via Option 2. One can see: with a single destination and a journey of a length between one and seven days, it is for example most consistent to sleep in a tent, a hostel or a hotel, and so on. One can also see: Even though the calculation procedures are different, the results are pretty similar. (It is a very common observation that the consistency values only get shifted.)

Figure 5.3 illustrates the decisive difference between the two calculating options: A duration of less than 24 hours and a travel distance of more than 1000 km have been chosen. Obviously the configurations containing these two values have very, very low consistency values (Option 1, on the left side). The average consistency value of the two respective tuples (comprising each the considered value and one of the selected values) however does not necessarily have to be terribly low (Option 2, on the right side).

This can be an advantage in case that the user just absolutely wants to try to combine two rather inconsistent values. Then Option 1 will just tell him that the combination doesn’t make sense, while Option 2 would still deliver a little information about which other values fit best with the two selected values.

However it is obviously questionable if the user should even have the opportunity to select two inconsistent values (like in the last example). To prevent him from doing so, the configuration space can in advance be reduced by all configurations comprising **very bad** or **worst** tuples (or **inconsistent** triples). This makes the approach even more similar to MA/Carma. It improves the meaningfulness of the Option 2-model, but in return the Option 1-model gets less useful: as all configurations with bad tuples have been deleted, only configurations with rather good scores remain. Taking the average does the rest and so the consistency values often lie pretty close to each other. Figure 5.4 illustrates this.

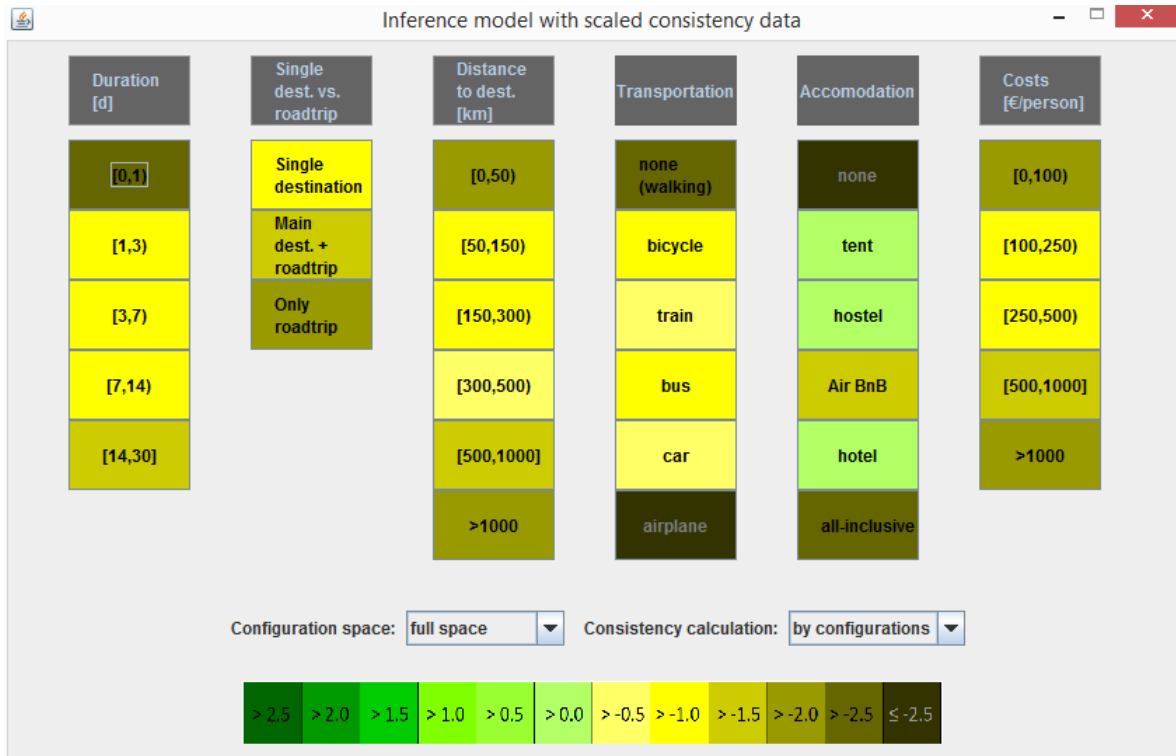
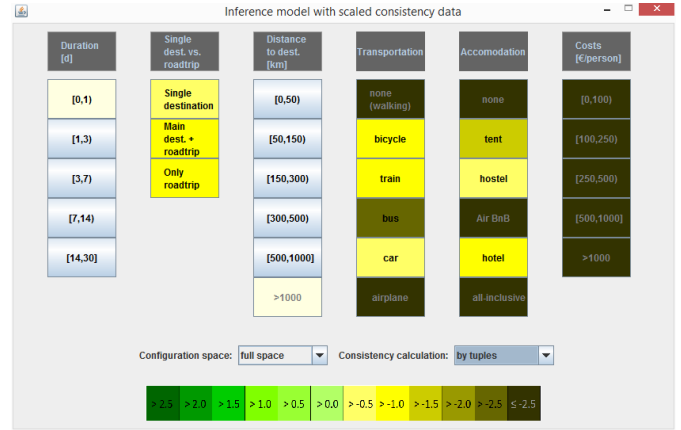
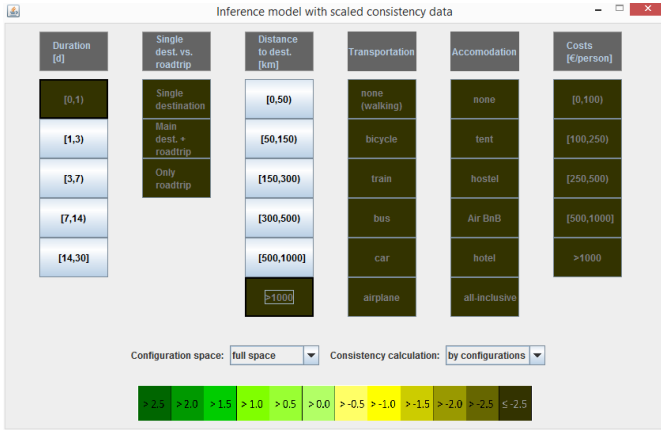


Figure 5.1: Scaled Inference Model with the full configuration space as underlying model and the Option 1-calculation (by configurations). No parameter values fixed.



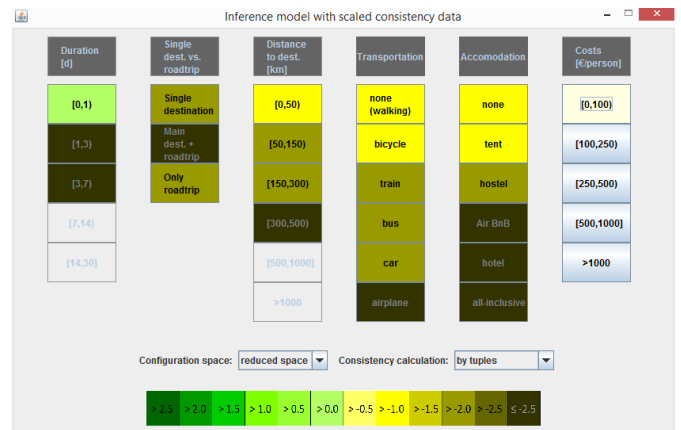
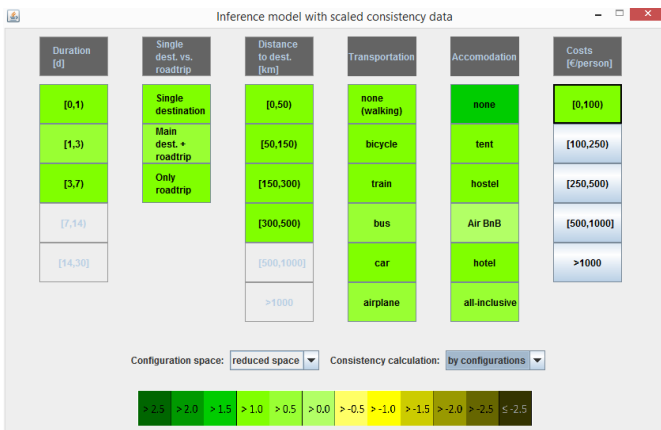
(a) Calculation of color-coded configuration values by Option 1 (by configurations). (b) Calculation of color-coded configuration values by Option 2 (by tuples).

Figure 5.2: Scaled Inference Model with the **full** configuration space as underlying model. Values “[1,3]” and “[3,7]” of the parameter “Duration” and value “Single Destination” of the parameter “Single dest. vs. roadtrip” fixed.



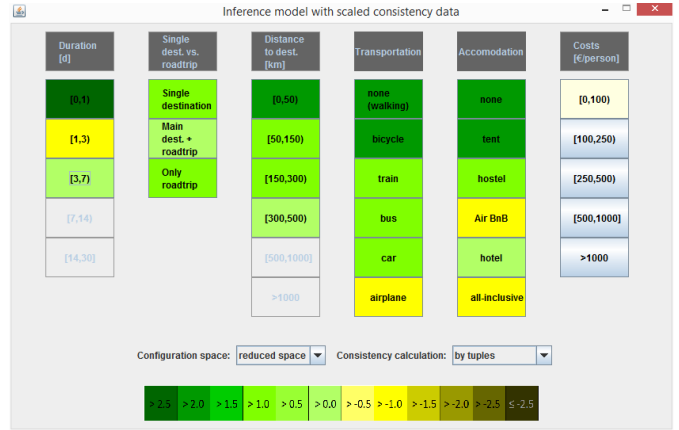
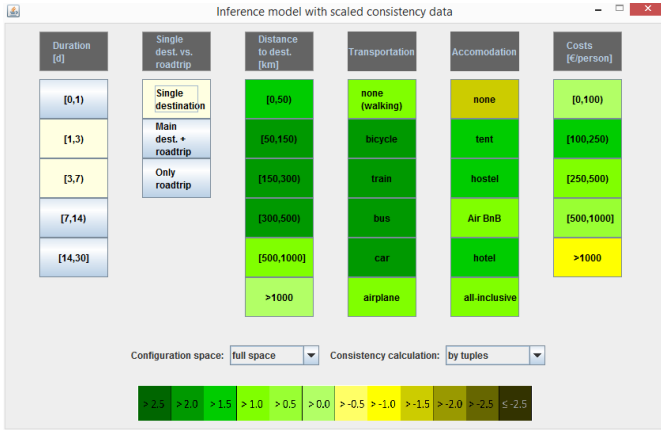
(a) Calculation of color-coded configuration values by Option 1 (by configurations). (b) Calculation of color-coded configuration values by Option 2 (by tuples).

Figure 5.3: Scaled Inference Model with the **full** configuration space as underlying model. Value “[0,1)” of the parameter “Duration” and value “>1000” of the parameter “Distance to dest.” fixed.



(a) Calculation of color-coded configuration values by Option 1 (by configurations). (b) Calculation of color-coded configuration values by Option 2 (by tuples).

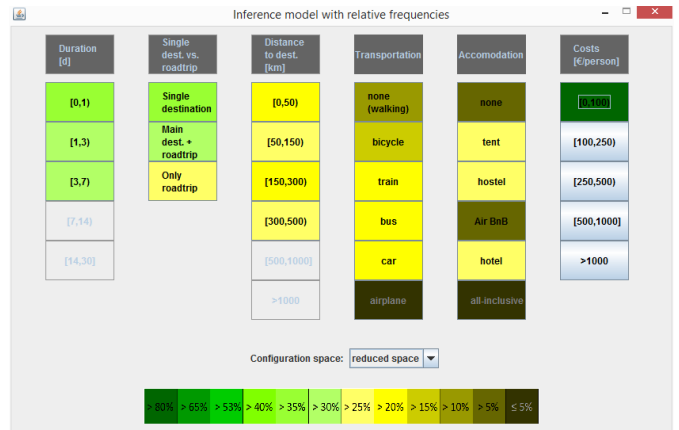
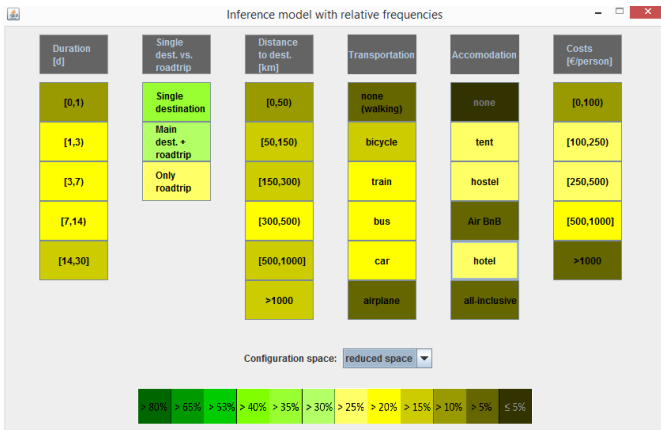
Figure 5.4: Scaled Inference Model with the **reduced** configuration space as underlying model. Value “[0,100)” of the parameter “Costs” fixed.



(a) Replicate of figure 5.2b, only with equidistant weighting for the consistency calculation.

(b) Replicate of figure 5.4b, only with equidistant weighting for the consistency calculation.

Figure 5.5: Weighting function for Option 2: Scaled Inference Model with calculation of color-coded configuration values via Option 2 (by tuples), with the **equidistant weighting function** \bar{w}_1 .



(a) Frequency distributions in the reduced configuration space (reduced by very bad and worst tuples).

(b) Frequency distributions of the subset of the reduced space with the value "[0,100]" in the parameter "Costs".

Figure 5.6: Inference model over the reduced configuration space showing **relative value frequencies** instead of consistency values.

Scaled Inference Model and the weighting function

In all figures that have been described so far the weighting function \bar{w}_2 from Section 3.4 has been used to calculate the color-coded consistency values of parameter values (so a weighted and not the equidistant approach). So also for the calculation of the average consistency values of the tuples of Option 2, the **very bad** tuples have been weighted with -10 and the **worst** tuples have been weighted with -30 . That explains the low average scores of the Option 2-figures so far.

In practice it turns out that Option 1 does only yield good results with a weighted approach. Otherwise the averages of the consistency values usually lie so close together that with steps of 0.5 no differences in the coloring are discernible. Option 2 however does work pretty well also with the equidistant approach. To illustrate this, Figure 5.5 shows the subfigures 5.2b and 5.4b - but with the color-coding calculated with the equidistant weighting function \bar{w}_1 .

Conclusion: The weighted approach for the consistency calculation is always preferable anyway (confer Section 3.4). In principle, the equidistant approach can however also be used for a scaled inference model. In case of the Option 1-calculation however, the intervals of a color-coding would therefore have to be smaller and/or not equally distributed.

Inference model with relative frequencies

The idea of coloring the value panels corresponding to the values' relative frequency in the reduced configuration space is also not useless. At least it brings some insight into the structure of the reduced space (namely the frequency distributions of the values). The actual goal of GMA is to find the *consistent* solutions (and principally not the most frequent). But if the configuration space is reduced by the inconsistent configurations, the frequencies in the reduced space do indirectly indicate the consistencies. Figure 5.6 shows how this can look like. Figure 5.1 and figure 5.6a have a pretty similar appearance, even though one shows consistency values and the other frequencies in the reduced space. But one can not rely on this phenomenon, the two concepts rest fundamentally different. Comparing figure 5.4b/5.5b to figure 5.6b gives an impression. In the end the frequencies are not what one is principally looking for. They may help understand the structure of the reduced configuration space, but in the end the goal is to find distinct, consistent problem solutions - and therefore one should rather concentrate on the consistency values.

6. Multidimensional Scaling (MDS) for GMA - theoretical backgrounds

At the beginning of this thesis there stood the idea of enhancing GMA with computer support - in a way that the focus would be shifted from single configurations to groups of configurations. If there are hundreds or thousands of configurations left in the reduced configuration space, one could of course go through one configuration after another and start grouping them into different categories. (With a holiday trip this could maybe be “short holidays nearby with little budget”, “long holidays far away that cost quite a lot” and “bicycle tours of various duration”.) However this would of course be very time-consuming and exhausting. It would be most convenient if one had a computer program doing the categorization automatically.

The formal configuration space can be imagined as an n -dimensional grid cuboid. And the reduction of the configuration space can be imagined as a chiseling process. Bit by bit the grid nodes of inconsistent configurations are removed - and in some cases there might emerge interesting shapes and clusters. Clustering algorithms try to detect such clusters - which would otherwise be left hidden in the n dimensions. Multidimensional Scaling (MDS), closely related to Principal Component Analysis, is a clustering method that projects the n -dimensional figure onto less dimensions, for example onto two dimensions (that can then be graphically depicted). Thereby MDS tries to preserve the distances between the data points as good as possible which also leads to an as-good-as-possible preservation of the clusters. Applied to GMA and the reduced configuration space, these clusters (that would be made visible by MDS) could represent different rather homogeneous groups of configurations. As the clusters are made visible on two dimensions, the structure of the reduced configuration space can then be explored visually (however always bearing in mind that it is only a projection). In Chapter 7 the GMAvisual prototype will be presented which implements MDS for GMA. But first the mathematical

backgrounds of the classical MDS algorithm shall be outlined. It is an ingenious procedure that makes clever use of linear algebra to transform given distances between data points to coordinates.

6.1 Classical Multidimensional Scaling¹

6.1.1 Calculating distances from coordinates

Given an $n \times m$ matrix containing the m coordinate values for n points (or might it of course be m parameter values of n sample configurations).

The following matrix $\mathbf{X}_{3 \times 2}$ would thus contain the coordinate values of three two-dimensional points ($p_1 = (x_{11}, x_{12})$, $p_2 = (x_{21}, x_{22})$, $p_3 = (x_{31}, x_{32})$):

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

In [BG97] Borg and Groenen depict a clever way to calculate the distance matrix of the points from their coordinate matrix. Later the reverse algorithm will be interesting: calculating the matrix of coordinates from a matrix of distances or similarities. For a better understanding the calculation of the distance matrix shall be included here anyway.

Starting from some coordinate matrix $\mathbf{A}_{n \times m}$, the goal is to obtain a symmetric square matrix $\mathbf{D}_{n \times n}$ where each entry d_{ij} represents the distance between point p_i and point p_j . The squared Euclidean distances are calculated by

$$d_{ij}^2 = \sum_{a=1}^m (x_{ia} - x_{ja})^2 = \sum_{a=1}^m (x_{ia}^2 + x_{ja}^2 - 2x_{ia}x_{ja}).$$

For the next step consider the exemplary case of $\mathbf{X}_{3 \times 2}$. With the formula for d_{ij}^2 the matrix of squared distances, written $\mathbf{D}^{(2)}$, is

$$\begin{aligned} \mathbf{D}^{(2)} &= \begin{bmatrix} 0 & d_{12}^2 & d_{13}^2 \\ d_{21}^2 & 0 & d_{23}^2 \\ d_{31}^2 & d_{32}^2 & 0 \end{bmatrix} \\ &= \sum_{a=1}^2 \begin{bmatrix} x_{1a}^2 & x_{1a}^2 & x_{1a}^2 \\ x_{2a}^2 & x_{2a}^2 & x_{2a}^2 \\ x_{3a}^2 & x_{3a}^2 & x_{3a}^2 \end{bmatrix} + \sum_{a=1}^2 \begin{bmatrix} x_{1a}^2 & x_{2a}^2 & x_{3a}^2 \\ x_{1a}^2 & x_{2a}^2 & x_{3a}^2 \\ x_{1a}^2 & x_{2a}^2 & x_{3a}^2 \end{bmatrix} - 2 \sum_{a=1}^2 \begin{bmatrix} x_{1a}x_{1a} & x_{1a}x_{2a} & x_{1a}x_{3a} \\ x_{2a}x_{1a} & x_{2a}x_{2a} & x_{2a}x_{3a} \\ x_{3a}x_{1a} & x_{3a}x_{2a} & x_{3a}x_{3a} \end{bmatrix}. \end{aligned}$$

With $\mathbf{c} = \begin{bmatrix} \sum_{a=1}^2 x_{1a}^2 \\ \sum_{a=1}^2 x_{2a}^2 \\ \sum_{a=1}^2 x_{3a}^2 \end{bmatrix}$, $\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $\mathbf{x}_a = \begin{bmatrix} x_{1a} \\ x_{2a} \\ x_{3a} \end{bmatrix}$ this dissolves to:

$$\mathbf{D}^{(2)} = \mathbf{c}\mathbf{1}^T + \mathbf{1}\mathbf{c}^T - 2 \sum_{a=1}^2 x_a x_a^T = \mathbf{c}\mathbf{1}^T + \mathbf{1}\mathbf{c}^T - 2\mathbf{X}\mathbf{X}^T \quad (6.1)$$

¹The whole section has been outlined from the MDS compendia of Borg and Groenen ([BG97]) and Cox and Cox ([CC01]). Also [Wic03] has been a great help in getting a quick understanding of the field.

The matrix $\mathbf{B} = \mathbf{X}\mathbf{X}^T$ is called *scalar product matrix* and will later on play an important role.

For now this decomposition makes it most easy to read the distance matrix from the scalar product matrix, which is illustrated by the following example:

Let \mathbf{X} take the following values: $\mathbf{X} = \begin{bmatrix} 3 & 7 \\ 4 & 2 \\ 0 & 1 \end{bmatrix}$

The scalar product matrix will therefore be: $\mathbf{B} = \mathbf{X}\mathbf{X}^T = \begin{bmatrix} 3 & 7 \\ 4 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 4 & 0 \\ 7 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 58 & 26 & 7 \\ 26 & 20 & 2 \\ 7 & 2 & 1 \end{bmatrix}$

The values of \mathbf{c} are just the diagonal entries of the scalar product matrix: $\mathbf{c} = \begin{bmatrix} 58 \\ 20 \\ 1 \end{bmatrix}$

And so:

$$\mathbf{D}^{(2)} = \begin{bmatrix} 58 & 58 & 58 \\ 20 & 20 & 20 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 58 & 20 & 1 \\ 58 & 20 & 1 \\ 58 & 20 & 1 \end{bmatrix} - 2 \begin{bmatrix} 58 & 26 & 7 \\ 26 & 20 & 2 \\ 7 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 26 & 45 \\ 26 & 0 & 17 \\ 45 & 17 & 0 \end{bmatrix}.$$

Then taking the square root of all elements gives the distance matrix $\mathbf{D} \approx \begin{bmatrix} 0 & 5.1 & 6.7 \\ 5.1 & 0 & 4.1 \\ 6.7 & 4.1 & 0 \end{bmatrix}$.

6.1.2 Calculating coordinates from distances

Now we want to address the reverse problem: Given a matrix of distances \mathbf{D} , how can we calculate a matrix of point coordinates that match these distances?

The method that Borg and Groenen call *Classical Scaling* (also known as *Torgerson Scaling*) consists of two major steps: First the scalar product matrix \mathbf{B} is calculated from the matrix of squared distances $\mathbf{D}^{(2)}$ (via double centering) and then the coordinate matrix \mathbf{X} is calculated from $\mathbf{B} = \mathbf{X}\mathbf{X}^T$ (via eigendecomposition).

Step 1 - Double Centering

Note first that a matrix can be *centered* by multiplying it with the matrix $\mathbf{J}_n = \mathbf{I}_n - n^{-1}\mathbf{1}_n\mathbf{1}_n^T$.² When \mathbf{J} is multiplied with an n -dimensional vector \mathbf{v} the mean of the vector's entries is subtracted from each of them. That way the entries of the resulting vector $\mathbf{v}_c = \mathbf{J}\mathbf{v}$ add up to zero.

Example: $n = 2$ and $\mathbf{v} = \begin{bmatrix} 4 \\ 9 \end{bmatrix}$

$$\mathbf{v}_c = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 4 \\ 9 \end{bmatrix} = \begin{bmatrix} -2.5 \\ 2.5 \end{bmatrix}$$

² \mathbf{I}_n is the $n \times n$ identity matrix.

If \mathbf{J}_n is multiplied to some $n \times n$ matrix from the left, the *columns* of the matrix are centered. If it is multiplied from the right, it is the *rows* that are being centered. If \mathbf{J}_n is multiplied to a matrix from the left and from the right, this procedure is called *double centering*. What happens is, that the respective column mean and the respective row mean are subtracted from the matrix entries, and the mean of all matrix entries is readded. All columns and all rows then sum up to zero.

Example:

$$\begin{aligned} \mathbf{J}_3 \begin{bmatrix} 3 & 2 & 7 \\ 0 & 9 & 1 \\ 1 & 0 & 5 \end{bmatrix} \mathbf{J}_3 &= \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ -1/3 & 2/3 & -1/3 \\ -1/3 & -1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 3 & 2 & 7 \\ 0 & 9 & 1 \\ 1 & 0 & 5 \end{bmatrix} \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ -1/3 & 2/3 & -1/3 \\ -1/3 & -1/3 & 2/3 \end{bmatrix} \\ &= 1/3 \begin{bmatrix} 5 & -5 & 8 \\ -4 & 16 & -10 \\ -1 & -11 & 2 \end{bmatrix} \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ -1/3 & 2/3 & -1/3 \\ -1/3 & -1/3 & 2/3 \end{bmatrix} = 1/9 \begin{bmatrix} 7 & -23 & 16 \\ -14 & 46 & -32 \\ 7 & -23 & 16 \end{bmatrix} \end{aligned}$$

When we only have a distance matrix \mathbf{D} and are looking for a corresponding coordinate matrix \mathbf{X} , we can assume that the underlying coordinates are equally distributed in their respective dimensions, i.e. we can assume that \mathbf{X} is column-centered.³ In this case $\mathbf{B} = \mathbf{X}\mathbf{X}^T$ already is double-centered.

Example:

$$\begin{bmatrix} 3 & -6 \\ 0 & -3 \\ -3 & 9 \end{bmatrix} \begin{bmatrix} 3 & 0 & -3 \\ -6 & -3 & 9 \end{bmatrix} = \begin{bmatrix} 45 & 18 & -63 \\ 18 & 9 & -27 \\ -63 & -27 & 90 \end{bmatrix}$$

If you center the vector $\mathbf{1}_n$ consisting of only ones, you get a vector of zeros, so $\mathbf{J}\mathbf{1} = \mathbf{0}$ and $\mathbf{1}^T\mathbf{J} = \mathbf{0}^T$.

With this knowledge we can go back to Equation 6.1 that has been deduced in the previous section:

$$\mathbf{D}^{(2)} = \mathbf{c}\mathbf{1}^T + \mathbf{1}\mathbf{c}^T - 2 \sum_{a=1}^2 x_a x_a^T = \mathbf{c}\mathbf{1}^T + \mathbf{1}\mathbf{c}^T - 2\mathbf{X}\mathbf{X}^T = \mathbf{c}\mathbf{1}^T + \mathbf{1}\mathbf{c}^T - 2\mathbf{B}$$

Multiplying both sides with the centering matrix \mathbf{J} (both from the left and from the right) yields:

$$\begin{aligned} \mathbf{J}\mathbf{D}^{(2)}\mathbf{J} &= \mathbf{J}(\mathbf{c}\mathbf{1}^T + \mathbf{1}\mathbf{c}^T - 2\mathbf{X}\mathbf{X}^T)\mathbf{J} = \mathbf{J}\mathbf{c}\mathbf{1}^T\mathbf{J} + \mathbf{J}\mathbf{1}\mathbf{c}^T\mathbf{J} - \mathbf{J}(2\mathbf{X}\mathbf{X}^T)\mathbf{J} \\ &= \mathbf{J}\mathbf{c}\mathbf{0}^T + \mathbf{0}\mathbf{c}^T\mathbf{J} - 2\mathbf{J}\mathbf{B}\mathbf{J} = -2\mathbf{B} \end{aligned} \tag{6.2}$$

And so the scalar product matrix can easily be calculated: $\mathbf{B} = \mathbf{X}\mathbf{X}^T = -\frac{1}{2}\mathbf{J}\mathbf{D}^{(2)}\mathbf{J}$.

³We can do this, because the resulting coordinates can be translated, rotated, etc. As we only know the distances between the points, the coordinates can only be calculated with respect to each other and not absolutely. Or as Cox and Cox put it: "To overcome the indeterminacy of the solution due to arbitrary translation, the centroid of the configuration of points is placed at the origin." [CC01]

Step 2 - Eigendecomposition

Now it all comes down to getting the coordinate matrix \mathbf{X} out of the scalar product matrix $\mathbf{B} = \mathbf{X}\mathbf{X}^T$. This is done via eigendecomposition of the matrix \mathbf{B} .

The matrix \mathbf{B} is symmetric. Symmetric matrices are also diagonalizable. Therefore exists a matrix \mathbf{P} and a diagonal matrix Λ such that $\mathbf{P}^{-1}\mathbf{B}\mathbf{P} = \Lambda$. Thereby the columns of \mathbf{P} are just the eigenvectors of \mathbf{B} , the corresponding eigenvalues are the diagonal entries of Λ .

As scalar product matrices additionally have nonnegative eigenvalues, $\mathbf{B} = \mathbf{P}\Lambda\mathbf{P}^T$ can be written as $\mathbf{B} = (\mathbf{P}\Lambda^{1/2})(\mathbf{P}\Lambda^{1/2})^T = \mathbf{U}\mathbf{U}^T$ with $\Lambda^{1/2}$ being Λ with diagonal elements $\lambda_i^{1/2}$.

The input to the algorithm was the matrix of distances \mathbf{D} . The scalar product matrix is quickly calculated via the equation $\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{D}^{(2)}\mathbf{J}$. Then the eigendecomposition of the matrix \mathbf{B} is effected. This is a method of linear algebra which yields the eigenvalues of a matrix as well as the corresponding eigenvectors. Writing the eigenvectors as the columns of a matrix gives \mathbf{P} and writing the eigenvalues as diagonal entries of a diagonal matrix gives Λ . So the coordinate matrix \mathbf{U} is easily found via effecting the eigendecomposition. Only taking the eigenvectors of the first x highest eigenvalues and writing them into a matrix \mathbf{P}_x , writing the first x eigenvalues themselves into a diagonal matrix Λ_x and multiplying these two matrices yields x -dimensional coordinates for the data points in a matrix $\mathbf{X} = \mathbf{P}_x\Lambda_x^{1/2}$. This is the most interesting point: classical MDS does not only calculate coordinates from distances, it also allows to reduce the dimensions of the resulting coordinates as far as desired. This is due to the basis transformation that is implicitly effected. The eigenvector corresponding to the biggest eigenvalue holds the most information about the structure of the data set, the second one the second most, and so on. Reducing the coordinate dimensions to two or three will therefore definitely still mean a loss of information, but as little as anyhow possible. This characteristic of the eigendecomposition is most useful: two-dimensional coordinate embeddings can be calculated from any data set - given the distances between the data points.

In the next section we will see how distances (or better: dissimilarities) between GMA configurations can be calculated. With this knowledge, a distance matrix of all configurations of the reduced configuration space can be calculated (a matrix containing all the distances between any configuration and any other). Giving this distance matrix into the MDS algorithm that has just been explained, yields two-dimensional coordinates for all of the configurations of the reduced space. And with these coordinates the configurations can be plotted to a two-dimensional plane (as will be shown in Chapter 7). The arising picture will be a projection of the n -dimensional reduced configuration space onto two dimensions - and will therefore bear some (not all) markings of the n -dimensional structures.

6.2 Similarity measures for categorical data

As outlined in the previous section, the algorithm of classical MDS calculates coordinate embeddings from distance matrices. When the distances between the data points are not exactly measurable (like maybe Euclidean distances), it's rather their *similarities* that

are determined. Similarities can then be translated to dissimilarities which then can fulfill the function of distances. So, when MDS shall be applied to the reduced configuration space of GMA, first a similarity matrix of the contained configurations is needed, so a matrix which entries each indicate the similarity of a specific pair of configurations. Fortunately these similarities between configurations don't need to be evaluated by hand - there are *similarity measures* for this kind of data, called *categorical*:

In GMA the value sets of the problem parameters are finite, each containing a fix number of values. So the value ranges are not continuous, but partitioned into finite categories. Some of them could of course easily be made continuous⁴, but this would imply some major changes in the GMA modelling process (especially in the consistency assessment). So for now let all value ranges remain categorical. Defining distance measures for continuous data is very common, but what is the (dis-)similarity, what is the "distance" between "tent" and "hotel"? They are obviously different, but how can this difference be measured?

The most basic approach is the so-called *overlap* similarity measure (from which respectively the overlap dissimilarity measure can be derived). As Boriah et al. put it in [BCK08], "the simplest way to find similarity between two categorical attributes is to assign a similarity of 1 if the values are identical and a similarity of 0 if the values are not identical." Let s_o be this function. So, with two categorical values v_a and v_b (of the same parameter), the overlap similarity is measured by:

If $v_a = v_b$: $s_o(v_a, v_b) = 1$.

Else: $s_o(v_a, v_b) = 0$.

With $v_k(c)$ meaning the k -th value of c , the similarity $S_o(c_i, c_j)$ of two *vectors* c_i and c_j of categorical values (like the GMA configurations) would then be calculated by:

$$S_o(c_i, c_j) = \frac{1}{n} \sum_{k=1}^n s_o(v_k(c_i), v_k(c_j)) \quad (6.3)$$

Let's look at some example configurations and their overlap similarities:

$c_1 = ([0,1), \textit{Single destination}, [0,50), \textit{none (walking)}, \textit{none}, [0,100))$

$c_2 = ([14,30], \textit{Single destination}, >1000, \textit{airplane}, \textit{all-inclusive}, >1000)$

$c_3 = ([0,1), \textit{Single destination}, [0,50), \textit{bicycle}, \textit{tent}, [0,100))$

$c_4 = ([0,1), \textit{Only roadtrip}, [0,50), \textit{bicycle}, \textit{tent}, [0,100))$

$$S_o(c_1, c_1) = 1 \quad S_o(c_1, c_2) = \frac{1}{6} \quad S_o(c_1, c_3) = \frac{4}{6} \quad S_o(c_1, c_4) = \frac{3}{6}$$

It has been mentioned that the overlap measure is the simplest form of similarity measure for categorical data. There have been various approaches in various domains to improve it. In [BCK08] Boriah et al. introduce the problem, bring together fourteen categorical measurements from different fields and study their performance on outlier detection tasks. The formalization of the overlap measure and the information following in this section are all adapted from this paper. -

When it comes to trying to improve the similarity measurement of categorical data,

⁴For example could the value range of the parameter "Costs" in the Holiday Trip problem just be $[0; \infty[$.

the central question is: What are the characteristics of categorical data? Boriah et al. detect the size of the data set N (so the size of currently examined reduced configuration space), the number n of parameters of the problem, the sizes m_i of the value sets V_i and the frequencies $f_i(v_j)$ of the values v_j within the value sets V_i . With the frequencies, sample probabilities $p_i(v_j) = \frac{f_i(v_j)}{N}$ can be calculated. The various approaches all try to usefully include these characteristics to improve the similarity measurement.

To give a certain idea of what that can look like, one example measure shall be presented: the Inverse Occurrence Frequency measure (IOF). The formula of the IOF measure for configurations S_{IOF} is the same as for the overlap measure, interesting is the IOF measure for values s_{IOF} :

$$S_{IOF}(c_i, c_j) = \frac{1}{n} \sum_{k=1}^n s_{IOF}(v_k(c_i), v_k(c_j))$$

$$s_{IOF}(v_k(c_i), v_k(c_j)) = \begin{cases} 1, & \text{if } v_k(c_i) = v_k(c_j), \\ \frac{1}{1 + \log f_k(v_k(c_i)) \times \log f_k(v_k(c_j))}, & \text{otherwise.} \end{cases}$$

If the values that are comprised by the compared configurations and that they don't have in common have high average frequencies, the similarity between these configurations will be lower than if they had low average frequencies. As Boriah et al. put it: "The inverse occurrence frequency measure assigns lower similarity to mismatches on more frequent values." Even more generally speaking: Configurations both having less frequent values are more similar. The lower the frequency of two values (of the same parameter), the more they are treated as being the same value. So, figuratively speaking, the distances are stretched on the dimensions of frequent values.

The *Occurrence Frequency* measure (OF) shall be quickly mentioned as well. The OF measure gives the opposite weighting of the IOF measure: Configurations with *more* frequent values are more similar. Analogously S_{OF} and s_{OF} are defined as follows:

$$S_{OF}(c_i, c_j) = \frac{1}{n} \sum_{k=1}^n s_{OF}(v_k(c_i), v_k(c_j))$$

$$s_{OF}(v_k(c_i), v_k(c_j)) = \begin{cases} 1, & \text{if } v_k(c_i) = v_k(c_j), \\ \frac{1}{1 + \log \frac{N}{f_k(v_k(c_i))} \times \log \frac{N}{f_k(v_k(c_j))}}, & \text{otherwise.} \end{cases}$$

As already mentioned, Boriah et al. investigate fourteen different similarity measures for categorical data. All of them take into account different combinations of the mentioned characteristics of the categorical data set. To evaluate the different measurements, they apply outlier detection tasks on several data sets - measuring the similarities of the data sets with the different measures. Their conclusion: In many situations the overlap measure does not perform too well and using the additional information does often make sense. There are some measures that consistently perform better than others (on a variety of data). But: "No single measure is always superior or inferior. This has to be expected since each data set has different characteristics." Especially, there are some measures with complementary performance (OF and IOF for example, as could be expected). On

tasks on which one of them performs well, the other one performs badly and vice versa. When it comes to choosing a similarity measure for applying MDS on the reduced configuration space of GMA, this is probably the most important information to keep in mind: there will probably not be one similarity measure that performs best with all GMA problems and all sizes of reduced configuration spaces. So for software that visualizes GMA using similarity measures, it will be a good idea to let the user flexibly choose from a variety of different similarity measures. Some might perform better with the one problem, others with another one.

However this should of course be subject to further research. Just as Boriah et al. found some measures that consistently performed better than others with outlier detection tasks, there could be some measures that significantly yield better results when combined with MDS for visualizations of GMA.

Converting similarities to dissimilarities

As described above, the input to the MDS algorithm is a distance or dissimilarity matrix. However, so far, this section only examined *similarity* measures for categorical data. Boriah et al. suggest to convert similarities to dissimilarities via the following formula:

$$sim = \frac{1}{1 + dist} \quad , \quad dist = \frac{1}{sim} - 1.$$

This approach has been adapted for the rest of this thesis, so also in the GMAvisual prototype.

7. GMAvisual - an MDS prototype for GMA

After these preliminaries, the prototype that has been implemented in the context of this thesis shall be presented and explained. GMAvisual brings together General Morphological Analysis and Multidimensional Scaling (respectively two-dimensional visual clustering in general). The focus of the software strongly lies on the cluster visualizations - that can be interactively explored in many different ways. The purpose of the two-dimensional cluster visualizations is to be an additional step of the GMA process. After having reduced and sorted the configuration space, the question is: what can be done to get a better and more direct insight into the structure of the remaining configuration space other than just to go through the list of configurations. And the high-end goal: might it be possible to - via computer support and visualizations - reduce the configuration space not only to a smaller set of configurations, but to few configuration categories? - In the Holiday Trip example, 16,200 configurations have been reduced to 4641 configurations via pairwise consistency analysis and then further to 1786 configurations via selective TCA. If we would go through the list of these 1786 configurations, we might be able to group them into different categories of holidays. This would make clear which the main options are and this would help a lot in the decision process of finally choosing one specific kind of holiday.

GMAvisual is the essence of this thesis: it works with binary¹ as well as with scaled consistency assessment, provides selective TCA and implements the weighting functions \bar{w}_1 , \bar{w}_2 and \bar{w}_3 from Section 3.4. The consistency values can be calculated with the triple correction factor or the configurations containing **very bad** or **worst** tuples or **inconsistent** triples can just be deleted.

Figure 2.1 already showed the simple realization of the morphological field (just being used for the data input), in Section 3.3 the implementation of the selective TCA was de-

¹Note: For a visualization to make sense, the *whole* binarily reduced configuration space needs to go into the scaling algorithm and not only the “first” x configurations.

picted. The pairwise CCA has been realized quite simplistically, without the preferable graphic cross consistency matrix (see Figure 7.1). Figure 7.2 shows the root of GMAvisual: a list of all GMA projects that have been completely entered so far (incl. at least pairwise CCA). Changes in the morphological fields and in the consistency assessment are not supported by the graphical interface - the focus is all on the visualization. When double-clicking on a project in the list, the *Visual Frame* opens. Various Visual Frames can be opened at the same time, also can the same project be opened simultaneously in several windows.

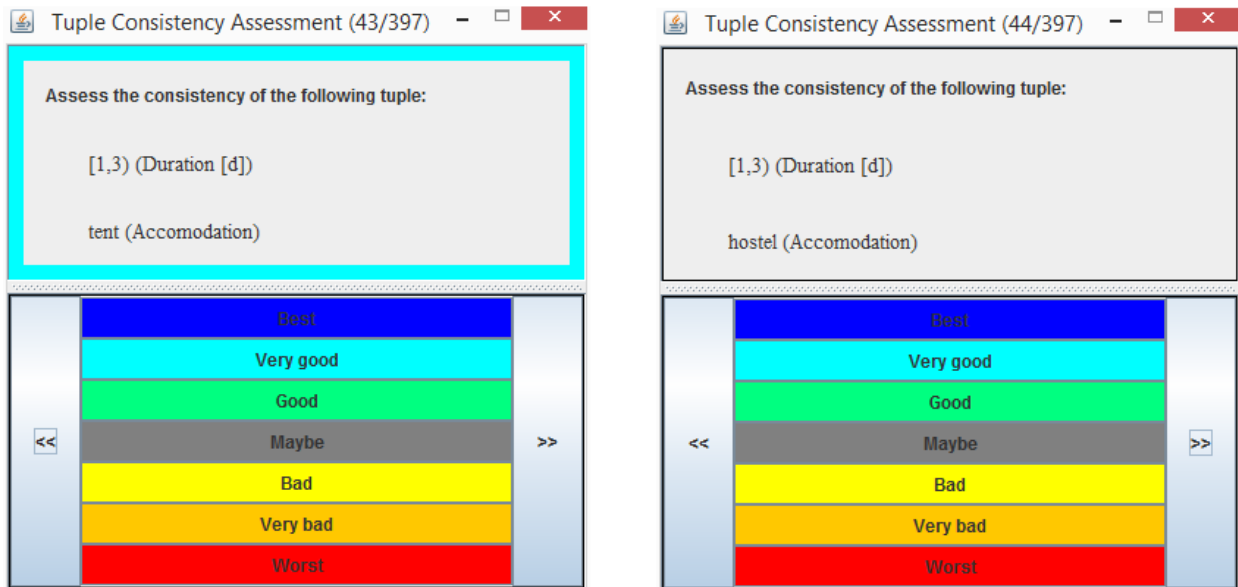


Figure 7.1: Simplistic cross consistency assessment in GMAvisual. The user just goes through all of the tuples one after another, assessing the consistency of the currently shown tuple via click on the respective Likert item.

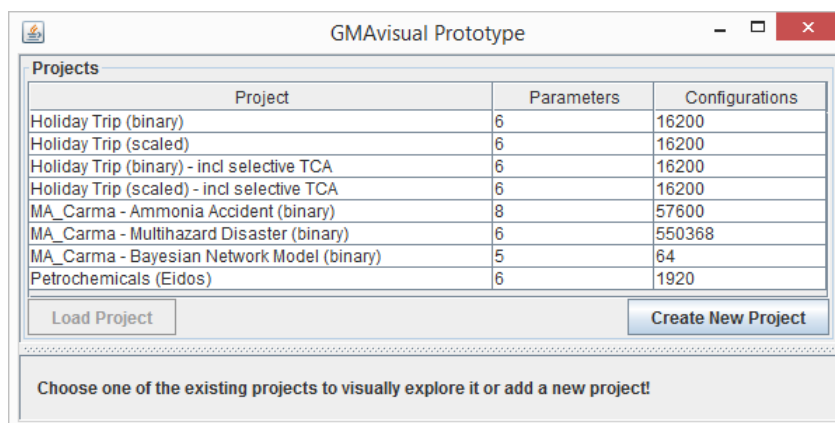


Figure 7.2: Root of GMAvisual with the list of available projects. The visual frame for exploring a project can be opened via the “Load Project”-button or via double-click on the list element. The button “Create New Project” leads the user to the input of a new morphological field.

7.1 Functions of GMAvisual by control device areas

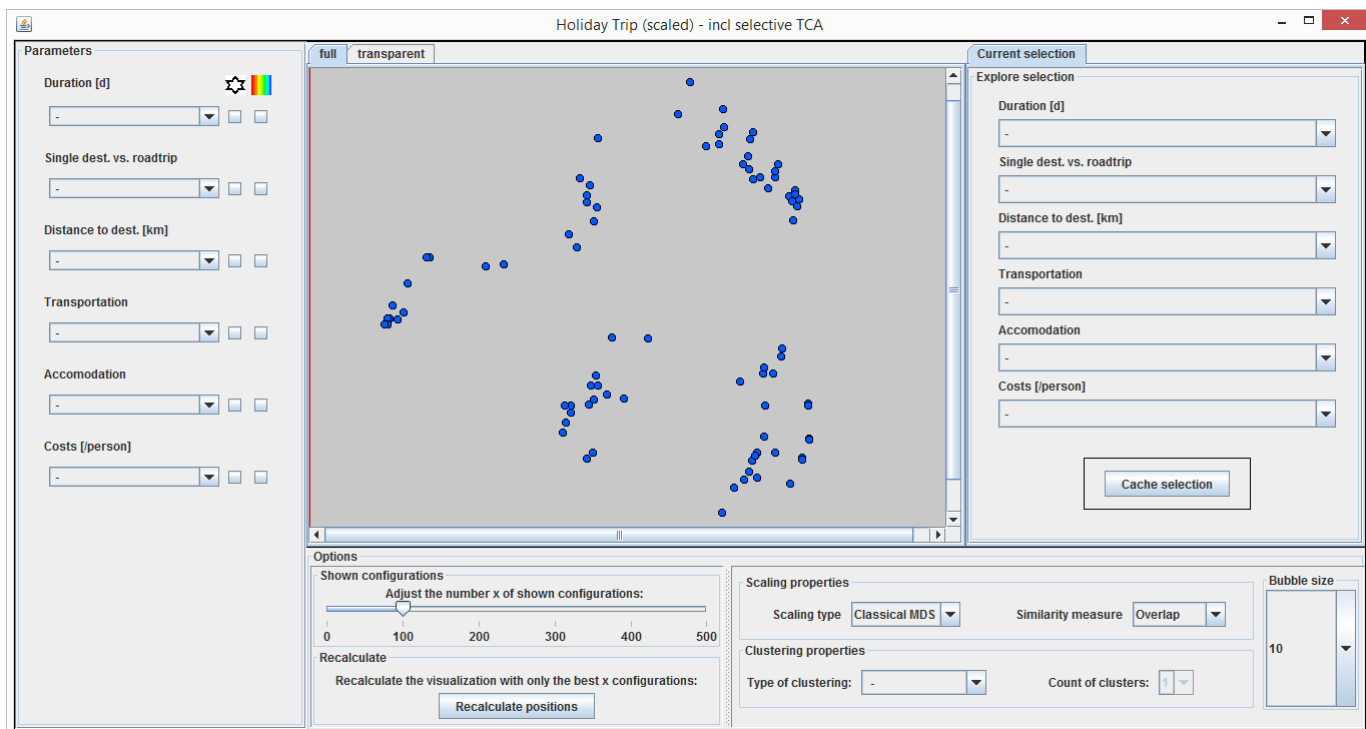


Figure 7.3: Visual frame in GMAvisual for the Holiday Trip problem. Frame has just been opened, the shown visualization has been calculated for the first 100 configurations (via Classical MDS and the overlap measure).

Figure 7.3 shows the Visual Frame for the Holiday Trip problem with scaled consistency data, right after opening it. In the center of the frame one can see the heart of the program, the visualization itself. This central area of the frame will be referred to as *visualization area*. By default the configurations are depicted as blue bubbles on a grey background.² In Figure 7.3 the best-ranking 100 configurations are visualized. The scaling has been done with the classical MDS algorithm, the distances have been measured via overlap. These properties can be seen in the *option area* at the bottom. The column-like area on the left will be referred to as *parameter area*, the one on the right, next to the visualization, as *explore area*.

Remark: For the visualization shown in the following sections, the consistency calculation has been done with the weighting function \bar{w}_2 , so with the weighted approach, the configurations with inconsistent tuples or triples have anyway been deleted from the configuration space. This approach just is the conceptually best one. We will see however, that other approaches (equidistant consistency calculation, not deleting any configurations) might sometimes yield “nicer” clustering results.

The various functions of GMAvisual that help the user to explore the reduced configuration space and to understand its structure are best explained via the different areas of

²The “bubbles” can have other than round shapes and are therefore also referred to as *panels*.

the Visual Frame. Each of them has its own genuine functions. These of course get more powerful when combined, but let's first have a look at them one after another.

7.1.1 Option area

The *option area* is the area at the bottom respectively on the bottom right of the Visual Frame. It contains various steering elements.

In the left part of the option area, the number of shown configurations can be adjusted. The maximum of shown configurations has arbitrarily set to 500. A higher number would well be possible, but then the MDS calculation needs uncomfortably much time. After all the consistency assessment is optimized now, so the first 500 configurations should be enough anyway.

With the slider the user can change the number of shown configurations. If he just moves the slider, there will be no new calculation of the clustering, but the bubbles of the configurations not belonging to the first x configurations will just be suppressed. That way the depicted clusters can be searched for the most consistent configurations. The upper two subfigures of Figure 7.4 show what happens if the slider is moved to around 50 (left side) and around 25 shown configurations (right side). The configurations at position 50 to 100 respectively at position 25 to 50 are just hidden, the others rest in place. If the slider is moved back, the hidden configuration bubbles just reappear in their position.

If the visualization has originally been calculated with the best x configurations, moving the slider to a position y greater than x will not have any effect. If the user wants more (or less) configurations to go into the MDS algorithm, he can press the "Recalculate positions"-button (highlighted in light blue whenever the slider has been changed and a new clustering could therefore be calculated). Then GMAvisual will recalculate the clustering with only the first y configurations and will depict representing bubbles at the new positions. The two subfigures at the bottom of Figure 7.4 show the MDS recalculations with only the best 50 (at the left) and only the best 25 configurations (at the right). On the left side one can see that the main structure prevails when recalculating, but on the right side this is less the case. Running MDS with a different set of first configurations, can completely change the look of the visualizations. After all MDS does not have a concept of semantic clusters, but just calculates two-dimensional coordinates from (overlap) distances - to fit as good as possible. The calculated embedding strongly depends on the data set that goes into the algorithm.

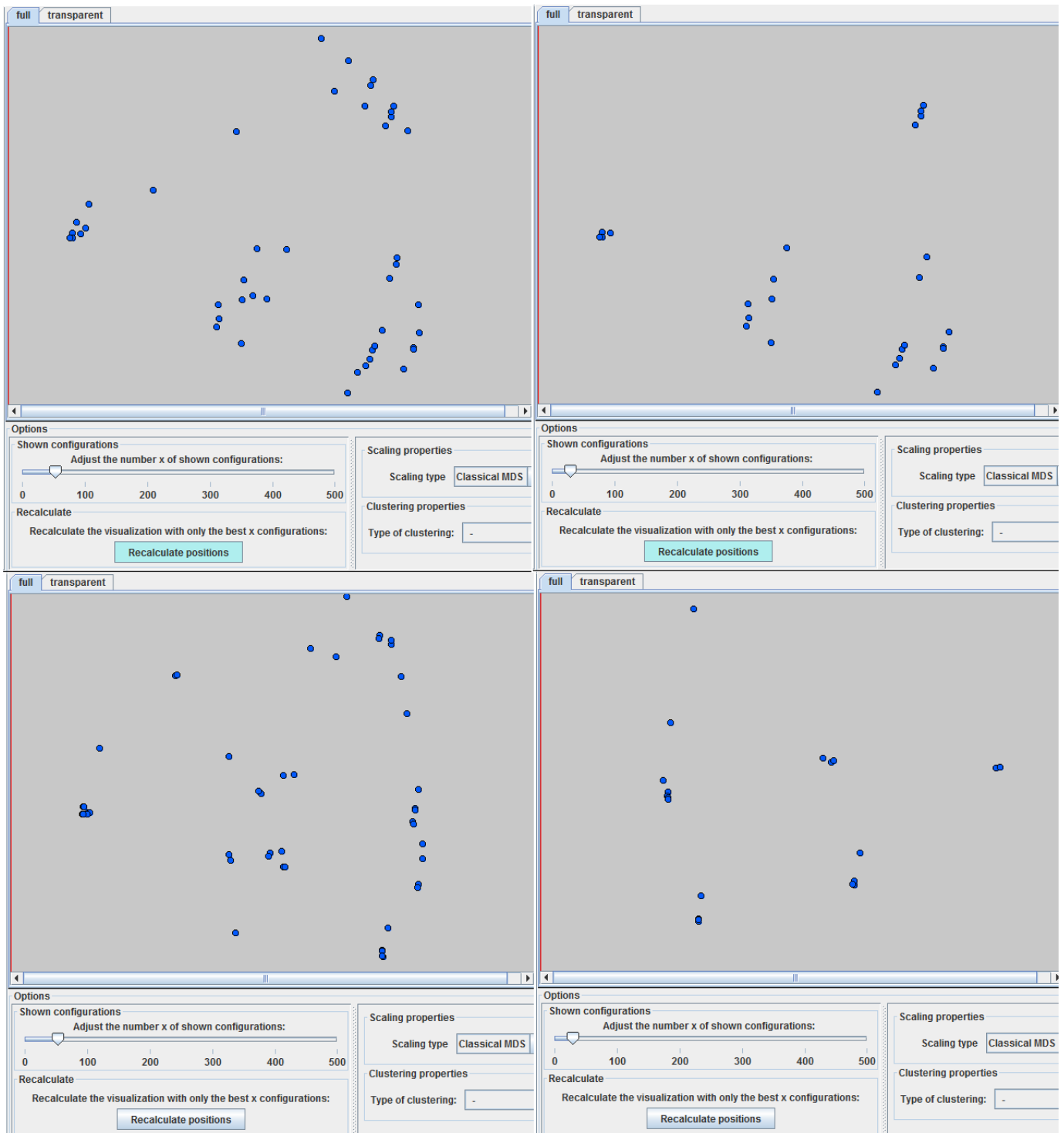


Figure 7.4: Adjusting the number of shown configurations. *Left*: Around 50 configurations shown. *Right*: Around 25 configurations shown.

In the subfigures at the top, the slider has been moved to 50 and 25 without recalculating the clustering. So just some of the bubbles have been hidden, without changing the position of the others. In the subfigures at the bottom, the MDS algorithm was launched again with only the 50 (25) best configurations. Their positions are recalculated. The others are not only hidden, but completely removed for the moment.

In the option area the user can furthermore choose a type of scaling and a similarity measure. In both cases, when choosing one of the items from the drop-down list, the visualization is recalculated at once - with the new properties. Besides there is a field with clustering properties. A type of standard clustering method can be chosen, as well as the wished number of clusters. Both the clustering properties as well as the scaling properties will be explained and illustrated later.

It may happen that the default value of the size of the configuration bubbles is inconvenient. Depending on the scaling algorithm, the similarity measure or just the number of depicted configurations, the bubbles might be uncomfortably small - or also too big, overlapping each other too much. Therefore, also the bubble size can be adjusted in the option area (at the field in the lower right corner). Figure 7.5 shows the visualization from Figure 7.3 (bubble size 10) alongside the same visualization with bubble size 15.

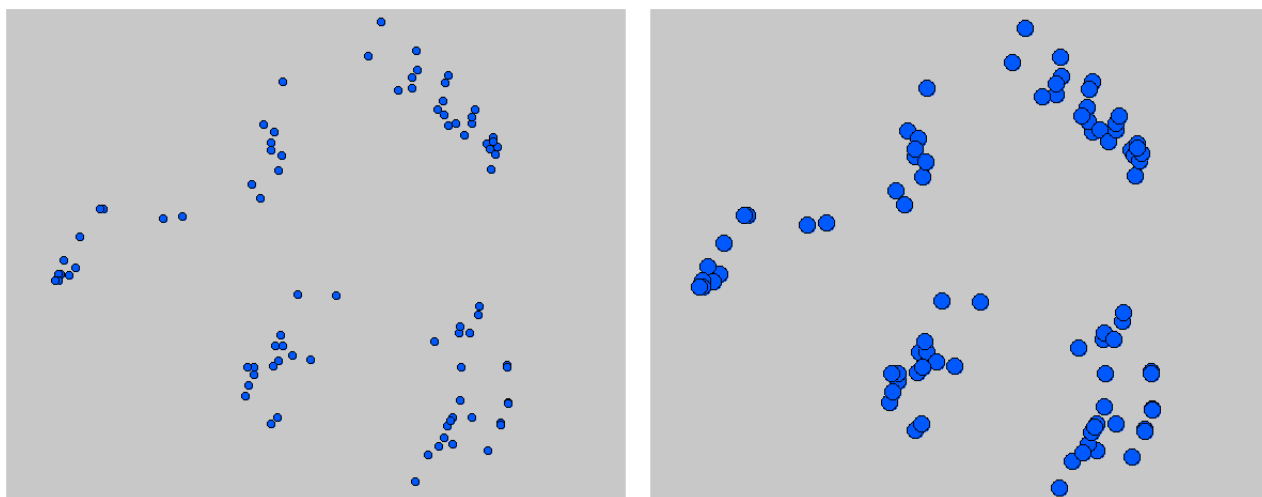


Figure 7.5: Adjusting the size of the configuration bubbles. *Left*: bubble size 10. *Right*: bubble size 15.

7.1.2 Parameter area

The *parameter area* at the left of the Visual Frame shows all parameters of the loaded project, combined with a drop-down list of the respective parameter values.

If one of the values is selected, it is “fixed” and the bubbles of all configurations that don’t contain this specific value at the respective slot are hidden from the visualization. This is one way for the user to flexibly interact with the visualization. He can thereby easily explore where to find which parameter values. The current GMAvisual prototype only allows to fix one value per parameter, but this could of course easily be enhanced. Figure 7.6 illustrates the fixation of parameter values and its use. In the upper left the value “Single destination” is selected, so all configurations that don’t have this value in the “Single dest. vs. roadtrip”-slot are hidden. In the upper right of Figure 7.6, contrarily, the value “Only roadtrip” is fixed. Obviously, referring to the whole clustering of the previous figures, the single-destination holidays are depicted at the edges, while the only-roadtrip holidays are located in the center. The bottom left of Figure 7.6 shows

all configurations containing the parameter value “[0,100)” in the “Costs”-slot - and the bottom right shows the combination of two fixed values: As “Single destination” *and* “[0,100)” have been fixed, the visualization obviously shows the intersection of the two visualizations at the left.

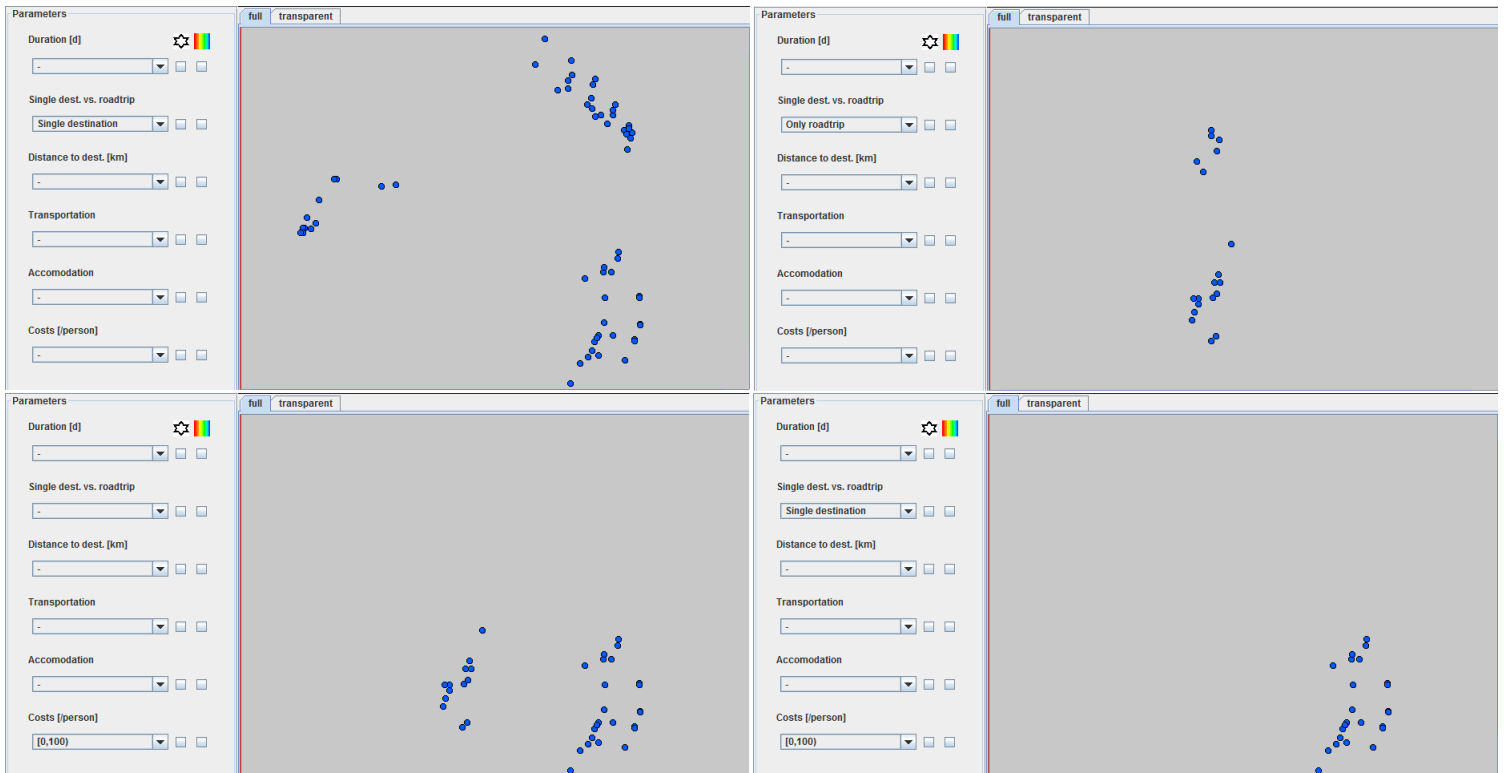


Figure 7.6: Fixing parameter values. *Upper left*: All configurations not containing the value “Single destination” are filtered out. *Upper right*: All configurations not containing the value “Only roadtrip” are filtered out. *Bottom left*: All configurations not containing the value “[0,100)” are filtered out. *Bottom right*: All configurations not containing the value “Single destination” *and* the value “[0,100)” are filtered out.

But there is more to the parameter area: with the small box next to the parameter’s drop-down list, the respective parameter can be chosen to get his values marked by shape and/or color. If the color box is marked, each value of the parameter gets assigned a color. The configuration bubbles in the visualization are then colored respective to the value they have in this parameter’s slot. A legend appearing at the bottom of the parameter area explains which color corresponds to which value. Analogously, shapes are assigned to the parameter’s values, in case its shape box is clicked. Figure 7.7 shows examples for both cases. In the left subfigure, the parameter “Costs” has been marked to be colored. On the bottom left the color legend appeared that explains that the value “[0,100)” is depicted in blue, the value “[100,250)” in green, and so on. The coloring can be a great help in exploring the visualization (respectively the reduced configuration space). In Figure 7.7, most of the very cheap holidays are located at the bottom right, the holidays

that cost between 100€ and 500€ are located in the upper right, and the expensive ones at the left. If his budget is low, the user can concentrate on the configurations on the right for his further exploring.

The right subfigure of Figure 7.7 - by shaping the configurations of a parameter - shows some information that we already know: the single-destination holidays (with the circle shape) are located in the periphery, the only-roadtrip holidays (with the star shape) and the mixed holidays (with the square shape) are located in the center of the visualization.

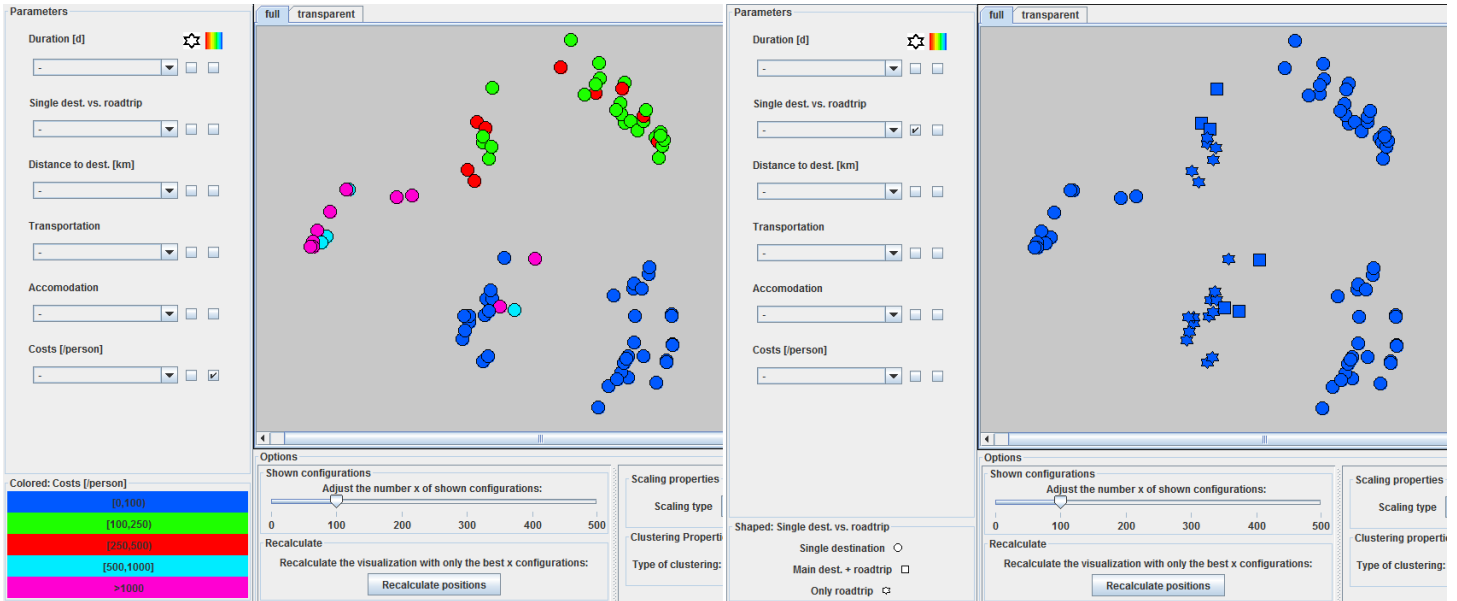


Figure 7.7: Coloring and shaping of parameter values. *Left*: The parameter “Costs” has been selected to be *colored*. The legend at the bottom left explains which color corresponds to which value of the parameter. *Right*: The parameter “Single dest. vs. roadtrip” has been selected to be *shaped*. The legend at the bottom left explains which shape corresponds to which value of the parameter.

In general, the coloring is more easily accessible than the shaping. The latter is better to be used for parameters that have few values. If there are circles on the one side and squares on the other, and then maybe some mixed area - this information can quickly be grasped (and can well be combined with the coloring of another parameter). But if there are more than five shapes distributed all over the visualization, the shaping will be of little use.

Figure 7.8 shows what it looks like, when the two possibilities are combined. There are no blue squares, so obviously there are no very cheap holidays among the best 100 that apply to the concept of having a main destination and then doing a roadtrip there. Also, there are no pink or light-blue stars, so there is no only-roadtrip holiday among the best 100 that costs more than 500€.

If the coloring of one parameter and the shaping of another are combined with the successive fixing of the values of a third parameter, there are already three dimensions of the problem that can be examined at the same time. In Figure 7.9 the parameter “Transportation” has been selected as the third dimension. The left side shows the situation with

the value “car” fixed, the right side with the value “airplane”. - With the car, there are obviously various possibilities, the majority of the consistent holidays with the car cost less than 500€ though. When traveling by plane, one should be ready to definitely spend more than 500€ - and obviously there is no only-roadtrip holiday by airplane among the consistent configurations.

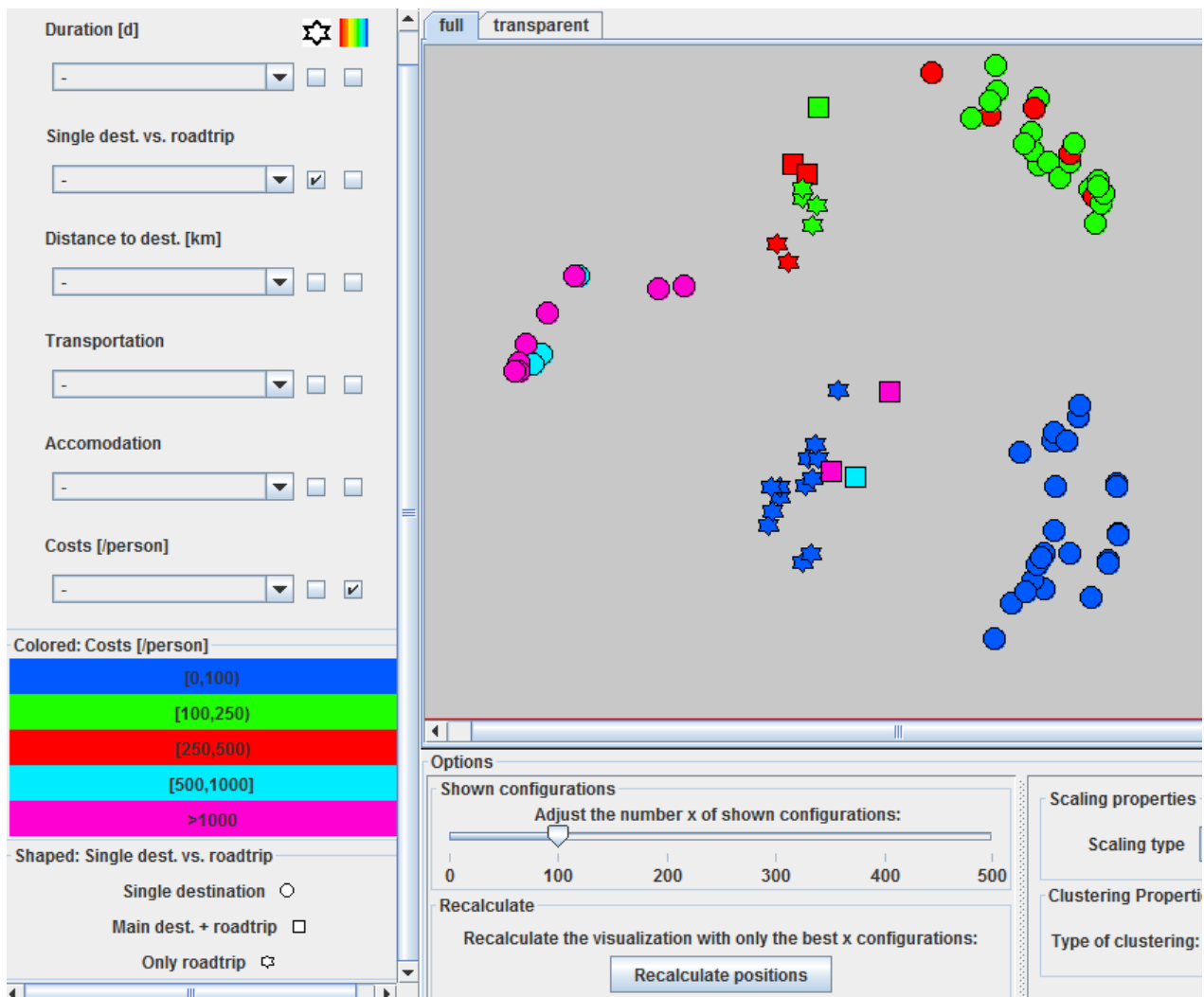


Figure 7.8: Combination of coloring and shaping. The parameter “Costs” has been selected to be colored, while at the same time the parameter “Single dest. vs. roadtrip” has been selected to be shaped. The legend explains the coloring as well as the shaping.

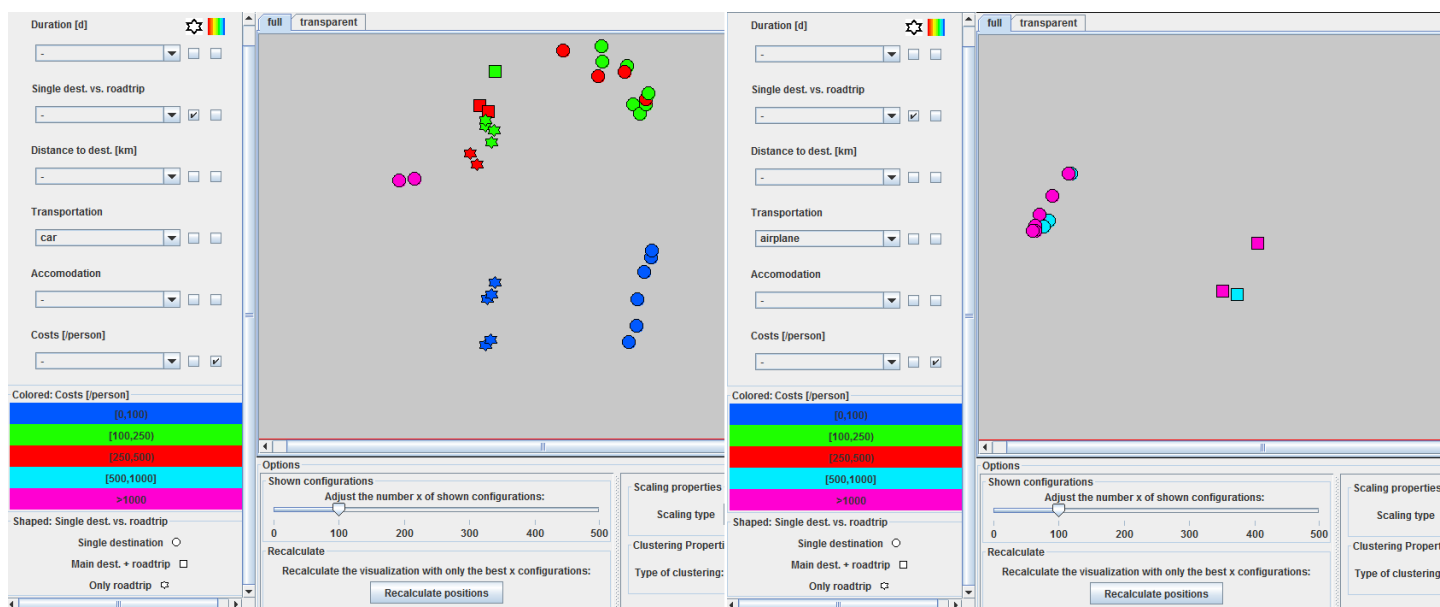


Figure 7.9: Exploring three dimensions. The parameter “Costs” has been selected to be colored, while at the same time the parameter “Single dest. vs. roadtrip” has been selected to be shaped. The function of being able to fix parameter values is used to explore a third dimension, namely the parameter “Transportation”. *Left*: The value “car” has been fixed. *Right*: The value “airplane” has been fixed. - The configurations not containing the respective value are again hidden from the visualization.

7.1.3 Visualization area

In GMAvisual there are three kinds of functions that are offered by the *visualization area* itself and affect only the visualization area. For once, when the mouse enters a configuration’s bubble (or more general: the configuration’s panel), the values of this configuration are depicted in a small tooltip (see Figure 7.10).

Furthermore there are two alternative views: one shows the configurations’ panels with full colors, the other with transparent colors. The latter can be practical in case many panels overlap. The user can jump between these two views via the tabs above the area. When the mouse enters a configuration panel - alongside the showing of the tooltip - the panel gets a white border (in full color mode) or gets painted with full color (in transparent color mode) (see again Figure 7.10).

The third genuine property of the visualization area is the zooming. The area allows to flexibly zoom in and out on the visualization - via scrolling with the mouse wheel. Most of the time it makes sense to have the whole visualization in sight (to overlook the different clusters), but small zooming motions are often very useful. Figure 7.11 shows two intermediate steps of zooming in on the upper right of the visualization.

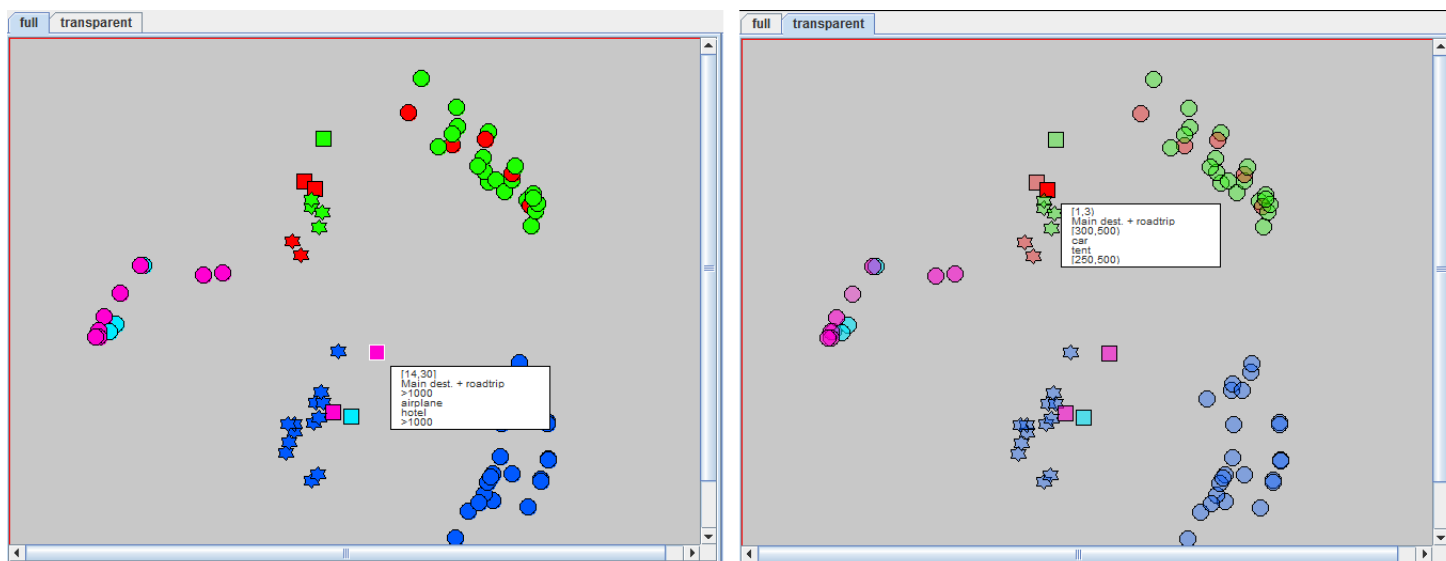


Figure 7.10: Tooltips and color modes. When the mouse enters the panel of a configuration, a small tooltip with the configuration's values is shown. *Left*: the full color mode. *Right*: the transparent color mode - overlapping panels are still fully visible.

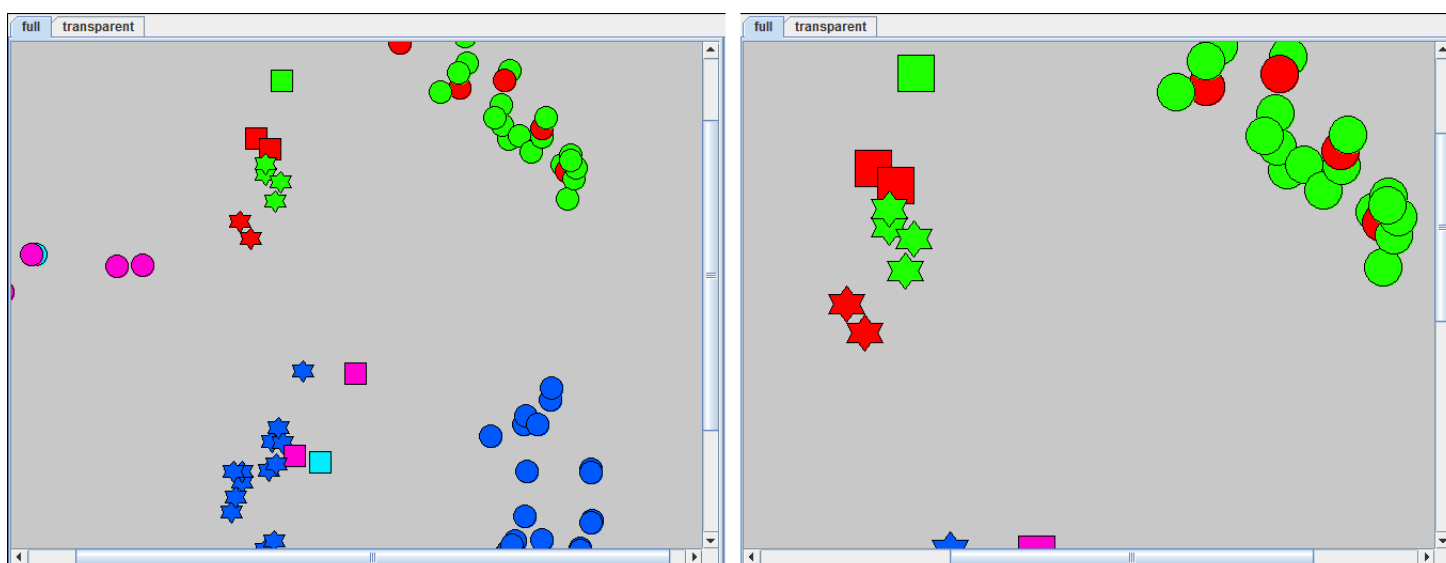


Figure 7.11: Zooming in on the visualization area. Two intermediate steps are depicted. The area allows to flexibly zoom in and out via scrolling with the mouse wheel.

7.1.4 Explore area

The function of the *explore area* on the upper right of the Visual Frame is closely related with the visualization area.

In the visualization area, single panels of configurations can be selected (and again be deselected) via mouse-click. Selected panels have white borders. Groups of configuration panels can be selected by drawing a selection frame with the mouse. When pressing control on the keyboard, additional panels can be added to the current selection. Otherwise a new selection makes the old selection be dropped. Figure 7.12 shows the selection of a single configuration (rather centrally located, with the white border) - and with this already the reaction of the explore area to selections on the visualization area: the explore area shows which kinds of values prevail in the current selection. If only one configuration is selected, this is pretty obvious: on all parameters' slots there is always just one value - and so always 100% have the respective value. So with the selection of one single configuration, one can just read the values of this configuration - but more conveniently and more permanently than in the tooltip.

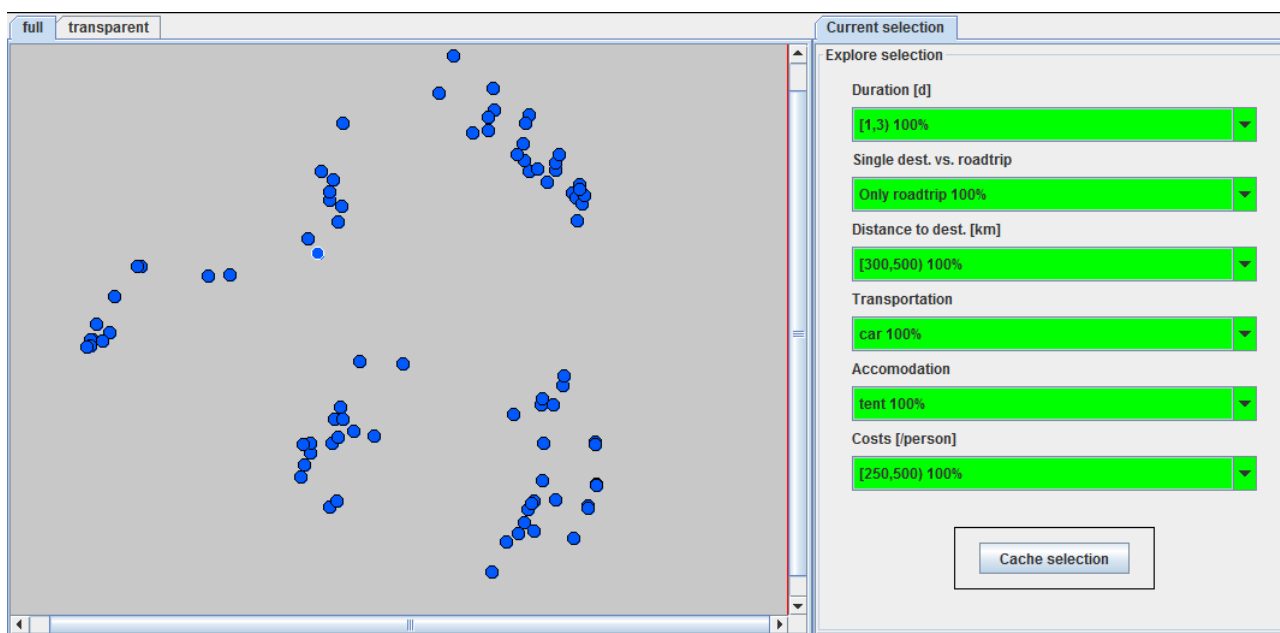


Figure 7.12: Explore area - selection of a single configuration.

It gets more interesting when a group of configurations is selected. In Figure 7.13 the cluster at the bottom right of the visualization has been selected. The explore area then shows which percentages of the selected configurations have a certain parameter value. The prevailing value, the value with the highest frequency, is listed at the top. The other values and their percentages can be looked up by clicking on the drop-down list. So, for example, all of the selected configurations have the value “[0,1)” in the “Duration”-slot. However, when it comes to “Distance to dest.” there are three equally frequent values in the set.

The coloring³ gives the user an immediate feedback, if there are values the majority of the selected configurations have in common - and if so, which. As already formulated above: the high-end goal for GMAvisual, and for GMA-software in general, would be to find a way to move the focus from single configurations to groups of configurations - and to do this grouping somehow automatically. Realizing this via cluster visualizations has the advantage that the user can still interactively, by and by, explore the clusters and decide which make sense for him and which don't. The homogeneity rates can be a good indicator for the quality of clusters.

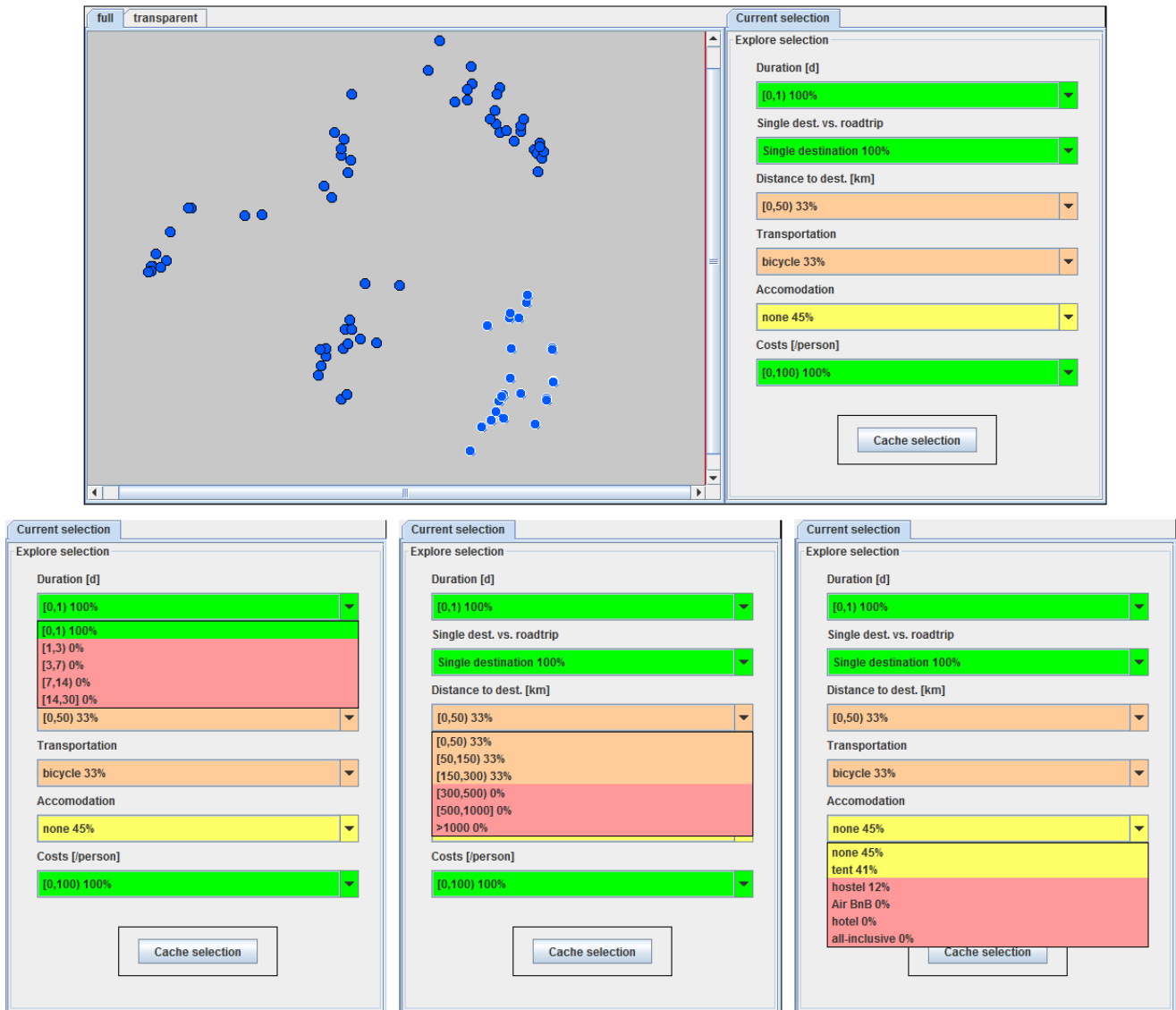


Figure 7.13: Explore area: selection of a group of configurations. The cluster at the bottom right has been selected by drawing a frame with the pressed mouse. The drop-down lists in the explore area show the relative frequencies of the different parameter values in percent.

³100% - 80%: green; 79% - 60%: olive; 59% - 40%: yellow; 39% - 20%: orange; 19% - 0%: red.

In Figure 7.13 all selected holiday configurations have a duration of less than 24 hours, cost less than 100€ and have a single destination. Looking at the distances (which first appear rather heterogeneous), it turns out that all have a travel distance of less than 300km. In 45% of the cases it's a day trip without the night, in 41% the trip will include a night in the tent, and in 12% in a hostel. The transportation-slot is pretty equally shared between bicycle, train, car and walking. So, examining the simple frequency distributions tells us that the cluster at the bottom right contains only configurations that represent cheap day trips to a single destination nearby.

Also most of the other discernible clusters represent distinct concepts of holidays. In this example problem the categories into which the MDS-clustering partitions the reduced configuration space are not very surprising - after all everybody knows that there are cheap day trips and long airplane journeys. However this example shows that bringing together GMA and MDS does work - in at least one case. If this would work for all or at least for the vast majority of GMA-problems, a method to further reduce the reduced configuration space to few categories of problem solutions would already have been found.

Caching selections

With a mouse-click on the “Cache selection”-button, the current selection can be cached. A small dialog asks for a name of the selection and then a tab is added to the explore area. Then other clusters (or of course single configurations) can be selected and examined in the explore area (tab “Current selection”) and the user can always jump back to the cached selection - with a click on the respective tab and without having to manually select it again. In Figure 7.14 three clusters have been cached, some other configurations have been selected and then the tab “Weekend with car and tent” has been clicked.

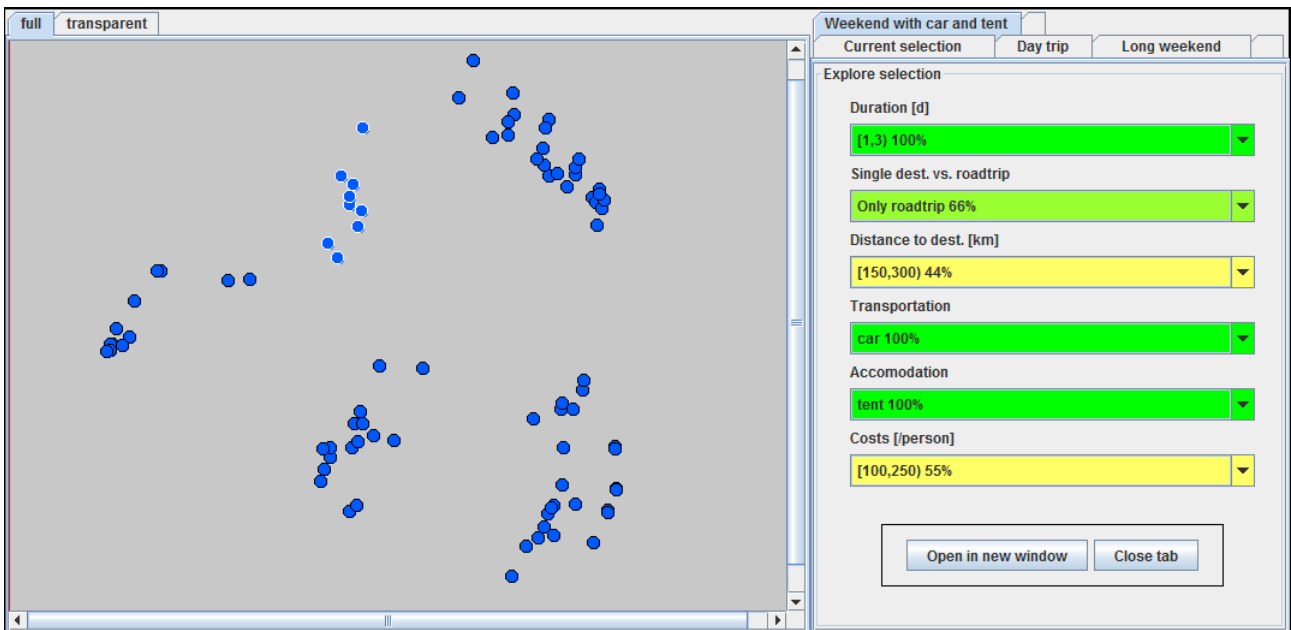


Figure 7.14: Explore area - caching selections. Three clusters have been cached, the cluster “Weekend with car and tent” has been selected.

Opening a selection in a new Visual Frame

A selection of configurations that has been cached can be opened in a new Visual Frame, so in a new window (as the button indicates). *Only* the selected configurations are then transferred to a new project model and can be examined with all the means that have been described so far. Most importantly the MDS is calculated with only the selected configurations. This can be useful in case there are some clear clusters on the one hand - and on the other hand some area in the visualization where no common concept can be found. Selecting this subset of configurations and applying MDS to it might clear the structure. But: MDS does often not perform too well with few data points here. This is probably due to the similarity measuring of the categorical data - which does just not yield very precise information.

To give an example: the cluster in the middle of the visualization area is not homogeneous. There are very short and very long trips, trips by bicycle and by airplane, very cheap and very expensive trips - judging by the relative frequencies of the explore area it is a big mixture of everything. In Figure 7.15 one can see how classical MDS lays out only the configurations of this cluster. The round ones at the left are all cheap, one-day roadtrips by bike or car, with little travel distance (green: sleeping in a tent; blue: not staying the night). The red rhombi at the right are long, expensive trips of the kind “Main dest. + roadtrip”. The reason why they appear mixed, in the center of the original visualization, seems to be that they are all no single-destination holidays and therefore don’t belong to one of the clear, outer clusters.

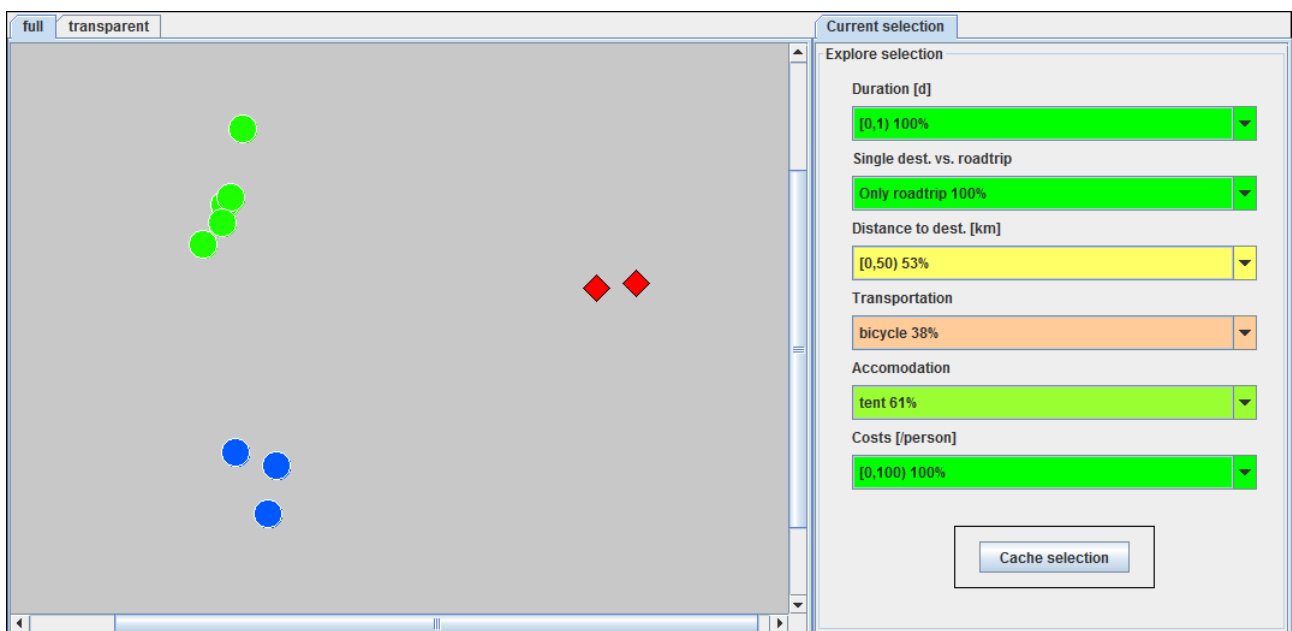


Figure 7.15: Opening a selection in a new Visual Frame. The heterogeneous cluster in the center of the original visualization has been selected and opened in a new window (circles: duration [0,1); rhombi: duration [14,30). green: tent; blue: no accommodation; red: hostel.)

Automatic clustering support

In the clustering field of the option area, the user can select a type of clustering and the number of wished clusters. Then the selected cluster algorithm calculates the wished number of clusters and the configurations belonging to the same calculated cluster get all colored in the same color.⁴ This function is no fundamental enhancement, but can be of great convenience.

In Figure 7.16 the k-means algorithm has calculated five clusters. As this algorithm works on the two-dimensional coordinates that have been calculated by MDS, the found clusters correspond with the visually discernible ones. When such a clustering is calculated, the clusters are automatically cached - being named by their colors. That way the user does not have to select and cache all the discernible clusters manually, but can directly jump from tab to tab respectively from cached selection to cached selection.

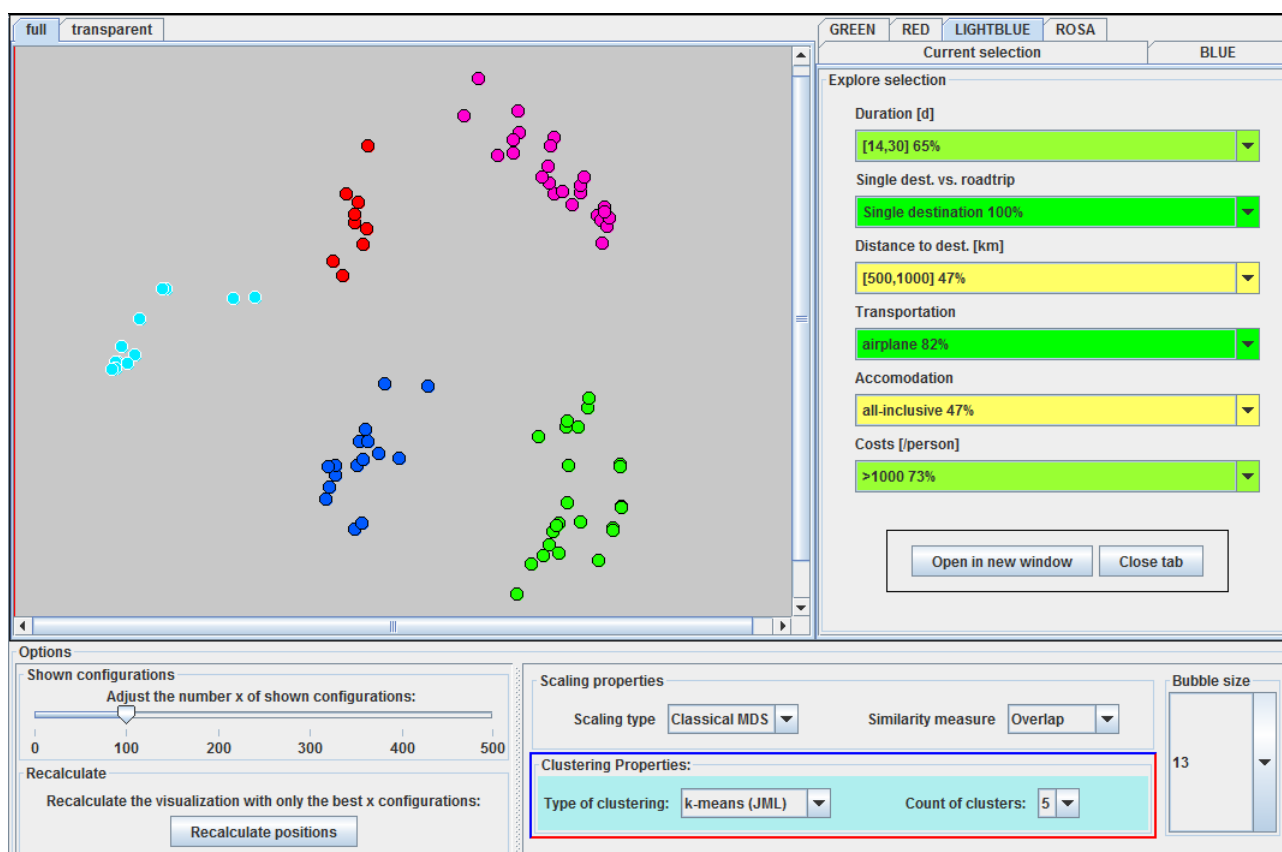


Figure 7.16: Automatic clustering support. Five clusters have been calculated via the k-means algorithm. Configurations belonging to the same calculated cluster get the same color and the clusters are cached as selections.

⁴If the parameter value coloring has been active, it is turned off - and vice versa.

7.2 MDS with different similarity measures

In Section 6.2 three of the fourteen similarity measures for categorical data from [BCK08] have been presented: the overlap measure, the IOF measure and the OF measure. As already mentioned above, GMAvisual allows the user to flexibly choose a similarity measure. With a click on an item in the respective drop-down list, a similarity measure is chosen whereupon the visualization is recalculated immediately - with the indicated scaling type and the chosen similarity measure. Evaluating the performance of similarity measures in this context is pretty challenging. What makes a visualization better than another one?

Some similarity measures might effect that more distinct clusters are discernible (than there were with other measures). As the goal is to partition the reduced configuration space into categories, this is at first hand desirable and could be a quality criterion. But do distinct clusters automatically represent solution categories that make sense? Maybe with another similarity measure the clusters would not be so distinct, but would indicate a better partitioning. - The question of a best-practice similarity measure for MDS with GMA could only be answered with a large user study that lets users analyze their real decision or strategy-planning problems with GMA and lets them then evaluate various visualizations. Here there shall only be given some examples of how it can look like if a GMA project is visualized with MDS and different similarity measures.

Figure 7.17 again shows the MDS-embedding of the 100 best-ranking holiday configurations. The subfigure in the middle is the well-known constellation - calculated with the overlap similarity measure. The subfigure at the bottom has been calculated with classical MDS and the IOF measure, the subfigure at the top with the OF measure. In the OF subfigure at the top, five clusters have been marked with colors (via the automatic clustering function with the k-means algorithm). The left half of the blue cluster has additionally been selected to the explore area as “Current selection”. The other two subfigures have each been calculated after that, so that the selected configuration sets (including the coloring) stayed the same. This way one gets an idea where which configurations/clusters are placed with which similarity measure.

The clusters that can be distinguished in the OF visualization at the top and that have been the basis for the coloring pretty well represent some common holiday categories and are best described as follows:

- **Pink:** long, expensive trip of the type “Main dest. + roadtrip”.
- **Red:** long, expensive trip of the type “Single destination”.
- **Dark blue - left part:** cheap day trip of the type “Only roadtrip”.
- **Dark blue - right part:** cheap day trip of the type “Single destination”.
- **Green:** weekend with car and tent - of the type “Only roadtrip”.
- **Light blue:** weekend trip of the type “Single destination” and “Main dest. + roadtrip” (the three light blue bubbles right at the top).

So, the OF visualization seems to do a pretty good job. And except for the fact that the three light blue bubbles right at the top should have been put into a proper cluster, also the coloring corresponds to the well-known holiday categories. Let's now examine the overlap visualization (in the middle of Figure 7.17) and the IOF visualization (at the bottom of Figure 7.17).

In the overlap visualization pretty much the same clusters could be distinguished. There is the cluster at the bottom right (dark blue), the one at the top right (light blue), and the one at the left (red). Clustering the "Only roadtrip" weekends together with the "Main dest. + roadtrip" weekends at the top in the middle is definitely not worse than the layout of the OF visualization. But the cluster at the bottom in the middle mixes two concepts: the "dark blue - left" configurations can not visually be distinguished from the "pink" ones.⁵ So one could say that - in this specific example and with this specific size of the reduced configuration space - the OF similarity measure performs better than the overlap similarity measure.

Also the IOF similarity measure performs pretty well (depicted in the bottom subfigure of Figure 7.17). The two kinds of dark blue clusters can be distinguished, the green cluster is placed next to the light blue configurations of the type "Main dest. + roadtrip" and the pink cluster could probably also be found. But all the area in the middle will probably not be partitioned into the same clusters as have been found with the other two similarity measures. These few clear categories would not be found - but instead there are just more, smaller clusters. In this case, the case of the Holiday Trip problem, this finer clustering in the center of the IOF visualization does not bring new insights or some better categorization. So one would probably judge the IOF visualization worst of the three (with only slight differences).

But as it has already been explained in Section 6.2, the performance of the different similarity measures always depends on the data set. To illustrate this, visualizations of another problem have been included. Figure 7.18 shows the overlap visualization (at the top), the OF visualization (at the left) and the IOF visualization (at the right) for one of the example problems that come with the Parmenides EIDOS suite. The three clusters have been color-marked in the overlap visualization and the colors stayed the same for the other two visualizations. As one can see: here the OF measure performs worst. Without the coloring no clusters would be discernible. The IOF visualization is however quite interesting. There even more than three clusters can be distinguished. So in this case, the IOF similarity measure might be the best option.

As it has been stated before: given that most probably there is not one similarity measure that will perform best for all GMA problems, it is good to allow the user to flexibly choose different similarity measures.

⁵This is the situation that has been examined in Section 7.1.4 and Figure 7.15.

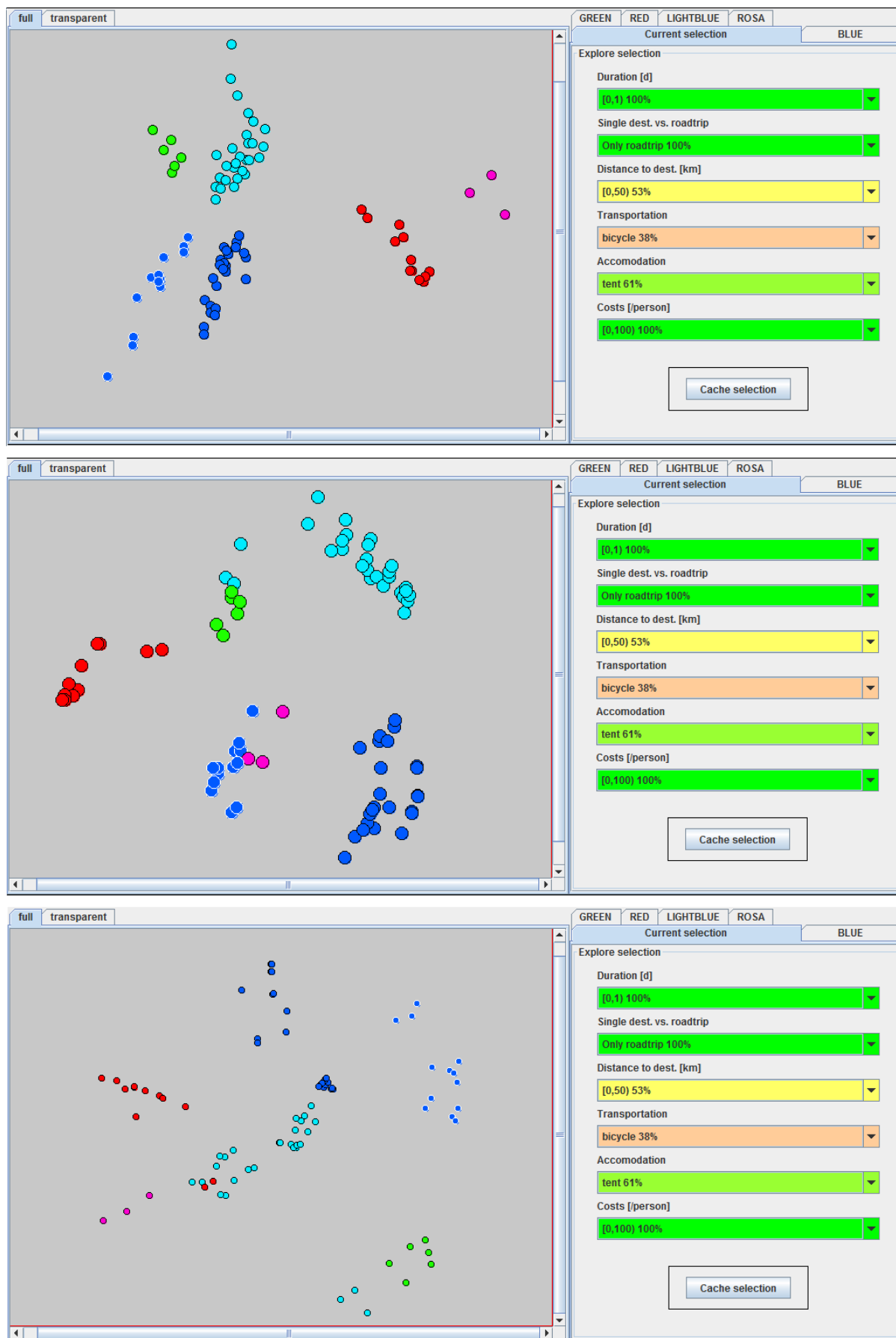


Figure 7.17: MDS with different similarity measures. Holiday Trip problem (best 100).
Top: OF measure. *Middle:* overlap measure. *Bottom:* IOF measure.

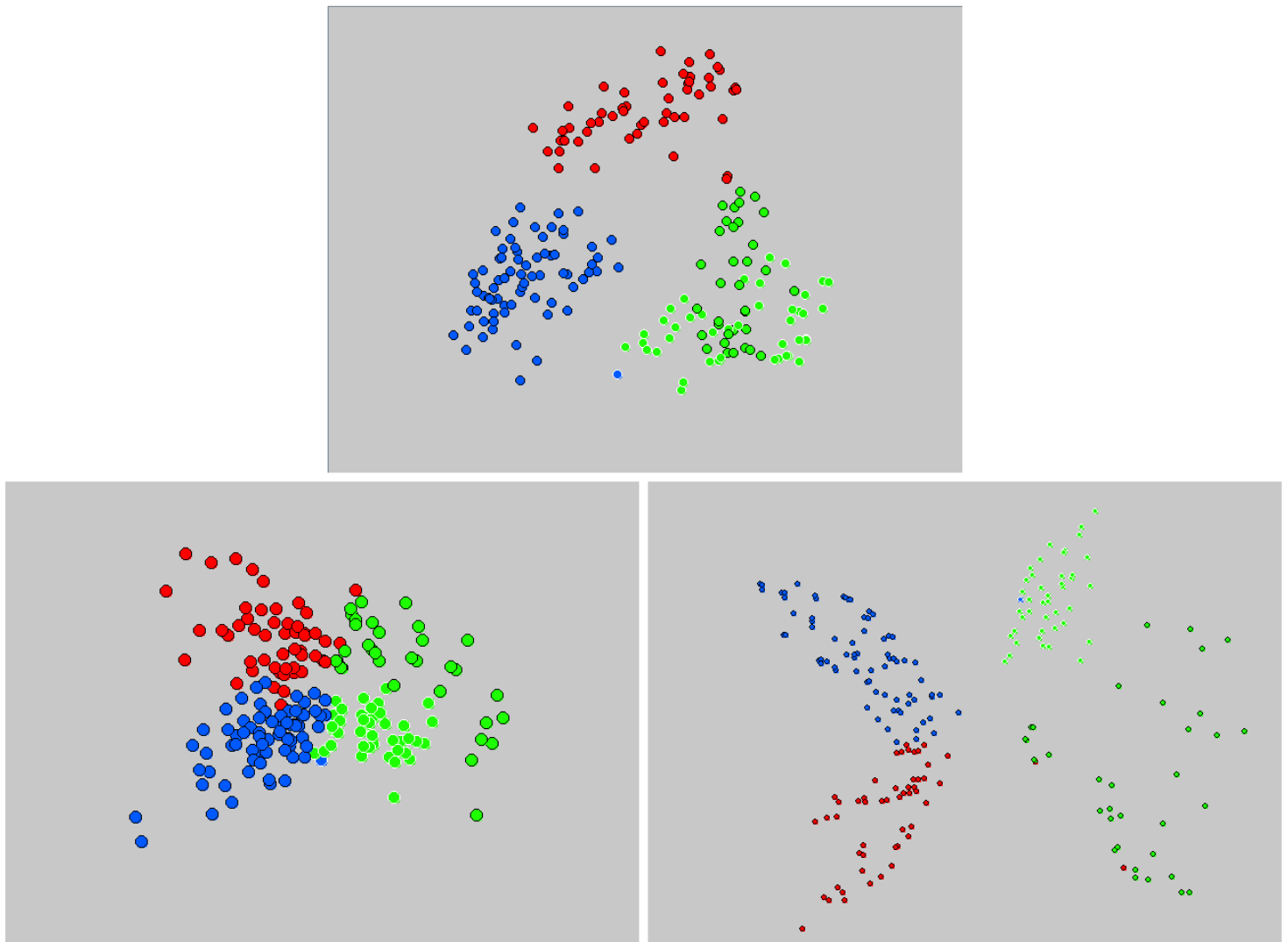


Figure 7.18: MDS with different similarity measures. Example problem from Parmenides EIDOS (best 200). *Top*: overlap measure. *Left*: OF measure. *Right*: IOF measure.

7.3 Other scaling algorithms - comparing MDS to t-SNE

Obviously there is not only classical Multidimensional Scaling, but a huge number of scaling and dimensionality reduction algorithms, developed over the time by various scientists for various purposes in various fields. The goal of this thesis was to examine the principal suitability of classical MDS for visualizing GMA configuration spaces. If other algorithms might be suited as well and especially a comparative evaluation of different algorithms has to be left to future investigations. However GMAvisual could be a platform for such investigations, as it also allows to select a type of scaling (in the option area, alongside the similarity measure selection). When a type of scaling is selected from the drop-down list, the visualization is automatically and immediately recalculated (with this scaling

type and the indicated similarity measure). To illustrate this, Laurens van der Maaten's t-Distributed Stochastic Neighbor Embedding (t-SNE) has been integrated into GMA-visual - as just one example of an alternative to classical MDS. On [vdM15] (where also the code of the algorithm can be found) it is stated that t-SNE is "a (prize-winning) technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets". Giving it a try, it turns out that classical MDS might not perform so bad after all: the same groups of configurations are discernable as clusters in the visualization. Figure 7.19 shows the well-known MDS clustering of the 100 best configurations of the Holiday Trip problem. Again the five distinguishable clusters have been color-coded - and then the clustering via t-SNE has been launched. That way all configurations keep their MDS cluster-color. This allows to see where which kinds of configurations land with the t-SNE embedding compared to with the MDS embedding. Obviously - as far as visually distinguishable clusters are concerned - classical MDS and t-SNE have quite the same results.

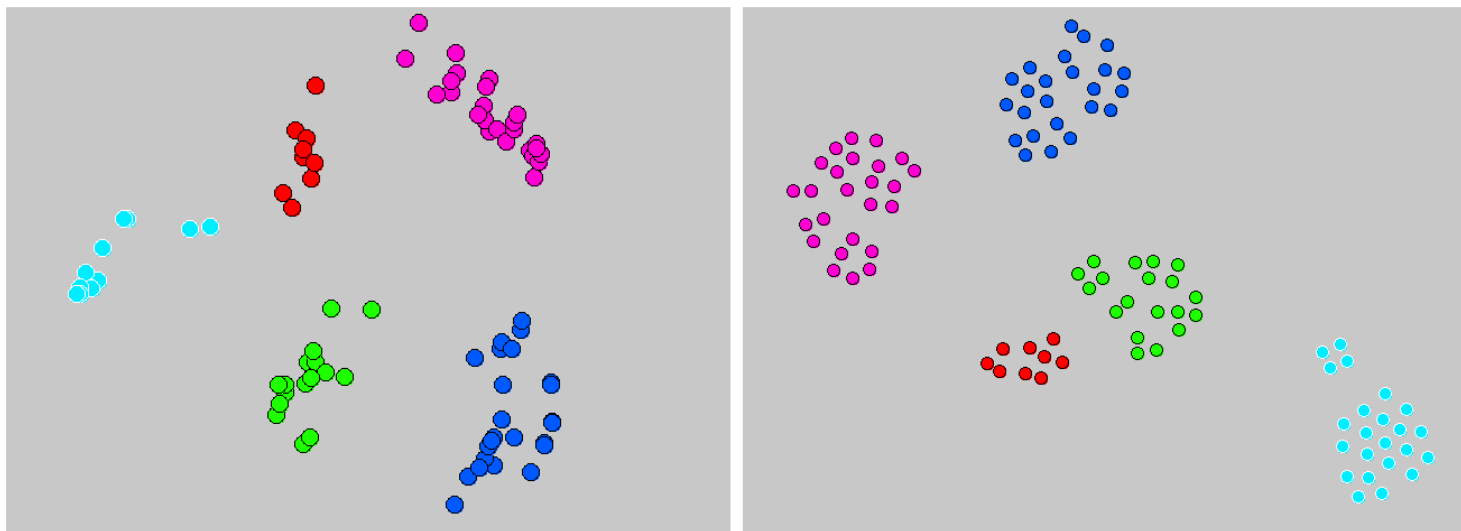


Figure 7.19: Alternative scaling algorithm t-SNE. Visualization of the Holiday Trip problem (best 100 configurations). *Left*: via MDS. *Right*: via t-SNE. The clusters' colors have been set on the MDS visualization, then the t-SNE has been launched - whereby all configurations kept their colors. The result: obviously the same clusters can be distinguished.

8. Conclusion

The backgrounds of General Morphological Analysis have been laid out, as well as the existing approaches of consistency assessment. After that, quite some time has been spent trying to show possibilities of how to improve the consistency assessment (selective Triple Consistency Assessment and weighting functions for scaled consistency assessment). The existing approaches of computer support have been presented (MA/Carma and Parmenides EIDOS), as well as the enhancement of the MA/Carma model with scaled consistency data: the Scaled Inference Model (SIM). Then the thesis turned on the mathematical preliminaries for plotting an n -dimensional categorical configuration space onto two dimensions: classical Multidimensional Scaling and similarity measures for categorical data were outlined. With these preliminaries, a prototype named GMAvisual has been developed that implements MDS for GMA and offers a variety of functions to explore the arising visualizations. These functions and their potential use have been thoroughly explained using screenshots of GMAvisual visualizations of the Holiday Trip example problem - that has been referred to throughout this thesis. The conclusion that could be drawn on the go: at least for this one example problem it does make sense to visualize the reduced GMA configuration space via MDS.¹ As the input to the MDS algorithm is a matrix of dissimilarities between the configurations, the output depends on the chosen similarity measure. The three similarity measures presented in the theory part are therefore then compared in practice: with the two presented cases and for the three selected measures it is not possible to say that one of them always performs best. At the end an MDS visualization has been compared to a visualization via t-SNE: with very different approaches the two algorithms still seem to find similar clusters in GMA configuration spaces. In any way: it will be a good idea for visual GMA software to offer various scaling algorithms and various similarity measures. It is very likely that the success of the selected combination of the two strongly depends on the specific problem

¹It also worked for most of the other available example problems: distinct clusters were discernible in the majority of the cases. However: if these clusters represent useful solution categories or not can only be judged by a person that knows the field of application the problem has been drawn from.

and the specific structure of the reduced configuration space.

So far so good. There is plenty of room for further investigation in the field. For once, the question of the best suited weighting function has not been completely answered. Then the GMA procedure itself and the consistency assessment could be enhanced to not only allow categorical attributes, but also ordered and even continuous ones. With consistency assessment involving the last case, consistency *functions* between parameters would have to be defined instead of filling out the cross consistency matrix. Such an enhancement would be desirable (why break a parameter like “Costs” into distinct categories?), but would of course change the face of the whole procedure and the visual embedding would have to be rethought. A whole other aspect would be, to try to find a good way to depict the triple consistency assessment (maybe with some three-dimensional matrix depiction onto which can flexibly be zoomed in and out).

But even if the GMA procedure itself and the consistency assessment are left as they are, there is a whole bunch of questions: Is there a dimensionality reduction algorithm that serves best for the great majority of GMA problems? If so, which? Is there a similarity measure that performs best with this algorithm, and if so, which? Are there maybe various equally-well performing algorithm-similarity measure pairs? And last but not least: Does visualizing the reduced configuration space in a two-dimensional clustering help at all in analyzing real-world problems - or does it just look nice? - All these questions can not be answered without a large and long user study. People that can actually make use of analyzing the problems they work on with GMA need to work with the visualizations, with different scaling types and with different similarity measures - and then need to evaluate the sense that the arising clusters make, bearing in mind that they should in the best case represent different solution categories.

Bibliography

- [BCK08] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. In *SDM 2008: Proceedings of the 8th SIAM International Conference on Data Mining, Atlanta*, pages 243–254, 2008. (cited on Page 50 and 69)
- [BG97] Ingwer Borg and Patrick Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag New York, Inc., 1997. (cited on Page 46)
- [CC01] Trevor F. Cox and Michael A.A. Cox. *Multidimensional Scaling*. Chapman & Hall/CRC, 2001. (cited on Page 46 and 48)
- [CCS94] R.G. Coyle, R. Crawshay, and L. Sutton. Futures assessment by field anomaly relaxation: A review and appraisal. *Futures*, 26(1):25–43, 1994. (cited on Page 13)
- [Fou15a] Parmenides Foundation. Parmenides EIDOS - visual reasoning and knowledge representation, 2015. Available at https://www.parmenides-foundation.org/fileadmin/redakteure/PDFs/1106_EIDOS_Folder_e.pdf. [Online; accessed 08-September-2015]. (cited on Page 33)
- [Fou15b] Parmenides Foundation. Parmenides EIDOS - website, 2015. Available at <https://www.parmenides-foundation.org/application/parmenides-eidos/>. [Online; accessed 08-September-2015]. (cited on Page 33)
- [Rhy74] Russell Rhyne. Technological forecasting within alternative whole futures projections. *Technological Forecasting and Social Change*, 6:133–162, 1974. (cited on Page 12)
- [Rhy95] Russell Rhyne. Field anomaly relaxation - the arts of usage. *Futures*, 27(6):657–674, 1995. (cited on Page 12)
- [Rit98] Tom Ritchey. Fritz Zwicky, “Morphologie” and policy analysis. In *16th EURO Conference on Operational Analysis, Brussels*, 1998. (cited on Page 3, 7, and 9)
- [Rit03] Tom Ritchey. MA/Carma - advanced computer support for general morphological analysis, 2003. Available at <http://www.swemorph.com/pdf/>

- macasper1.pdf. [Online; accessed 15-May-2015]. (cited on Page vii, 29, 30, 31, and 32)
- [Rit06a] Tom Ritchey. Modelling multi-hazard disaster reduction strategies with computer-aided morphological analysis. In *Reprint from the Proceedings of the 3rd International ISCRAM Conference, Newark, NJ*, 2006. (cited on Page 9, 11, and 15)
- [Rit06b] Tom Ritchey. Problem structuring using computer-aided morphological analysis. *Journal of the Operational Research Society*, 57(7):792–801, 2006. (cited on Page 3, 4, 7, and 9)
- [Rit09] Tom Ritchey. Threat analysis for the transport of radioactive material. *Packaging, Transport, Storage & Security of Radioactive Material*, 10(1):24–29, 2009. (cited on Page 9)
- [RSE02] Tom Ritchey, Maria Stenström, and Håkan Eriksson. Using morphological analysis for evaluating preparedness for accidents involving hazardous materials. *Swedish Morphological Society*, 2002. Adapted from an article presented at the 4th International Conference for Local Authorities, Shanghai. (cited on Page 15)
- [vdM15] Laurens van der Maaten. t-SNE - website, 2015. Available at <http://lvdmaaten.github.io/tsne/>. [Online; accessed 17-September-2015]. (cited on Page 73)
- [Wic03] Florian Wickelmaier. An introduction to MDS. *Sound Quality Research Unit, Aalborg University, Denmark*, 2003. (cited on Page 46)
- [Wik15] Wikipedia. Likert scale — Wikipedia, the free encyclopedia, 2015. Available at https://en.wikipedia.org/wiki/Likert_scale. [Online; accessed 16-July-2015]. (cited on Page 19)
- [ZDSM14] Marin Zec, Peter Dürr, Alexander W. Schneider, and Florian Matthes. Improving computer-support for collaborative business model design and exploration. In *Proceedings of the 4th International Symposium on Business Modeling and Software Design (BMSD), Luxemburg*, 2014. (cited on Page 4 and 7)

Eidesstattliche Erklärung:

Hiermit versichere ich an Eides statt, dass ich diese Bachelorarbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie dass ich die Bachelorarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Julius Kempf

Passau, den 21. September 2015