



**University of
Zurich**^{UZH}

Design and Implementation of a Prototype for TrAdeCIS

*Marc Heimgartner
Zurich, Switzerland
Student ID: 11-729-423*

Supervisor: Radhika Garg, Dr. Thomas Bocek
Date of Submission: September 30, 2015

Zusammenfassung

Entscheidungsfindung ist ein essentieller Teil unseres täglichen Lebens. Häufig wird jedoch der Entscheidungsfindungsprozess von äusseren Einflüssen, wie zum Beispiel Marketing oder Design beeinflusst, was darin resultiert, dass nicht mehr die optimale Entscheidung getroffen wird. In anderen Fällen ist es für eine Person einfach zu komplex die optimale Lösung zu finden, da es zu viele Kriterien gibt, welche beachtet werden müssen. Das Gebiet der multikriterielle Entscheidungsanalyse (MCDA) arbeitet an Algorithmen um die vorherig genannten Probleme zu beheben und somit die bestmögliche Lösung zu finden. Obwohl für viele Probleme diese Algorithmen nützlich sind, profitieren die mit einer hohen Anzahl an Kriterien und Alternativen am meisten. Ein gutes Beispiel dafür ist Cloud Computing (CC) wo es für einen Endnutzer praktisch unmöglich ist die optimale Lösung selbst zu finden, wegen der grossen Anzahl von Cloud Anbietern, welche wiederum verschieden Vertragsmodelle haben. Darum bietet Trade-off-Based Adoption Methodology for Cloud-Based Infrastructures and Services (TrAdeCIS) eine Lösung an, um den Nutzer bei der Entscheidungsfindung zu helfen. TrAdeCIS basiert auf zwei Algorithmen, Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), welcher benutzt wird für die technischen Anforderungen und Analytic Network Process (ANP), um die Business Leistungs Metriken (BPM) zu werten. Diese Arbeit implementiert TrAdeCIS als eine Webapplikation mit zusätzlicher Hilfe um Cloud Anbieter zu werten, in dem mögliche Kriterien vorgeschlagen werden. Beide zuvor genannten Algorithmen werden implementiert und durch ein Model abstrahiert, welches hilft die nötigen Eingaben zu vollziehen. Die Resultate für TOPSIS, wie auch ANP, werden in Diagrammen visualisiert und ein optionaler Tradeoff Modus kann benutzt werden um allfällige Anpassungen / Änderungen vorzunehmen, für den Fall, dass TOPSIS und ANP nicht das gleiche Result liefern. Dadurch wird später angezeigt was und um welchen Wert angepasst wurde, um das schlussendliche Resultat zu erhalten.

Abstract

Decision making is an essential part of our lives and has to be done on a regular basis. Often the process of decision making is influenced by other subjects like for example marketing and design and as a result not the optimal solution is chosen. In other cases it is simply too complex for a user to find the optimal solution by themselves because of too many possible criteria which should be taken into consideration. The field of Multiple-Criteria Decision Analysis (MCDA) works on algorithms to solve the previous mentioned problems and as a result find the best possible solution for a given multi-criteria problem. While certainly a lot of fields profit from such algorithms, they give the most value in assessing problems with numerous alternatives and criteria. One good example for such a field is Cloud Computing (CC), where it is almost impossible for an enduser to find the best solution, because of the many different providers which also have a large number of different possible subscription plans. Therefore Trade-off-Based Adoption Methodology for Cloud-Based Infrastructures and Services (TrAdeCIS) proposes a solution to find the best possible solution based on two multicriteria decision algorithms. Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) is used to assess the technical requirements and Analytic Network Process (ANP) is used to rank Business Performance Metrics (BPM). This thesis implements TrAdeCIS as a web application with additional help for assessing cloud providers by suggesting a list of possible criteria which could be compared. Both algorithms are implemented and abstracted by a model to simplify the process of entering the needed input for the algorithms. Finally the result is visualized for both TOPSIS and ANP and stored for later use. If both algorithms do not give the same result, an optional tradeoff mode can be used which will store how big the adjustments / compromises were to achieve a final solution. This does later on show what and by how much had to be adjusted in order to find the same result in TOPSIS as well as ANP.

Acknowledgments

I would like to thank Prof. Dr. Burkhard Stiller and the Communication Systems Group (CSG) for letting me work on this interesting and challenging bachelor thesis.

Special thanks go to my supervisor Radhika Garg, without her help and patient support the completion of this thesis would have certainly not been possible.

Finally I would like to thank all the people who supported me.

Contents

Zusammenfassung	i
Abstract	iii
Acknowledgments	v
1 Introduction	1
1.1 Description of Work	2
1.2 Thesis Outline	2
2 Related Work	5
2.1 Multi-attribute Decision Algorithms	5
2.1.1 Analytic Hierarchy Process	5
2.1.2 Analytic Network Process	6
2.1.3 Multi-Attribute Utility Theory	8
2.1.4 Measuring Attractiveness by a Categorical Based Evaluation Technique	9
2.1.5 Elimination Et Choix Traduisant la Realite	9
2.1.6 Preference Ranking Organization Method for Enriched Evaluation .	10
2.1.7 Technique for Order of Preference by Similarity to Ideal Solution .	10
2.1.8 Data Envelopment Analysis	12
2.2 Existing Decision Systems	12
2.3 Selection of Algorithms for TrAdeCIS	13

3	Architecture	15
3.1	Used Frameworks	15
3.1.1	Django	15
3.1.2	Chart.js	16
3.1.3	Cytoscape.js	16
3.1.4	Webpack	16
3.2	Architecture Outline	17
3.3	Architectural Decisions	19
3.3.1	TOPSIS	19
3.3.2	ANP	20
3.4	Creation walkthrough	23
3.5	Implementation Details	27
4	Stress / Performance Testing and Evaluation	31
4.1	Algorithm tests	31
4.2	Load up Time	35
5	Summary and Conclusions	37
	List of Figures	40
	List of Tables	42
A	Installation Guidelines	45
A.1	Development	45
A.2	Deployment	46
B	Contents of the CD	47

Chapter 1

Introduction

Decision making is a process which people face in their daily routines professional and private lives. A human resource manager, for example, may need to evaluate potential candidates for a job. A household may need to choose an insurance company. A person has to decide what he/she wants to eat for lunch. These examples show that decision problems arise frequently. The complexity of decision problems grows quite fast, as usually several criteria are taken into consideration.

Decision problems can be organized in the following categories [9]:

1. The choice problem: With the goal to select the single best option or reduce the initial set to equally “good” options. For example in human resource management where the right person has to be selected for a particular job.
2. The sorting problem: The options are sorted into ordered and predefined groups, called categories. The goal is to cluster options with similar characteristics. For instance, people by age: “children”, “adolescents”, “adults”, “middle age”, “pensioner”.
3. The ranking problem: Options are ordered from best to worst by means of scores or pairwise comparisons, etc. An example would be a ranking list for the best product.
4. The description problem: The goal is to describe options and their consequences. This is usually done in the first step to understand the characteristics of the decision problem.

Multi-Criteria Decision Analysis (MCDA) methods exist to aid the decision maker in finding a solution. Assessing an optimal cloud provider is exactly a case where MCDA can help greatly. The decision to change from a legacy system to a cloud based or to switch from a cloud provider to another is not a trivial task. Trade-off-Based Adoption Methodology for Cloud-Based Infrastructures and Services (TrAdeCIS) [8], therefore helps to find an optimal solution. Optimal in the sense that it helps to find the best solution for each user based on their requirements. TrAdeCIS makes use of two MCDA algorithms (a) The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) and Analytic Network Process (ANP). TOPSIS is used to rank alternative solution based

on technical requirements. ANP is used to rank Business Performance Metrics (BPM) which need more complex model to facilitate valuing the influence of the alternatives on the criteria, but also the criteria on the alternatives. While in TOPSIS the alternative solutions are ranked based on the relevant factors from the technical perspective, they are evaluated in ANP from the business value perspective. An introduction into MCDA methods and especially TOPSIS and ANP is given in Section 2.1. Finally a tradeoff is computed between the optimal solution for the technical requirements and the different possible good solution for the BPMs. The result is the solution which is best suited for the specific needs of the user by entering and weighting criteria the user controls the optimal result for his desires.

1.1 Description of Work

The goals are to provide an easy to use TrAdeCIS application which helps in decision finding. Although it is mainly designed for comparing cloud providers, it can also be used for other use cases. Furthermore the application should have an intuitive user interface which guides and abstracts well the different steps necessary to rank the alternatives. Finally it should interactively visualize the results.

This thesis is focused on designing and implementing a reliable, scalable prototype for TrAdeCIS. It therefore incorporates the following tasks:

1. Implementing the needed multi criteria decision analysis algorithms (TOPSIS and ANP)
2. Implementation of the trade-off algorithm
3. Construction of a web-app containing the following features:
 - Three-tier Permission system
 - Visualization of the input and the result (TrAdeCIS is applicable to solve any general decision making problem, but provides additional help for cloud specific analysis)
 - Data Handling and persistency of the results
4. Extensibility, it should be applicable for any case, but provide additional help for cloud specific decisions by suggesting possible criteria for TOPSIS as well as ANP.
5. Testing, stress and performance testing as well as overall bug fixing to provide a reliable product

1.2 Thesis Outline

The structure of this thesis is the following: Chapter 2 provides a introduction to the theory of multi criteria decision analysis, with the focus on the algorithms which are used

by TrAdeCIS. The subsequent Chapter describes the architecture of TrAdeCIS, the used technologies, conventions, as well as detailed information about the implementation of the algorithms discussed in chapter 2. Chapter 4 is about testing and evaluation to make sure that TrAdeCIS is reliable and accurate. The last chapter describes open questions and summarises the results of this thesis, finalized by a conclusion.

Chapter 2

Related Work

2.1 Multi-attribute Decision Algorithms

Multi-criteria decision analysis (MCDA) methods have been developed to solve decision problems. Table 2.1 shows a number of MCDA methods and which kind of problem they can solve (see Chapter 1 for explanation of the problems). This is not an exhaustive list of all the possible methods.

Table 2.1: MCDA Problems and Methods [9]

Choice problems	Ranking problems	Sorting problems	Description problems
AHP	AHP	AHPSort	
ANP	ANP		
MAUT/UTA	MAUT/UTA	UTADIS	
MACBETH	MACBETH		
PROMETHEE	PROMETHEE	FlowSort	GAIA, FS-Gaia
ELECTRE I	ELECTRE III	ELECTRE-Tri	
TOPSIS	TOPSIS		
DEA	DEA		

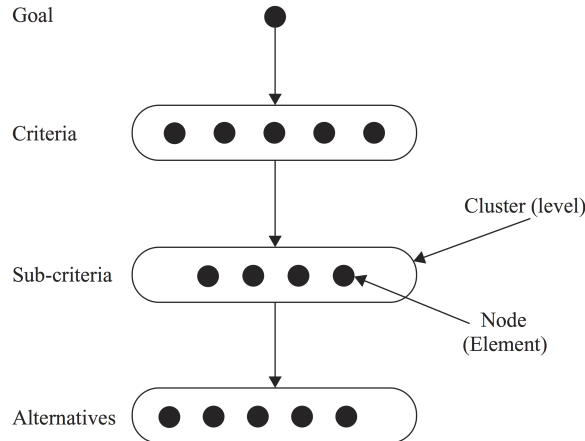
Following a short introduction for each method and a more in detail description for the two chosen methods for TrAdeCIS (TOPSIS and ANP).

2.1.1 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) needs two basic steps to compute a result:

(1) The problem which should be solved with AHP has to be structured according to a hierarchy with the top element representing the goal of the decision. The second level represents the criteria and the lowest level the alternatives, as shown in Figure 2.1. Additional levels of complexity can be added as sub-criteria.

Figure 2.1: Linear Hierarchy of AHP [9]



(2) The priorities are computed based on pairwise comparisons which are provided by the user. Three types of pairwise comparisons have to be computed:

Criteria priorities. Importance of each criterion with respect to goal

Alternative priorities. Importance of an alternative with respect to one specific criterion.

Global alternative priorities. Criteria priorities and alternative priorities are intermediate results to compute the global value.

The pairwise comparisons matrices are evaluated on a scale shown in Table 2.2.

Table 2.2: Pairwise Comparison Scale [9]

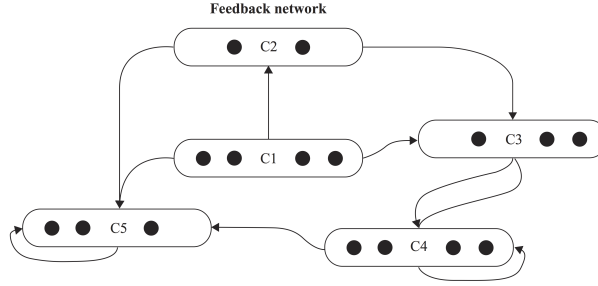
Degree of importance	Definition
1	Equal importance
2	Weak
3	Moderate importance
4	Moderate plus
5	Strong importance
6	Strong plus
7	Very strong or demonstrated importance
8	Very, very strong
9	Extreme importance

2.1.2 Analytic Network Process

The Analytic Network Process (ANP) is a generalization of AHP. It also is the only shown method where dependencies can arise between any of the elements [16]. Other methods

are hierarchical and therefore only allow dependencies along the hierarchy (goal, criteria, sub-criteria, alternatives). Opposed to that Figure 2.2 shows ANP as a network model, where the hierarchical layers are no longer necessary and replaced by clusters of nodes. This allows for two new possible categories of dependencies: inner dependencies of clusters and outer dependency of clusters.

Figure 2.2: Five-Cluster ANP Network [9]



Inner dependencies allow to model a more realistic result if the nodes of a cluster are not independent. For example the evaluation of candidates for job, the possible criteria could be years of experience, programming capabilities and foreign language capabilities. In this case years of experience and programming capabilities could be modeled dependent. This allows that a candidate with high programming capabilities and many years of experience is not higher evaluated as someone with extreme high foreign language capabilities. Dependency can occur in each cluster, even in the alternatives, but this is a rare case.

Dependency of clusters means that there is another dependency between two clusters (usually alternative to criteria). This could for example be the case when the weighting of the decision is dependent on which alternative is chosen.

The computation of ANP can be divided into two main parts:

1. Construction of the super matrix
 - (a) The decision maker enters for each criteria and alternative a pairwise comparison matrix
 - (b) The values for the super matrix are obtained by computing the eigenvectors of the pairwise comparison matrix and writing them as column vectors in the super matrix
2. Computation of the limit matrix from the super matrix
 - (a) Next the super matrix has to be normalized with Equation (2.1) to obtain a column stochastic matrix.

$$r_{ca} = \frac{x_{ca}}{\sqrt{\sum_{a=1}^n x_{ca}^2}} \quad (2.1)$$

- (b) Raise the matrix by the power of 3.

- (c) Repeat the previous two steps until the difference between the matrix from step $n-1$ and step n are smaller than a predefined value.

It can be shown that if the supermatrix is column stochastic the limit exists [16]. The reason why the super matrix is raised to high powers is, that we want to take influences along all possible paths in the supermatrix into consideration and therefore also the indirect influences. The following example should clarify why the matrix is raised by the power of 3. The supermatrix has to be raised by an arbitrary high number until it converges. Table 2.3 shows a simple example supermatrix with two criteria and two alternatives. Table 2.4 shows the result of raising it by the power of 2. Because of the matrix multiplication clusters of the values switched places. By raising it by an odd number the initial structure is preserved Table 2.5. While any odd number works this also increases the complexity and decreases the number of iterations.

Table 2.3: Example Supermatrix

	Criteria 1	Criteria 2	Alternative 1	Alternative 2
Criteria 1	0	0	0.5	0.5
Criteria 2	0	0	0.5	0.5
Alternative 1	0.5	0.5	0	0
Alternative 2	0.5	0.5	0	0

Table 2.4: Resulting Limitmatrix by Power of 2

	Criteria 1	Criteria 2	Alternative 1	Alternative 2
Criteria 1	0.5	0.5	0	0
Criteria 2	0.5	0.5	0	0
Alternative 1	0	0	0.5	0.5
Alternative 2	0	0	0.5	0.5

Table 2.5: Resulting Limitmatrix by Power of 3

	Criteria 1	Criteria 2	Alternative 1	Alternative 2
Criteria 1	0	0	0.5	0.5
Criteria 2	0	0	0.5	0.5
Alternative 1	0.5	0.5	0	0
Alternative 2	0.5	0.5	0	0

2.1.3 Multi-Attribute Utility Theory

Multi-Attribute Utility Theory (MAUT) is based on the theory that a decision maker tries to optimize a function which aggregates all their points of view [9]. This utility function

is called U and has to include all preferences of the decision maker. The decision maker has to construct and tailor U at the beginning of the decision process [11]. The function U provides a way of measuring the preference of the alternatives. In order to evaluate the *utility* of an alternative several predefined criteria can be included to compute the “utility score” $U(a)$ of an alternative. The alternatives A are therefore evaluate like the following:

$$\forall a, b \in A : U(a) > U(b) : a \text{ is preferred to } b \quad (2.2)$$

$$\forall a, b \in A : U(a) = U(b) : a \text{ and } b \text{ are indifferent} \quad (2.3)$$

2.1.4 Measuring Attractiveness by a Categorical Based Evaluation Technique

Measuring Attractiveness by a Categorical Based Evaluation Technique (MACBETH) superficially has much resemblance with AHP both receive pairwise comparisons from the user. They differ in two major points: 1) MACBETH uses an interval scale opposed to the ratio scale which AHP uses. 2) The calculation process behind AHP is different from MACBETH. In order to receive a result three steps have to be completed [11]:

(1) Like in any other MCDA method the first step is to construct the problem, *e.g.*, AHP or ANP. The structure to model MACBETH is a tree or hierarchy with the distinction between criteria and non-criteria nodes. Non-criteria nodes do not directly influence the decision and are only there to help with the evaluation of the criteria nodes. The structure of the tree differs from an AHP criteria tree as it only allows one criterion node to be set between the top of the tree and the leaves.

(2) Computation of scores by pairwise comparisons. In order to receive a result three different types of scores have to be computed:

Weighting criteria. Measure the attractiveness of each criterion to the top goal.

Scores of options. Measure the attractiveness of an option to one specific criterion.

Overall score of options. *Criteria weight* and *option scores* are only intermediate results which are used to compute the overall result.

(3) Finally and optionally a sensitivity analysis can be executed.

2.1.5 Elimination Et Choix Traduisant la Realite

Elimination Et Choix Traduisant la Realite (ELECTRE) belongs to the outranking methods, which are based on pairwise comparison of the options [11]. There exists a number of different ELECTRE methods which can be used to solve different decision problems (see Chapter 1 for explanation of the problems).

Table 2.6: Different ELECTRE Methods [9]

Decision Problem	Method
Choice problem	ELECTRE I
	ELECTRE IV
	ELECTRE Is
Ranking problem	ELECTRE II
	ELECTRE III
	ELECTRE IV
Sorting problem	ELECTRE-Tri-B
	ELECTRE-Tri-C

2.1.6 Preference Ranking Organization Method for Enriched Evaluation

Preference Ranking Organization Method for Enriched Evaluation (PROMETHEE) is based on the computation of preference degrees which is a score between 0 and 1. It expresses the preference of one solution over another, where 0 stands for no preference, 1 for strong preference and something between 0 and 1 for an intensity of preference. The preference degrees are unidirectional, meaning that preference of action B over action A can not be deduced by the action A over B.

This method includes three main steps [9]:

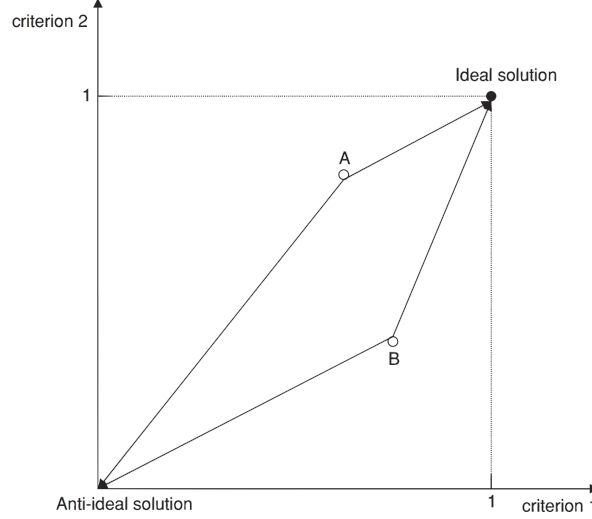
- (1) Computation of preference degrees: The decision maker can choose between two types of preference functions: a linear function, where the preference will gradually increase or gaussian function, where the increase will be exponential. This basically limits the possible values which a preference degree can take.
- (2) Computation of unicriterion flow: Because a conclusion out of the preference degree tables is non trivial, the unicriterion flow is computed, which measure how an action is preferred over all other actions or how it is preferred by all other actions.
- (3) Computation of global flow: The previous step was done to value each criterion, to receive a final result all those values have to be taken together. The decision maker has to provide relative importance for each criterion to show the importance of a criterion over another.

2.1.7 Technique for Order of Preference by Similarity to Ideal Solution

Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) is based on the concept that geometrically the optimal solution is that one, which has the shortest distance to the best solution and the longest distance to the worst solution [8], see Figure 2.3. TOPSIS allows for compensation between different criteria, meaning that high negative

criteria can be balanced by an equal strong positive criteria. The possible alternatives are first weighted, then for each criteria the scores are normalized and finally the distance between each alternative and the ideal solution, as well as the anti ideal solution, are computed. The process of the computation is summarized in the following steps [17]:

Figure 2.3: Geometrical Representation of TOPSIS [9]



1. Construct a decision matrix of n alternatives a with respect to m criteria c so that $X = (x_{ca})$, where $c = 1, \dots, m$ and $a = 1, \dots, n$
2. Normalize matrix and weights

$$r_{ca} = \frac{x_{ca}}{\sqrt{\sum_{a=1}^n x_{ca}^2}} \quad (2.4)$$

3. Construct weighted normalize matrix

$$v_{ca} = w_c * r_{ca} \quad (2.5)$$

4. Determine positive ideal solution

$$A^+ = (v_1^+, \dots, v_m^+) \quad (2.6)$$

and negative ideal solution

$$A^- = (v_1^-, \dots, v_m^-) \quad (2.7)$$

where $v_c^+ = \max_a(v_{ca})$ if criteria c has a positive effect or $v_c^+ = \min_a(v_{ca})$ if criteria c has a negative effect. Likewise $v_c^- = \min_a(v_{ca})$ if criteria c has a positive effect or $v_c^- = \max_a(v_{ca})$ if criteria c has a negative effect.

5. Calculate the distance for each alternative to the ideal solution A^+

$$d_a^+ = \sqrt{\sum_c (v_c^+ - v_{ca})^2} \quad (2.8)$$

and to the anti-ideal solution A^-

$$d_a^- = \sqrt{\sum_c (v_c^- - v_{ca})^2} \quad (2.9)$$

6. Compute the relative closeness coefficient for each alternative:

$$C_a = \frac{d_a^-}{d_a^+ + d_a^-} \quad (2.10)$$

2.1.8 Data Envelopment Analysis

Data Envelopment Analysis (DEA) is defined as [9]:

Firm efficiency is defined as the ratio of the sum of its weighted outputs to the sum of its weighted inputs.

This is another terminology but resemblance other MCDA algorithms, *e.g.*, TOPSIS. Firm can be considered as alternatives, outputs as criteria to be maximized and inputs as criteria to be minimized. The difference to other algorithms is that the weights assigned to outputs and inputs are not allocated by the user. Each firm has its own set of weights which are calculated by a linear optimization function.

DEA provides the following features to the decision maker:

- It indicates the efficiency of a firm by computing a score, which says if the firm has room for improvement or is completely efficient.
- It calculates the difference by which the firm should be increased or decreased in order to become efficient.
- It indicates if the size of the firm change in order to minimize the average total cost.
- It indicates which other firms need to be analyzed in order to optimize itself.

2.2 Existing Decision Systems

While there certainly already exist decision systems which incorporate MCDA methods, *e.g.* M-MACBETH [5] or Super Decisions [7], they tend to have shortcomings in either that they use one single MCDA algorithm to assess both technical requirements and business performance metrics or they do not provide additional help for Cloud Computing. For the reasons showed in the next section it is not favorable to use one algorithm for both, because either the complexity is higher or the MCDA algorithm has limitations in what it can model.

2.3 Selection of Algorithms for TrAdeCIS

Choosing the right MCDA method to apply to solve a problem is not easy to justify. There is a great variety of MCDA procedures with different underlying theory basis. Currently there is no possibility of deciding whether one method makes more sense than another in a specific problem situation. A possible way to decide which method is best suited is to consider the needed input, the data and parameters, the effort to compute the result and the form and validity of the output. Table 2.7 shows a comparison of methods ranked after the effort for the input.

Table 2.7: Required Inputs for MCDA Ranking or Choice Method [9]

MCDA method	Effort input	Input	Output
MAUT	Very HIGH	utility function	Complete ranking with scores
ANP		pairwise comparisons on a ratio scale and interdependencies	Complete ranking with scores
MACBETH		pairwise comparisons on an interval scale	Complete ranking with scores
AHP		pairwise comparisons on a ratio scale	Complete ranking with scores
ELECTRE		indifference, preference and veto thresholds	Partial and complete ranking (pairwise outranking degrees)
PROMETHEE		indifference and preference thresholds	Partial and complete ranking (pairwise preference degrees and scores)
TOPSIS	Very LOW	ideal and anti-ideal option	Complete ranking with closeness score
DEA		no subjective inputs required	Partial ranking with effectiveness score

It gets obvious from Table 2.1 and from Table 2.7 that TOPSIS is the best alternative for the evaluation of the technical requirements in TrAdeCIS. It has minimal effort for input and the output is a complete ranking of the alternatives. Since we want the best possible solution for the technical requirements we can simply take the top result of TOPSIS as the result for our technical requirements.

All the shown methods apart from ANP assume that the criteria are independent (see Section 2.1.2). ANP is able to model dependent criteria and therefore results in better results in such cases and also leaves the possibility for the decision maker which interrelations the different clusters have. Therefore even though the computation is more complex

than other MCDA algorithms, it leaves the freedom open to model more complex graphs, resulting in better / more accurate results.

Chapter 3

Architecture

3.1 Used Frameworks

This section shows the frameworks which were used to build TrAdeCIS. All the libraries which were used in the frontend are introduced in the Section 3.1.4.

3.1.1 Django

Django is an open source full stack web framework in Python. Compared to other more light-weight alternatives, Django provides a lot of features, which simplify the process of creating a web application and therefore don't need to be reinvented. For example, it includes an Object-Relational Mapper (ORM) with which models for the database can be created and queries executed. If the database has to be exchanged it is a simple question of changing settings, all the models which were created with the ORM stay exactly the same for all databases. Furthermore, Django provides a templating language, an authentication and permission system and form classes, which allow to use forms server sided, as well as client sided for validation of the input data. These are just some examples to show the power of Django.

Django builds around the concept of building *apps* which can be made reusable with minimal effort. Therefore a Django project consists of many apps which should reflect just the intended purpose of that app. Another major benefit of Django is the big and great community around Django which constantly release and maintain third party apps. Chances are that if you need a specific feature there exist already an app which is tested and maintained by the community.

Following are the third party apps which were used to build TrAdeCIS:

django-bootstrap3 Django provides a Form class which provides logic for rendering the form in the browser as well as validate the form on the server. This library allows to render these forms in bootstrap3 style.

django-braces Provides a number of reusable mixins, which simplify common scenarios *e.g.*, only showing a page if the user is logged in.

django-debug-toolbar A debug toolbar which shows information to debug while developing.

django-filter Allows to filter a list of results based on defined parameters.

django-formtools Allows to build a form wizard / multi-page form.

django-guardian Provides a system to set per object permission, by default Django only allows setting permission based on classes and not on objects of that classes.

django-groups-manager Hierarchy access structure for permissions. Allows to define a hierarchy of access rights.

3.1.2 Chart.js

Chart.js is a visualization library which has predefined chart types, *e.g.*, bar chart, pie chart, line chart. The dynamic charts and the easy to use syntax allow to create good looking charts with minimal setup. TrAdeCIS uses this framework to visualize the ranking of the alternatives.

3.1.3 Cytoscape.js

Cytoscape.js is graph theory library for analysis and visualization. It allows to dynamically create and delete elements of graphs, as well as many hooks to integrate various interactions with the graph. This library is used to build a model for ANP to simplify visualization as well as interaction with data input for ANP.

3.1.4 Webpack

Webpack is a pack bundler for javascript which allows to modularize required javascript libraries in standardized ways like CommonJs, AMD or ES6. Django provides it's own system for making static files available, but rather than inserting all javascript dependencies with Django, Webpack bundles all files into a bundle which is then served with Django. This solves some common problems which javascript otherwise has, *e.g.*, it does not pollute the global scope. Also is it easier to declare the needed dependencies directly in the javascript file rather than insert multiple global files.

Apart from the already named libraries Chart.js and Cytoscape.js there are a number of other Javascript modules which were used either to implement a specific feature or as dependency of other libraries:

Bootstrap Responsive web design framework as well as default styling.

Bootstrap-multiselect Does improve multiselect elements to look more in a bootstrap way and improves the features.

cytoscape-edgehandles Allows to draw edges in a Cytoscape graph.

Jmat A javascript computation library for linear algebra and statistics.

Jquery Javascript library for Document Object Model (DOM) traversal, manipulation and event handling. Often used as dependency by the frameworks listed here.

Napa Allows node package manager (npm) to clone repositories. (Used for Jmat)

Noty A notification plugin for alert dialogs.

Typeahead.js A typeahead library.

Webpack-bundler-tracker Used in the integration between Django and Webpack.

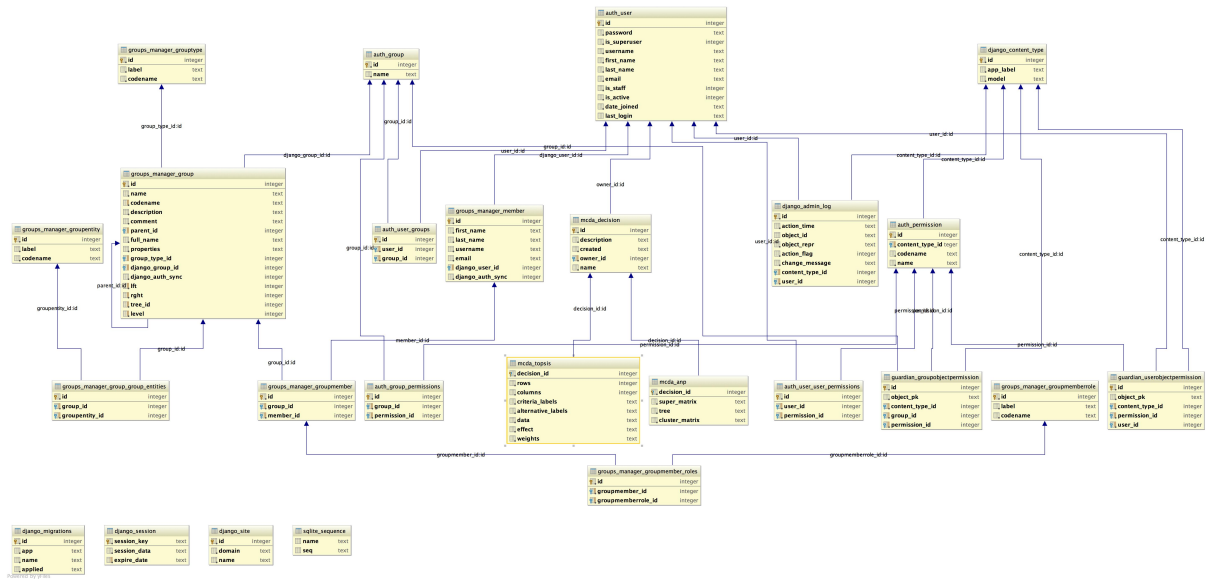
x-editable Library for in place editing of elements in the detail page.

3.2 Architecture Outline

TrAdeCIS is separated into two apps (see Section 3.1.1) with different intentions. The app “account” has been created to handle the permission system and therefore the access rights as well as login and logout of TrAdeCIS. It also manages a basic settings page where depending on the access level of user administration task can be executed. Depending on the access level of the account (user, organization leader, admin) only some parts of the content is visible and editable. For an admin everything is visible and editable. For an organization leader only the data from his own organization is visible and editable. This includes the data from users in this organization, but excludes the data of other organization leaders in the same organization. Lastly users can only see and edit their own data.

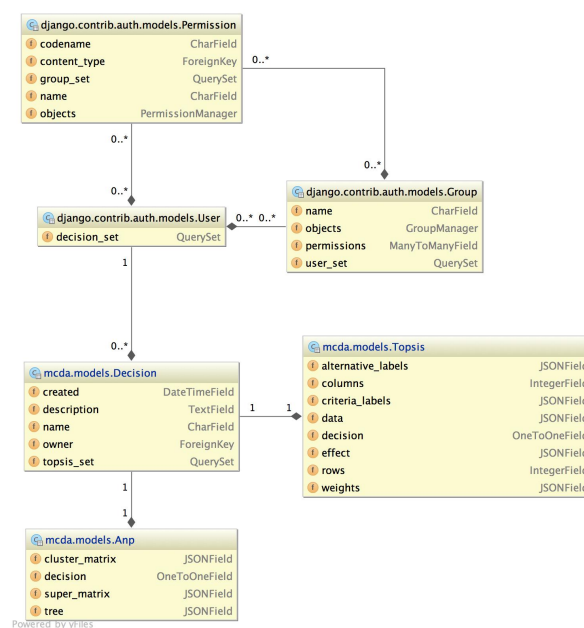
The app “mcda” is the core of the application and manages everything around creating, viewing, updating, or deleting a decision. For each user the decision is persistent in the database as shown in Figure 3.1.

Figure 3.1: Database Model



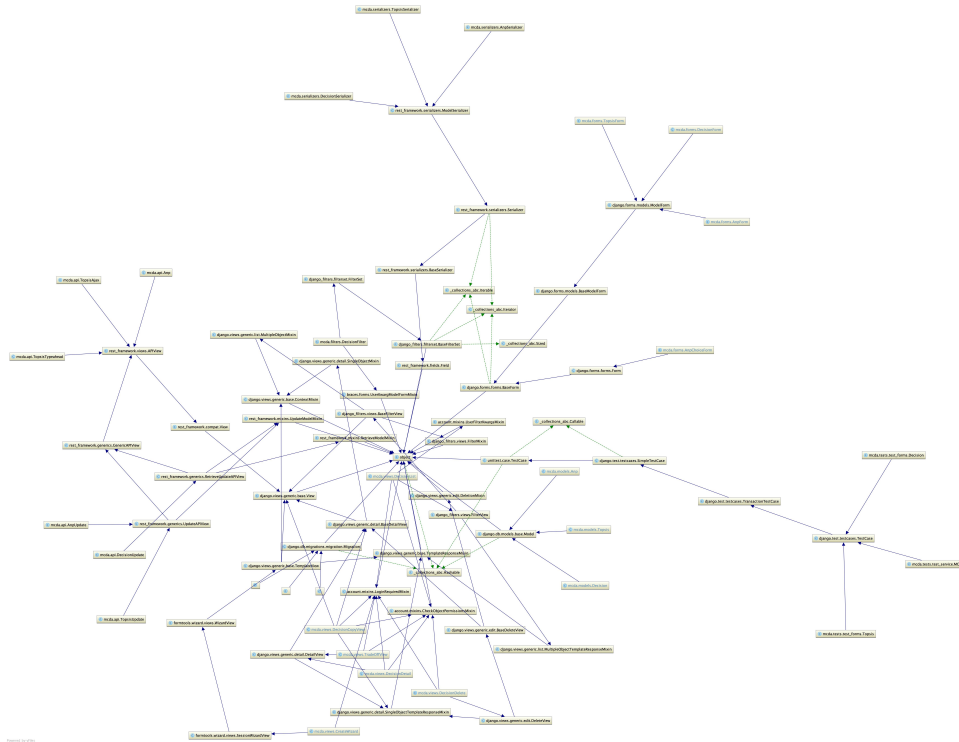
The general information to each entry is stored in the model Matrix and each MCDA algorithm (TOPSIS and ANP) have their own Django model to store their values Figure 3.2.

Figure 3.2: Django Models



An overview over the whole structure of the application is given by Figure 3.3

Figure 3.3: Application Structure



3.3 Architectural Decisions

This project has several features where no standardized way exists to implement this. This partially arises from the needed input from the user for MCDA models TOPSIS and ANP, but as well from how these should be shown so that they are intuitive to the user. Because of that added complexity and that the user experience should be responsive in working with the algorithms their input and visualizations is mostly implemented in Javascript.

3.3.1 TOPSIS

Figure 3.4: Input for TOPSIS

Number of alternatives:

Number of criteria:

Relative ranking of alternatives for each criteria:

Alternatives	Criteria1	Criteria2	Criteria3
A1	1	1	1
A2	1	1	1
A3	1	1	1

Relative priority of criteria:

Relative Priority	1	1	1

Select if the criteria contributes as the benefit or risk to the decision:

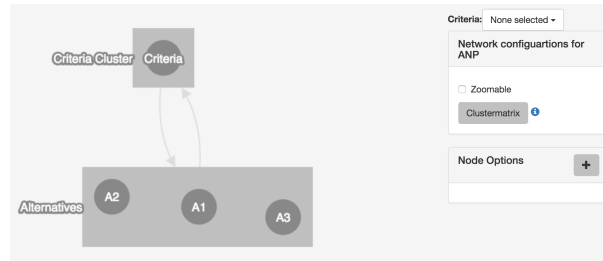
Risk(P)/Benefit(B)	Benefit	Benefit	Benefit

[Previous](#) [Next](#)

TOPSIS expects input in form of a matrix. In HTML there exists no prebuilt way to handle matrix input. Therefore a dynamic table has been implemented which consists of several input or select elements in a table which does grow or shrink depending on the set number of alternatives or number of criteria. On submit these values are parsed from the table into a JSON representation and stored in the backend.

3.3.2 ANP

Figure 3.5: Input for ANP



The model for ANP is by far more complex than the model for TOPSIS as it expects more input as well as constructing the model by the user. As shown in Figure 3.5 the model consists of one cluster “Alternatives”, which is taken from the TOPSIS model and can not be changed and additional clusters which are added by the user with the respected subcriteria. The connection between the clusters is an abstraction and symbolizes in which direction the clusters are connected. Normally each subcriteria should be connected to the subcriteria of an other cluster, but this would result in an unclear model.

Figure 3.6: Selection of a Subcriteria in the ANP Model

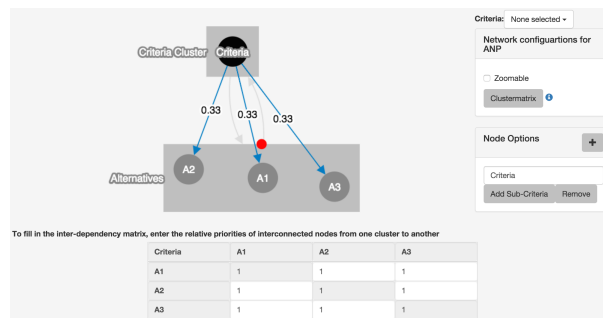
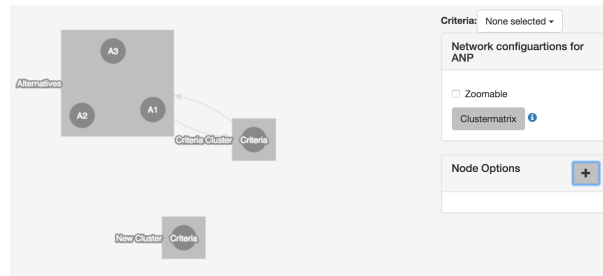


Figure 3.6 shows how this problem is solved: Upon selection of a subcriteria all the connections which are possible in the model are shown as well as the pairwise comparison matrix on bottom (see Section 2.1.1 for more detail on the comparison matrix). To make the valuing more intuitive for the user it also shows absolute values on the arrows.

Upon selection of a cluster or a criteria the node options show fields, where the name can be changed, the node can be deleted or an additional node can be added (in the right of Figure 3.6).

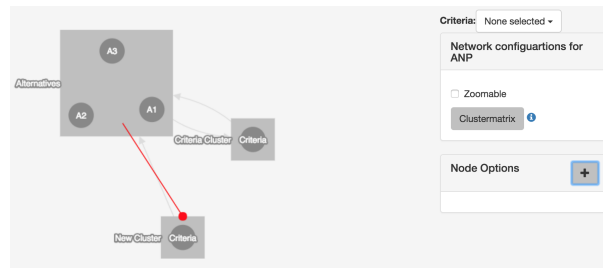
Figure 3.7: Adding an Additional Cluster



An additional cluster is added by pressing the “+” button in the node options (Figure 3.7).

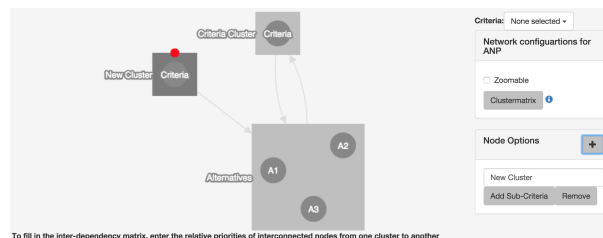
When hovering over a cluster a red dot is shown in the top middle, which can be dragged in order to draw an arrow from the cluster to another selected cluster. All connections between the criteria of the clusters are generated automatically (see Figure 3.8).

Figure 3.8: Drawing Edges (connecting clusters)



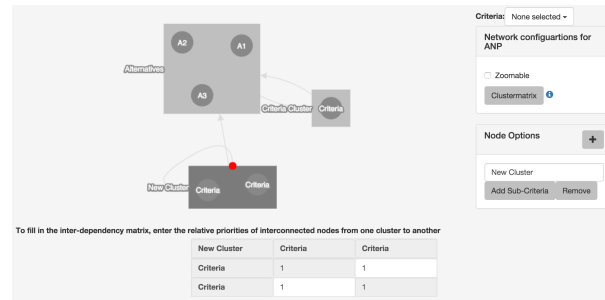
This “new” cluster has the same options as a criteria node when selected (changing the name, adding a subcriteria or deleting). By adding a subcriteria the cluster gains an additional criteria (Figure 3.9). When a criteria is selected and a subcriteria is added, it changes the selected criteria to a subcluster and adds a new criteria.

Figure 3.9: Adding a Subcriteria



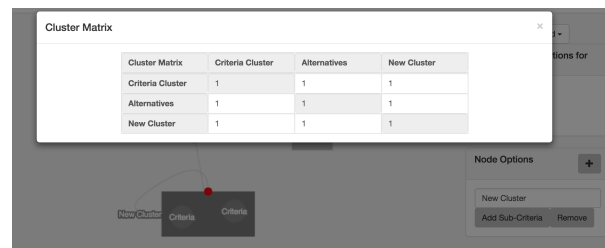
Although it is not an often case the model allows to add additional valuing of the criteria in a cluster. This need arises when a criteria has more influence on the cluster than another. This is done by adding a self loop on the cluster and selecting the cluster, see Figure 3.10.

Figure 3.10: Valuing of the Criteria in a Cluster



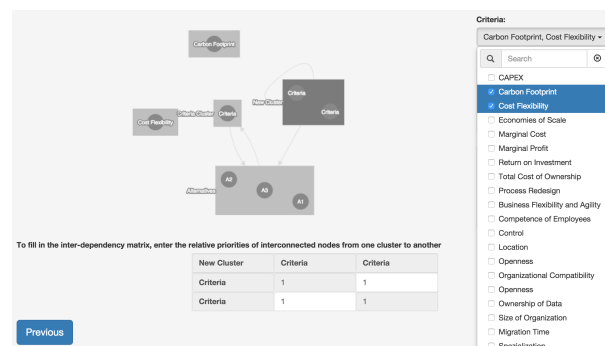
Another possible way to differentiate the result is on the level of clusters. The model allows to value clusters differently and therefore prioritizing a cluster over others, see Figure 3.11.

Figure 3.11: Valuing of the Different Cluster



The choices field gives a selection of possible criteria to support the creation of models in the context of evaluating cloud providers (Figure 3.12). Upon checking a criteria the criteria as well as the cluster are generated. The user should then draw the edges for this cluster.

Figure 3.12: Predefined Criteria



On submit the resulting supermatrix, as well as the whole model will be stored for the user.

3.4 Creation walkthrough

Now after the models where separately introduced the whole workflow of how to create a decision is shown in an example:

We assume that we want to asses three possible IaaS providers to find the optimal one for us. Therefore a “new Decision” is clicked in the navigation bar. The general flow is: (1) General Information page, (2) TOPSIS page, (3) ANP page, (4) decision detail page

- (1) First we enter the name of the decision and an optional description.

Figure 3.13: General Information page

Table 3.1: IaaS Provider Example Technical Requirements

Alternatives	Price	Storage	RAM
Provider A	648	40	2
Provider B	745	50	3.5
Provider C	701	20	4

- (2) Since we have three criteria and three alternatives we set those values. Next we enter the matrix for TOPSIS composed of the different criteria and alternatives and the respective values. The user is expected to weight the criteria by their importance for the user. In our example we value price as highest criteria. Finally we set if the criteria is a risk or a benefit. This basically means if a high value has a positiv effect or a negative effect on the result. In the example price is valued as a risk (a high price is a disadvantage for the result, on the opposite are high storage or ram values positiv).

Figure 3.14: TOPSIS page

The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) is a technique for solving decision problems by determining the relative advantage of available alternatives. Using this method TradeCS obtains a ranked list of available alternative cloud-based solutions, which are evaluated on technical requirements.

Number of alternatives: 3

Number of criteria: 3

Relative ranking of alternatives for each criteria:

Alternatives	Price	Storage	RAM
Provider A	548	40	2
Provider B	740	50	3.5
Provider C	701	20	4

Relative priority of criteria:

Relative Priority: 5 3 2

Select if the criteria contributes as the benefit or risk to the decision:

Risk(Benefit)	Risk	Benefit	Benefit

Previous Next

- (3) Next we create the ANP model by adding an additional subcriteria in the “Criteria” cluster. We rename the two subcriteria to Complexity and Cost Reduction. After that we fill in the values as shown in the following pairwise comparison matrix by clicking on a criteria.

Table 3.2: Pairwise Comparison Matrix for Complexity

Complexity	Provider A	Provider B	Provider C
Provider A	1	0.5	3
Provider B	2	1	2
Provider C	0.33333	0.5	1

Table 3.3: Pairwise Comparison Matrix for Cost Reduction

Cost Reduction	Provider A	Provider B	Provider C
Provider A	1	2	3
Provider B	0.5	1	3
Provider C	0.33333	0.33333	1

Table 3.4: Pairwise Comparison Matrix for Provider A

Provider A	Complexity	Cost Reduction
Complexity	1	1
Cost Reduction	1	1

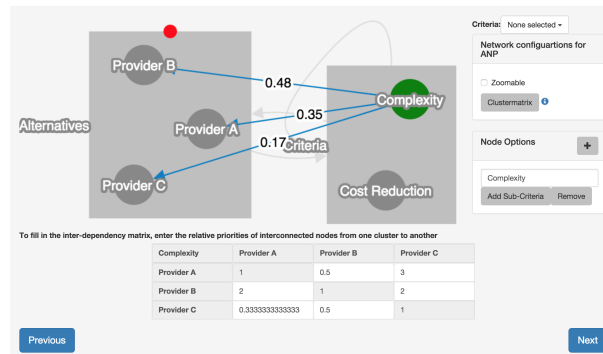
Table 3.5: Pairwise Comparison Matrix for Provider B

Provider B	Complexity	Cost Reduction
Complexity	1	3
Cost Reduction	0.33333	1

Table 3.6: Pairwise Comparison Matrix for Provider C

Provider C	Complexity	Cost Reduction
Complexity	1	2
Cost Reduction	0.5	1

Figure 3.15: ANP Model with Comparison Matrix for Complexity



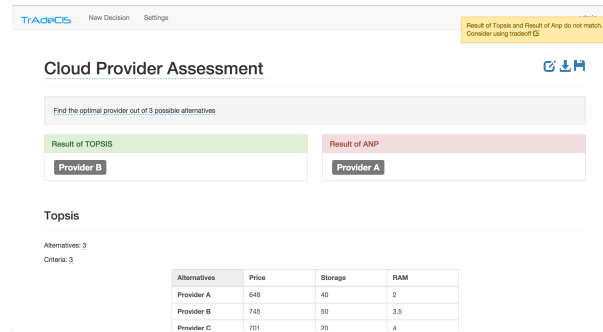
Finally we are also adding a self loop to the cluster “Criteria”. This enables us to value the subcriteria of that cluster differently by clicking on the cluster. In this case we are saying that Complexity is twice as important as Cost Reduction.

Figure 3.16: Valuing of the Cluster Criteria



- (4) By clicking next we are redirected to the decision detail page. We see here our added data as well as the results for TOPSIS and ANP. In this case the result of TOPSIS and ANP did not match, what adds the notification box in the right, which suggests using the tradeoff mode. The tradeoff mode does hide TOPSIS, as the technical requirements should not change and stores changes so that they can be showed later.

Figure 3.17: Detail Page (Overview)



On the detail page the values can be changed and the charts which visualize the results are updated to reflect the changes. These changes are not saved unless the save button is pressed (the disk symbol in the right corner).

Figure 3.18: Detail Page (TOPSIS)

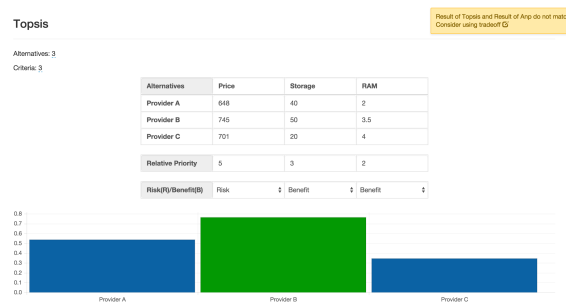
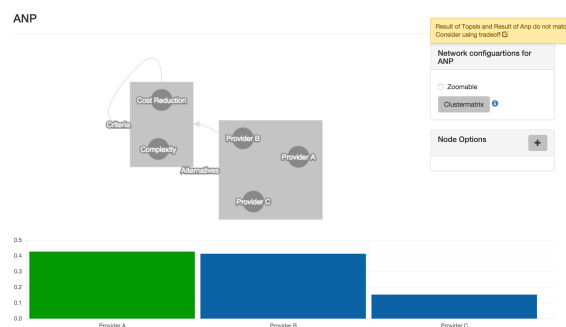


Figure 3.19: Detail Page (ANP)



In our example the result of TOPSIS is “Provider B” and the result of ANP is “Provider A” (see Table 3.7). As we see in the previous image in the ANP model Provider A and Provider B are relatively close. We therefore enter the tradeoff mode and change the cluster priorities so that Complexity is now 3 times as important as Cost Reduction. This results in “Provider B” being now the result of ANP. It also show by how much the values had to be changed in order to achieve this result. If we did the changes without the tradeoff mode this text would not be shown.

Table 3.7: Result of TOPSIS and ANP

	Provider A	Provider B	Provider C
TOPSIS	0.5401	0.7672	0.3486
ANP	0.4291	0.4155	0.1554

Figure 3.20: Result after the tradeoff



It lays in the decision of the user if he wants to use the tradeoff mode to make changes or if he wants to do them without it. The implications are that in the tradeoff mode TOPSIS is hidden (the technical requirements should be static) and changes can be made in ANP. This compromise in the business value is recorded and shown (see Figure 3.20).

3.5 Implementation Details

In this sections the implementations for the two main algorithms TOPSIS and ANP are shown. In both cases numpy is used to compute the results.

The implementation for TOPSIS follows directly the shown algorithm in Section 2.1.7:

1. First the weights, as well as the matrix are normalized in order to make all values comparable regardless of the initial scale and then multiplied in order to create the weighted matrix.
2. Next the minimum arrays and the maximum arrays are created, which hold the maximum values respectively the minimum values for each criteria.
3. After that the ideal solution and the anti ideal solution are computed with the help of the “has positive effect” array, which consists of true and false values for each criteria. The ideal solution has the maximum value if it has a positive effect and the minimum if the criteria has a negative effect. The opposite holds true for the anti ideal solution.

4. With that the euclidean distance can now be computed from the weighted matrix and the ideal solution, as well as from the weighted matrix and the anti ideal solution.
5. In the final step the relative closeness is computed, which is the measure which result is the best. The best result is nearest to the optimal solution and farthest away from the anti ideal solution.

The complexity of the TOPSIS algorithm is $O(m * n^2)$, where m is the number of alternatives and n the number of criteria

```

1  def topsis(matrix, weights, has_positiv_effect, normalization=
    vector_normalization):
2
3      # normalize and apply weights
4      weighted_matrix = normalization(weights) * normalization(matrix)
5
6      # extract min and max values for each column
7      mins = numpy.min(weighted_matrix, axis=0)
8      maxs = numpy.max(weighted_matrix, axis=0)
9
10     # create ideal and anti ideal arrays
11     ideal = numpy.where(has_positiv_effect, maxs, mins)
12     anti_ideal = numpy.where(has_positiv_effect, mins, maxs)
13
14     # calculate distances to the ideal and anti ideal arrays
15     distance_ideal = norm(weighted_matrix - ideal, axis=1)
16     distance_anti_ideal = norm(weighted_matrix - anti_ideal, axis=1)
17
18     # compute relative closeness
19     relative_closeness = distance_anti_ideal / (distance_ideal +
        distance_anti_ideal)
20
21     return relative_closeness

```

Listing 3.1: Algorithm for TOPSIS

The algorithm for ANP is divided into two major parts which were already introduced in Section 2.1.2. The creation of the supermatrix is handled by the frontend, because otherwise numerous pairwise comparison matrix would have been sent to the backend.

Frontend creation of supermatrix (Javascript):

```

1  supermatrix: function (clusterNodes) {
2      var children = graph.findChildren(clusterNodes);
3      var matrix = utils.matrix(children.length, 0);
4
5      children.each(function (column, sourceNode) {
6          children.each(function (row, targetNode) {
7              matrix[row][column] = graph.getValue(sourceNode, targetNode)
                || 0;
8          });
9      });
10
11     return matrix
12 }

```

Listing 3.2: Generation of the Supermatrix

The supermatrix is a $N \times N$ matrix, where N is the number of total criteria (including the alternatives). The supermatrix is constructed by iterating over the children of all cluster nodes and adding the value which is composed of the following: (1) the absolute value which were shown on an edge from a source node to a target node (the value which was computed from the pairwise comparison matrix), (2) valuing of the criteria in a cluster (if in a cluster criteria valued more or less than other criteria in the same cluster), (3) the valuing of the clusters (if a cluster is higher valued than another cluster). Important to note is that by design the alternatives are always the last values in rows as well as columns.

Backend computation of the result (Python):

```

1  def anp(super_matrix, number_alternatives):
2
3      limit_m = limit_matrix(super_matrix)
4
5      x, y = limit_m.shape
6      start = x-number_alternatives
7      end = x
8
9      return [limit_m[i][0] for i in range(start, end)]

```

Listing 3.3: Algorithm for ANP

In order to compute the result for ANP the supermatrix as well as the number of alternatives have to be provided. The limit matrix is computed then with the dimension of the matrix and the number of alternatives the final result can be created. Important to note by design the alternatives are the last values in matrix. By knowing the number of result the values can be extracted from the computation of the limit matrix.

```

1  def limit_matrix(matrix):
2      result = matrix
3      previous_matrix = result
4
5      while True:
6          result = linear_normalization(numpy.linalg.matrix_power(result
7              , 3))
8
9          if numpy.isnan(numpy.sum(result)):
10             raise ArithmeticError('received not a number')
11
12         if numpy.allclose(previous_matrix, result):
13             break
14
15         previous_matrix = result
16
17     return result

```

Listing 3.4: Algorithm for the Limitmatrix

The limit matrix is computed by repeatedly raising the supermatrix by the power of 3 (see Section 2.1.2 for explanation) and checking if the difference of the previous matrix and the current lays in a tolerable range, otherwise the process is continued. This process usually need around three to five iterations to find a result. The complexity of the algorithm is $O(n^{3^x})$, where x is the number of iterations and n the dimension of the matrix.

Chapter 4

Stress / Performance Testing and Evaluation

4.1 Algorithm tests

This section will show all intermediate steps in the process of the computation of both algorithms TOPSIS and ANP by using the values of other publications as input the result can also be assessed in terms of the rightfulness.

For the example of **TOPSIS** the input data is taken from [18]:

Table 4.1: Weights from example [18]

Criteria	C1	C2	C3	C4	C5	C6
Weights	0.3	0.2	0.1	0.1	0.14	0.16

Table 4.2: TOPSIS example [18]

Supplier	C1	C2	C3	C4	C5	C6
1	500	1000	50	75	55	700
2	600	1200	40	50	70	800
3	400	900	30	60	50	600
4	700	1400	75	80	90	1000
5	100	400	20	25	60	250
6	800	1200	30	70	95	900
7	300	1100	35	85	88	750
8	225	550	45	65	85	400
9	115	300	15	26	20	225
10	350	800	25	68	78	575

After the first step the normalized matrix is obtained which is the same as in [18].

Table 4.3: Normalized matrix

Supplier	C1	C2	C3	C4	C5	C6
1	0.3381506	0.33236534	0.39497625	0.37313895	0.23980456	0.33122761
2	0.40578072	0.39883841	0.315981	0.2487593	0.3052058	0.37854584
3	0.27052048	0.29912881	0.23698575	0.29851116	0.21800414	0.28390938
4	0.47341084	0.46531148	0.59246438	0.39801488	0.39240746	0.47318229
5	0.06763012	0.13294614	0.1579905	0.12437965	0.26160497	0.11829557
6	0.54104096	0.39883841	0.23698575	0.34826302	0.41420787	0.42586406
7	0.20289036	0.36560188	0.27648338	0.42289081	0.38368729	0.35488672
8	0.15216777	0.18280094	0.35547863	0.32338709	0.37060704	0.18927292
9	0.07777464	0.0997096	0.11849288	0.12935483	0.08720166	0.10646602
10	0.23670542	0.26589228	0.19748813	0.33831264	0.34008646	0.27207982

The weighted matrix does not give the same result as in the original, because in this implementation the weights are also normalized with vector normalization.

Table 4.4: Weighted and normalized matrix

Supplier	C1	C2	C3	C4	C5	C6
1	0.22961038	0.15045472	0.08939867	0.08445603	0.0759881	0.11995175
2	0.27553245	0.18054567	0.07151894	0.05630402	0.09671212	0.13708772
3	0.1836883	0.13540925	0.0536392	0.06756483	0.06908009	0.10281579
4	0.32145453	0.21063661	0.13409801	0.09008643	0.12434416	0.17135965
5	0.04592208	0.06018189	0.03575947	0.02815201	0.0828961	0.04283991
6	0.3673766	0.18054567	0.0536392	0.07882563	0.13125217	0.15422368
7	0.13776623	0.16550019	0.06257907	0.09571684	0.12158095	0.12851974
8	0.10332467	0.0827501	0.08045881	0.07319523	0.11743615	0.06854386
9	0.05281039	0.04513642	0.0268196	0.02927809	0.02763203	0.03855592
10	0.16072726	0.12036378	0.04469934	0.07657347	0.10776494	0.0985318

Likewise before the ideal solution and anti ideal solution are not the same as in the original for the same reason.

The ideal solution $A^+ = [0.3673766, 0.21063661, 0.13409801, 0.09571684, 0.13125217, 0.17135965]$

The anti ideal solution $A^- = [0.04592208, 0.04513642, 0.0268196, 0.02815201, 0.02763203, 0.03855592]$

Table 4.5: Distanz for each supplier

(a) Distanz to the ideal solution		(b) Distanz to the anti ideal solution	
Supplier	Distanz to the ideal solution	Supplier	Distanz to the anti ideal solution
1	0.17442224	1	0.24674523
2	0.1310645	2	0.29719981
3	0.23501134	3	0.18774624
4	0.04677883	4	0.38163207
5	0.3988242	5	0.05812708
6	0.08920784	6	0.38609836
7	0.24860002	7	0.21374382
8	0.31658291	8	0.13630832
9	0.41308467	9	0.00697975
10	0.25508287	10	0.17754463

The final result is the same as in the original. Then minor discrepancy can be explained by that the original uses rounded values in the intermediate steps. [0.58586014, 0.69396353, 0.44409905, 0.8908085 , 0.1272063 , 0.81231501, 0.46230489, 0.30097363, 0.0166159 , 0.41038684]

For **ANP** the input is taken from [3]:

Table 4.6: Pairwise Comparison Matrix for AD-HOT

AD-HOT	ANTALYA	ISTANBUL	IZMIR
ANTALYA	1	400/75	400/150
ISTANBUL	75/400	1	75/150
IZMIR	150/400	150/75	1

Table 4.7: Pairwise Comparison Matrix for AV-HOS

AV-HOS	ANTALYA	ISTANBUL	IZMIR
ANTALYA	1	26/192	26/49
ISTANBUL	192/26	1	192/49
IZMIR	49/26	49/192	1

Table 4.8: Pairwise Comparison Matrix for WELL-KNOWN

WELL-KNOWN	ANTALYA	ISTANBUL	IZMIR
ANTALYA	1	334/1955	334/338
ISTANBUL	1955/334	1	1955/338
IZMIR	338/334	338/1955	1

Table 4.9: Pairwise Comparison Matrix for ANTALYA

ANTALYA	AD-HOT	AV-HOS	WELL-KNOWN
AD-HOT	1	1/5	1/3
AV-HOS	5	1	2
WELL-KNOWN	3	1/2	1

Table 4.10: Pairwise Comparison Matrix for ISTANBUL

ISTANBUL	AD-HOT	AV-HOS	WELL-KNOWN
AD-HOT	1	2	4
AV-HOS	1/2	1	1
WELL-KNOWN	1/4	1	1

Table 4.11: Pairwise Comparison Matrix for IZMIR

IZMIR	AD-HOT	AV-HOS	WELL-KNOWN
AD-HOT	1	1/3	1/2
AV-HOS	3	1	2
WELL-KNOWN	2	1/2	1

The resulting supermatrix differs from the original. In this case the reason is that the valuing for the clusters (clustermatrix) as well as for the valuing of criteria in a cluster is also included. In this example since no cluster is valued higher than another all clusters are multiplied by 0.333333 (explaining the lower values).

Table 4.12: Resulting Supermatrix

	AD-HOT	AV-HOS	WELL-KNOWN	ANTALYA	ISTANBUL	IZMIR
AD-HOT	0	0	0	0.036111	0.194444	0.055555
AV-HOS	0	0	0	0.194444	0.077777	0.180555
WELL-KNOWN	0	0	0	0.102777	0.061111	0.099999
ANTALYA	0.213888	0.033333	0.041666	0	0	0
ISTANBUL	0.038888	0.238888	0.247222	0	0	0
IZMIR	0.080555	0.061111	0.041666	0	0	0

The resulting limitmatrix after 4 iterations shows clearly that the results correlate.

Table 4.13: Resulting Limitmatrix

	AD-HOT	AV-HOS	WELL-KNOWN	ANTALYA	ISTANBUL	IZMIR
AD-HOT	0	0	0	0.360024	0.360024	0.360024
AV-HOS	0	0	0	0.397075	0.397075	0.397075
WELL-KNOWN	0	0	0	0.242899	0.242899	0.242899
ANTALYA	0.301836	0.301836	0.301836	0	0	0
ISTANBUL	0.507593	0.507593	0.507593	0	0	0
IZMIR	0.190570	0.190570	0.190570	0	0	0

4.2 Load up Time

This section analyses the performance of TrAdeCIS with respect to how long certain tasks need to execute or to load. While TOPSIS scales well even with growing number of alternatives and criteria, ANP does not. Not only the more demanding algorithm, but also the more complex model and the additional input does limit the size at which it is user friendly to work with it.

The Table 4.14 shows an overview of the performance of the ANP model with growing number of nodes. The **load up time** shows how long the initial rendering takes if a certain amount of nodes are present. The **selection time and matrix generation** shows how long the generation of the pairwise comparison matrix, as well as the highlighting in the model takes (this refers to the problem shown in Figure 3.6). Note that the value for selection time is the number of nodes in a cluster, while for load up time it is the total number of nodes.

Table 4.14: ANP Time Measurements

Number of nodes	load up time in ANP	selection time and matrix generation
3	414 ms	100 ms
10	486 ms	250 ms
25	637 ms	600 ms
50	826 ms	1400 ms
100	1256 ms	4100 ms

With respect to load up time of the pages the detail page of a Decision is the one which needs the longest to load (shows TOPSIS as well as ANP model). The Figure 4.1 and Figure 4.2 show how long the page needed to fully load for one example with 10 nodes in ANP and the other with 50 nodes in ANP. The initial page load does not take long however until all DOM elements which require JQuery are loaded it takes around 1 second. After that time two requests are sent for the computation of the result for TOPSIS and ANP.

Figure 4.1: Load up Time Detail Page for 10 nodes

Name	Method	Status	Type	Initiator	Size	Time	Timeline - Start Time	1.00 s	1.50 s	▲
2/	GET	200	document	Other	4.4 KB	123 ms				
vendor.css	GET	200	stylesheet	(index):11	(from cache)	0 ms				
decision-detail.css	GET	200	stylesheet	(index):13	(from cache)	0 ms				
table.css	GET	200	stylesheet	(index):17	(from cache)	0 ms				
style.css	GET	200	stylesheet	(index):16	(from cache)	0 ms				
logo.png	GET	200	png	(index):29	(from cache)	0 ms				
vendor.js	GET	200	script	(index):115	(from cache)	0 ms				
decision-detail.js	GET	200	script	(index):210	(from cache)	0 ms				
data:application/oc...	GET	200	font	(index):208	(from cache)	0 ms				
topsis/	POST	200	xhr	vendor.js:26	266 B	85 ms				
anp/	POST	200	xhr	vendor.js:26	276 B	79 ms				

From the second load up time we see that the computation of the result for ANP did take longer with 50 nodes, but it is still in a tolerable amount of time.

Figure 4.2: Load up Time Detail Page for 50 nodes

Name	Method	Status	Type	Initiator	Size	Time	Timeline - Start Time	1.50 s	2.00 s	▲
3/	GET	200	document	Other	8.3 KB	160 ms				
vendor.css	GET	200	stylesheet	(index):11	(from cache)	0 ms				
decision-detail.css	GET	200	stylesheet	(index):13	(from cache)	0 ms				
table.css	GET	200	stylesheet	(index):17	(from cache)	0 ms				
style.css	GET	200	stylesheet	(index):16	(from cache)	0 ms				
logo.png	GET	200	png	(index):29	(from cache)	0 ms				
vendor.js	GET	200	script	(index):115	(from cache)	0 ms				
decision-detail.js	GET	200	script	(index):210	(from cache)	212 ms				
data:application/oc...	GET	200	font	(index):136	(from cache)	0 ms				
topsis/	POST	200	xhr	vendor.js:26	295 B	86 ms				
anp/	POST	200	xhr	vendor.js:26	295 B	120 ms				

The overall load up time of the whole page is around 1.5 seconds and 3 seconds depending on the complexity of the model and the number of the nodes.

Chapter 5

Summary and Conclusions

Like shown Section 3.3 and Section 4 TrAdeCIS has achieved the planned features in terms of providing a framework which uses TOPSIS and ANP combined to gain one result, which can be used to evaluate different multi-criteria problems. The overall performance is in a tolerable range, but can certainly be improved.

The model which were introduced in Section 3.3 to abstract the algorithms and make them more visual, as well as simplify the input for the algorithms need further discussion. While they achieve to simplify the process for several criteria and alternatives, the complexity rises the higher the number gets. Especially in terms of screen width there is a limit in size after which the tables no longer fit into the screen. With respect to ANP the number of necessary comparison matrices, as well as the initial complexity to produce the model are also correlated to the number of criteria and alternatives. This is the reason why the input for alternatives was limited to 30.

The algorithms have a major discrepancy in terms of performance. While TOPSIS does scale well with high numbers and the necessary input is not that numerous, ANP does badly in both.

The implemented prototype for this thesis is an initial working product which can be refined to further improve the proposed model by TrAdeCIS.

These are the major points which could be improved in a further work:

(a) Model improvements

- The ANP model could be improved with respect to performance in rendering. However there is certainly a point after which it does not make sense to improve ANP further to support more nodes, for the previously mentioned reasons.
- Consistency checks for ANP. Currently the input in the pairwise comparison matrix are not checked in terms if they are consistent, e.g., A is bigger than B and C is bigger than B, logically C would also be bigger than A, but this could be entered otherwise. Therefore an error message could be shown which informs the user about the inconsistency.

- Tradeoff suggestions. Tradeoff is currently done manually. A possible implementation could give suggestions which change would make the biggest impact for the tradeoff, what would limit the effort for the user.

(b) Webpage improvements

- As Section 4.2 showed JQuery seems to be a major slow down. DOM manipulations have their cost and therefore should be limited, however some manipulations in this case are necessary, e.g. the pairwise comparison matrix can have different sizes depending on the selected criteria. A possible solution could be a less heavy framework for the manipulations.
- Single Page Application (SPA). The performance could be improved by implementing TrAdeCIS as SPA, this would lower the request size as only on the initial load static files would have to be downloaded (if not cached).

Bibliography

- [1] *Advances in Decision Analysis*. Mathematical Modelling: Theory and Applications. Springer, 1999.
- [2] Shingo Aoki et al. “Data Envelopment Analysis for a Supply Chain”. In: *Artificial Life and Robotics* 15.2 (2010).
- [3] *The Analytic Hierarchy and Analytic Network Processes* 32 (2003).
- [4] Eddie W. L. Cheng and Heng Li. “Contractor Selection Using the Analytic Network Process”. In: *Construction Management and Economics* 22.10 (2004).
- [5] BANA Consulting. *M-MACBETH*. URL: <http://www.m-macbeth.com/>.
- [6] S.C. Deshmukh. “Preference Ranking Organization Method Of Enrichment Evaluation (Promethee)”. In: *International Journal of Engineering Science Invention* 2.11 (2013).
- [7] Creative Decisions Foundation. *Super Decisions*. URL: <http://www.superdecisions.com/>.
- [8] Radhika Garg and Burkhard Stiller. *Trade-off-Based Adoption Methodology for Cloud-Based Infrastructures and Services*. Tech. rep. Department of Informatics IFI, University of Zurich, Switzerland, 2014.
- [9] Alessio Ishizaka and Philippe Nemery. *Multi-criteria Decision Analysis: Methods and Software*. Wiley, 2013.
- [10] Laura Meade and Joseph Sarkis. *Strategic Analysis of Logistics and Supply Chain Management Systems using the Analytical Network Process*. Tech. rep. Pergamon, 1998.
- [11] *Multiple Criteria Decision Analysis: State of the Art Surveys*. International Series in Operations Research and Management Science. Springer, 2005.
- [12] Komaragiri Srinivasa Raju and A. Vasan. “Multi attribute utility theory for irrigation system evaluation”. In: *Water Resources Management* 21.4 (2007).
- [13] Thomas L. Saaty. “Decision Making - the Analytic Hierarchy and Network Processes (AHP/ANP)”. In: *Journal of Systems Science and Systems Engineering* 13.1 (2004), pp. 1–35.
- [14] Thomas L. Saaty. “Fundamentals of the Analytic Network Process - Dependence and Feedback in Decision-Making with a Single Network”. In: *Journal of Systems Science and Systems Engineering* 13.2 (2004), pp. 129–157.
- [15] Thomas L. Saaty. “The Analytic Hierarchy and Analytic Network Measurement Processes: Applications to Decisions under Risk”. In: *European Journal of Pure and Applied Mathematics* 1.1 (2008).
- [16] Thomas L. Saaty and Luis G. Vargas. *Decision Making with the Analytic Network Process*. Second. Springer, 2013.

- [17] Lee-Ing Tong, Chung-Ho Wang, and Hung-Cheng Chen. “Optimization of Multiple Responses Using Principal Component Analysis and Technique for Order Preference by Similarity to Ideal Solution”. In: *The International Journal of Advanced Manufacturing Technology* 27.3 (2005).
- [18] *Application of Topsis Method for Supplier Selection in Manufacturing Industry* (2012).

List of Figures

2.1	Linear Hierarchy of AHP [9]	6
2.2	Five-Cluster ANP Network [9]	7
2.3	Geometrical Representation of TOPSIS	11
3.1	Database Model	18
3.2	Django Models	18
3.3	Application Structure	19
3.4	Input for TOPSIS	19
3.5	Input for ANP	20
3.6	Selection of a Subcriteria in the ANP Model	20
3.7	Adding an Additional Cluster	21
3.8	Drawing Edges (connecting clusters)	21
3.9	Adding a Subcriteria	21
3.10	Valuing of the Criteria in a Cluster	22
3.11	Valuing of the Different Cluster	22
3.12	Predefined Criteria	22
3.13	General Information page	23
3.14	TOPSIS page	24
3.15	ANP Model with Comparison Matrix for Complexity	25
3.16	Valuing of the Cluster Criteria	25
3.17	Detail Page (Overview)	26
3.18	Detail Page (TOPSIS)	26

3.19 Detail Page (ANP) 26

3.20 Result after the tradeoff 27

4.1 Load up Time Detail Page for 10 nodes 36

4.2 Load up Time Detail Page for 50 nodes 36

List of Tables

2.1	MCDA Problems and Methods [9]	5
2.2	Pairwise Comparison Scale [9]	6
2.3	Example Supermatrix	8
2.4	Resulting Limitmatrix by Power of 2	8
2.5	Resulting Limitmatrix by Power of 3	8
2.6	Different ELECTRE Methods [9]	10
2.7	Required Inputs for MCDA Ranking or Choice Method [9]	13
3.1	IaaS Provider Example Technical Requirements	23
3.2	Pairwise Comparison Matrix for Complexity	24
3.3	Pairwise Comparison Matrix for Cost Reduction	24
3.4	Pairwise Comparison Matrix for Provider A	24
3.5	Pairwise Comparison Matrix for Provider B	24
3.6	Pairwise Comparison Matrix for Provider C	25
3.7	Result of TOPSIS and ANP	27
4.1	Weights from example [18]	31
4.2	TOPSIS example [18]	31
4.3	Normalized matrix	32
4.4	Weighted and normalized matrix	32
4.5	Distanz for each supplier	33
4.6	Pairwise Comparison Matrix for AD-HOT	33

4.7	Pairwise Comparison Matrix for AV-HOS	33
4.8	Pairwise Comparison Matrix for WELL-KNOWN	33
4.9	Pairwise Comparison Matrix for ANTALYA	34
4.10	Pairwise Comparison Matrix for ISTANBUL	34
4.11	Pairwise Comparison Matrix for IZMIR	34
4.12	Resulting Supermatrix	34
4.13	Resulting Limitmatrix	35
4.14	ANP Time Measurements	35

Appendix A

Installation Guidelines

A.1 Development

Necessary: Python3.4 and pip, node.js and npm

1. Create a new virtualenv with python 3.4.3
2. Activate the new virtualenv
3. `cd TrAdeCIS_Project`
4. Install all Python dependencies for development:
`pip install -r requirements/local.txt`
5. Install all Javascript dependencies:
`npm install`
6. after changes in js / css rebuild the webpack bundles by running
`./node_modules/bin/webpack`
7. `cd TrAdeCIS`
8. Migrate the database:
`./manage.py migrate`

A.2 Deployment

Necessary: Python3.4 and pip, node.js and npm, postgresql, nginx

Prebuild production ready webpack bundles: `./node_modules/bin/webpack -p`

On the server:

1. Create a new virtualenv with python 3.4.3
2. Activate the new virtualenv
3. `cd TradeCIS_Project`
4. Install all Python dependencies for deployment:
`pip install -r requirements/production.txt`
5. Install all Javascript dependencies: `npm install`
6. `cd TradeCIS`
7. Migrate the database:
`./manage.py migrate --settings=config.settings.production`
8. collect static files:
`./manage.py collectstatic --settings=config.settings.production`
9. Create an auto start script for `gunicorn_start.sh`

Appendix B

Contents of the CD

1. Source files of the thesis
2. Source files of the developed prototype for TrAdeCIS (refere to Appendix A for how to install it)