Masters Program in Geospatial Technologies

THE USE OF ANALYTICAL HIERARCHY PROCESS IN SPATIAL DECISION SUPPORT SYSTEM FOR LAND USE MANAGEMENT

A case study of the Zambezi river valley in Mozambique

Miguel Antão de Praetere Carrilho

Dissertation submitted in partial fulfilment of the requirements for the Degree of *Master of Science in Geospatial Technologies*

NOVA IMS Information Management School
Universitat Jaume I
Westfälische Wilhelms-Universität Münster

THE USE OF ANALYTICAL HIERARCHY PROCESS IN SPATIAL DECISION SUPPORT SYSTEM FOR LAND USE MANAGEMENT

A case study of the Zambezi river valley in Mozambique.

Miguel Antão Carrilho

mcarrilho@isegi.unl.pt

NOVA Information Management School,

Universidade Nova de Lisboa,

Lisbon, Portugal.

Supervised by

Marco Painho, PhD

Professor, NOVA Information Management School,

Universidade Nova de Lisboa,

Lisbon, Portugal.

Co-Supervised by

Óscar Belmonte Fernández, PhD

Professor, Universitat Jaume I

Castellon, Spain.

Joel Dinis Silva, MSc

NOVA Information Management School,

Universidade Nova de Lisboa,

Lisbon, Portugal.

February 2015

ACKOWLEDGEMENTS

I would like to acknowledge that, Joel Dinis my co-supervisor, was a restless support on the development of this thesis. Joel Dinis is one of the brightest minds I had the joy of meeting, and with his guidance and shared knowledge I was able to finish the work here presented.

I must also recognize the constant supervising, and motivational spirit of Professor Marco Painho, contributing to a constant improvement of the work presented. This work had also the tremendous support of my friends and colleges Tiago Oliveira, Alexandre Baptista, Luis Almeida, Otávio Sian, Hugo Martins and Bezaye Tesfaye.

I would also like to acknowledge that, without my family, my mother Manuela Antão, brother, Pedro Carrilho and father Orlando Carrilho I would not have a motivation to fulfill this work. I would also like to recognize the great support and dedication of Vanessa Ligeiro, as my companion and best friend. I would like to thank also my friends and colleges of this master, for the support during this year and half of hard work. I would like to dedicate this work to my friend Gonçalo Amor, thus he remembers everyday what he fights for.

“I believe that the very purpose of our life is to seek happiness.” – Tenzin Gyatso (1998), *The art of Happiness A handbook for living*.

THE USE OF ANALYTICAL HIERARCHY PROCESS IN SPATIAL DECISION SUPPORT SYSTEM FOR LAND USE MANAGEMENT

A case study of the Zambezi river valley in Mozambique.

ABSTRACT

Geographic information systems give us the possibility to analyze, produce, and edit geographic information. Furthermore, these systems fall short on the analysis and support of complex spatial problems. Therefore, when a spatial problem, like land use management, requires a multi-criteria perspective, multi-criteria decision analysis is placed into spatial decision support systems. The analytic hierarchy process is one of many multi-criteria decision analysis methods that can be used to support these complex problems. Using its capabilities we try to develop a spatial decision support system, to help land use management. Land use management can undertake a broad spectrum of spatial decision problems. The developed decision support system had to accept as input, various formats and types of data, raster or vector format, and the vector could be polygon line or point type. The support system was designed to perform its analysis for the Zambezi river Valley in Mozambique, the study area. The possible solutions for the emerging problems had to cover the entire region. This required the system to process large sets of data, and constantly adjust to new problems' needs. The developed decision support system, is able to process thousands of alternatives using the analytical hierarchy process, and produce an output suitability map for the problems faced.

KEY WORDS

Analytical Hierarchy Process

Geographic information systems

Spatial Decision Support Systems

Logarithmic Least Squares Method

ACRONYMS

AHP	- Analytic Hierarchy Process
CR	- Consistency Ratio
CI	- Consistency Index
IR	- Index of Randomness
GDAL	- Geospatial Data Abstraction Library
GI	- Geographic Information
GIS	- Geographic Information System
LLSM	- Logarithmic Least Squares Method
MCDA	- Multi-criteria Decision Analysis
PBO	- Purpose-built offices
PROMETHEE	- Preference Ranking Organization Method for Enrichment Evaluations
SDSS	- Spatial Decision Support System
VNS	- Variable Neighborhood Search

TABLE OF CONTENTS

INDEX OF TABLES

INDEX OF FIGURES

1. INTRODUCTION

Geographic Information Systems (GIS) are used to manipulate, summarize, query, edit and visualize geographic information (Goodchild, 1992). Geographic information science was established because geographic information has unique properties and problems (Goodchild, 1992). Some individuals such as Ratti (2005) tend to see geographic information science as a modern approach to Geography. Longley *et al* (2010) argue that geographic information science was born in London, due to a cholera outbreak studied by John Snow in 1854. John Snow related the location of water wells to cholera outbreaks. The geographic perspective of problems is sometimes the key to the solution, like the cholera outbreak of 1854 in London.

GIS started to be used in the 1960's (Goodchild, 1993). The first GIS was developed in Canada for land inventory (Tomlinson, 1962), due to the vast measurements needed for the project. The only cost-efficient alternative was a computer based system (Goodchild, 1993). The role of GIS in many facilities' management systems provide a different perspective to data (Goodchild, 1992). Accessing different data types at the same time is another problem that stands for the use of GIS (Goodchild, 1993). The GIS are used for various purposes, such as land use management (Malczewski, 2006).

When it comes to land use planning, projects like the location of a future power plant or the reallocation of a city are a reality. These decisions call for several perspectives, because they affect third-party interests. This type of problems tend to involve several organizations, governmental, environmental and private corporations to make the final decision on the location. These decisions are made to achieve medium or long-term goals and need to be based on concrete evidence. Thus lots of spatial data is needed. For example, hydrology data, soil type data, soil price data, connections to industry supply chains like ports, main power lines, etc. Hence these type of decisions depend on a large number of criteria. For these reasons, land use management is a very complex task, justifying the need for the integration of a Multi-criteria Decision Analysis (MCDA) as a component of a SDSS (Marinoni, 2004). The use of MCDA in GIS has been widely studied (Lin et al., 2014; Malczewski, 2006a; Marinoni, 2004; Massei, Rocchi, & Paolotti, 2013;

Sugumaran & DeGroote, 2011). Decisions are the result of comparing alternatives over one or more criteria, that are relevant for the problem (Marinoni, 2004). They are results of an instinctive weighting process. Therefore the MCDA used, had to be intuitive in the decision thinking, requiring weight assigning, and an intuitive structure of the decision. The MCDA method found to cover these points was the Analytic Hierarchy Process (AHP) (Saaty, 1990).

The application software was developed to be part of the environmental strategic evaluation of the multi-sectorial plan for land use management of the Zambezi river Valley. This project has the final goal of helping land use planners make better evidence based decisions. This goal is assisted by a web-based GIS of the study area, part of the Zambezi river basin in Mozambique. To complement the decision making process the web-based GIS needed to be assisted by a decision support system. Since decisions are primarily spatial, a Spatial Decision Support System (SDSS) is the most suitable tool for these decision makers (Malczewski, 2006). Since the web-based GIS was planned to support all types of data, from vector type to raster type, and from all possible topics, from geology maps to population density maps, the SDSS had to support a broad spatial analysis processes.

The project was designed to manage land use in any area of the globe and was tested for the study area. Hence the resulting output of the GIS-MCDA had to consider, impartially, alternatives over an entire study region. The solution was to overlay a small enough grid on the area of study. With a grid format, the alternatives number can be very high, which in AHP MCDA like methods is not applicable (Malczewski, 2006). The idea is to develop a GIS-MCDA that is data flexible, taking in raster or vector-format data, and produce a suitability map of any study region, in a grid like format.

The research questions are:

1. Can the AHP be used in a SDSS, to support the diverse land use decisions ?
2. Is it possible to use open source technology to build a SDSS based on AHP?
3. Can the AHP be defined to support large numbers of alternatives?

The objectives of the thesis are to:

- Test the possibility to integrate AHP in a SDSS to better support land use related decisions.
- Test the possibility of open source technology to develop a SDSS using AHP.
- Test the possibility of using the AHP with a large number of alternatives.

The thesis is organized in the following way: the literature review will be presented in section 2, where an introduction to the MCDA methods in GIS will be given and a deeper investigation of the used algorithm used will follow. The use of the AHP in GIS will be reviewed also. In section 3 the study area and the data available for the project will be described. Section 4 describes the methodology used in the development of the software application. This chapter will be divided into the requirements, the system design, the implementation and the verification. Section 5 will describe the results and discussion of the testing example. To finalize Section 6 will present the conclusions and possible future improvements of the thesis.

2. LITERATURE REVIEW

This chapter will describe and characterize previous work, on the topic discussed in this thesis. It will also describe the analytic hierarchy process which is the MCDA used in the implementation.

2.1. Multi-criteria decision analysis in GIS

A Multi-criteria Decision Analysis (MCDA), is a decision making analysis that will bring into the decision multiple criteria affecting it. These methods began to emerge during the early 1970s, when researchers from the economical and decision making fields identified weaknesses in the neoclassical view of decision making and site location (Carver, 1991). MCDA are used in computer-based systems called Decision Support Systems. There are many MCDA algorithms, but they can be divided into Multi-attribute (MADA) or Multi-objective (MODA) decision analysis (Malczewski, 1999). The MODA techniques are characterized by a Multi-Objective problem decision. These methods are continuous, because the best solution may be found anywhere in a feasible region. The MADA techniques are characterized by Multi-Attribute decision problems, and are assumed to have a predetermined number of alternatives. The AHP is classified as a MADA technique (Malczewski, 2006).

When the decision being made has a spatial character it is called upon a Spatial Decision Support Systems (SDSS). The efforts to integrate GIS with MCDA have been contributory for developing the prototype of spatial decision support systems (Goodchild, 1993). These systems integrate the GIS with DSS to solve problems with a spatial dimension (Silva, Alçada-almeida, & Dias, 2014). The use of Multi-Criteria Decision Analysis in Geographic information Science has been widely applied and used since 1986 (Sugumaran & DeGroote, 2011). Carver (1991) is one of the most cited and one of the first researchers to apply the MCDA methods to GIS (Malczewski, 2006). The blooming of this combination came around the year 2000 with the broader use of technology, the recognition of the decision analysis and support systems by the GI Science community, and also resultant of the lower costs and user-friendly MCDA software (Malczewski, 2006). MCDA use in GIS has been widely researched (Lin et al., 2014; Malczewski, 2006; Marinoni, 2004;

Massei, Rocchi, & Paolotti, 2013; Sugumaran & DeGroote, 2011). Some of those researches are:

- A decision support system for optimizing dynamic courier routing (Lin et al., 2014).
- Development of a web-based multi-criteria spatial decision support system for the assessment of environmental sustainability of dairy farms (Silva et al., 2014).
- An exploratory approach to spatial decision support (Jankowski, Fraley, & Pebesma, 2014).

In the first mentioned research, developed by Lin *et al* (2014), the topic of the research is to create a decision support system (DSS), that optimizes dynamic courier routing operations. This work is attached to a spatial component, because routing problems are always have a spatial relation. Therefore, even if the research does not mention it, it has an intrinsic spatial component. The word dynamic means that, while the courier is executing the planned route, the DSS can exclude costumers that canceled the courier service or add new ones.

The proposed methodology to optimize the dynamic courier routing, is to integrate a hybrid neighborhood search algorithm with a DSS. After reviewing the literature on the scope of the research, the researchers conclude that there is potential for developing a DSS integrating a dynamic vehicle routing model. The proposed hybrid neighborhood search algorithm is composed of a heuristic method called IMPACT, a Variable Neighborhood Search (VNS), and a removal insertion heuristic. The IMPACT heuristic measures the three criteria for measuring the impacts that a costumer has on the routing of the courier. The own impact, the closeness between the time of starting the courier service and the earliest service time. The external impact, the affected difference of time window of un-routed costumer j if costumer c right before or after j. And the internal impact, defined by the distance increase, time delay and time gap, if consumer c is inserted between i and j. The VNS is used to improve the initial solution, obtained with IMPACT. This algorithm intends to optimize the number of couriers by using intra-route and inter-route searching processes. The final heuristic is used to remove or insert in the route of the courier a costumer. By removing the costumers that have been inserted in the

routing plan, and reinserting these with the new costumers excluding the canceled ones, this heuristic is able to redo the routing plan dynamically while the courier is executing his route. The research concludes that the hybrid neighborhood search approach is able to minimize traveling distance with fewer vehicles. It also concludes that for real-time environment it is an appropriate algorithm for decision making.

Another research reviewed tries to assess the environmental sustainability of dairy farms in a Portuguese region using a Web-based MCDA method in ArcGIS software. It proposes the use of a specific MCDA method, ELECTRE TRI. The research focuses on the main Portuguese milk production area, Entre-Douro-e-Minho.

The methodology proposed is the use of a Web MC-SDSS as a fully dynamic GIS-MCDA integration. This means the interface is completely integrated in a single system. The GIS platform used is the GIS proprietary software ArcGIS 9.3. The ELECTRE TRI method is an outranking MCDA approach. It is based on the pairwise comparisons between potential alternatives. The comparisons use an outranking relation, where one alternative outranks another if the former is considered, "not worse than" the latter. ArcGIS provides a macro development environment using Visual Basic for Applications (VBA) programming language allowing to extend its functionalities. The researchers conclude that the user-interface for configuration, prediction, visualization, and analysis of the model is user-friendly. The interface is the same for all procedures allowing a uniform, transparent and less technical effort demanding process from decision makers.

To finalize the review on diverse MCDA methods used in GIS a more complex, yet simplistic approach will be reviewed. The research by Jankowski *et al* (2014), attempts to solve problems with spatially-explicit decision variables, multiple objectives, and auxiliary criteria preferences. This research proposes the combination of two MCDA, a fuzzy logic system and a spatially adaptive genetic algorithm. The Fuzzy logic system, is based on a Multi-objective algorithm MOGA, a Multi-objective Genetic Algorithm. Firstly MOGA solves a mathematical optimization model achieving a Pareto non-denominated solution set. Then a MCDA using interactive decision maps technique is used to analyze trade-offs

between the selected Pareto-optimal solutions. Once the decision maker selects candidate solutions, these are evaluated as a multi-criteria decision problem. The research does not suggest a specific MCDA algorithm, instead it states that for each purpose the pros and cons of the algorithms should be evaluated. And adds the fact that spatial explicit criteria should be used in this final step of the decision. The researchers conclude that this approach can be applicable to a broader variety of problems.

2.2. Analytic hierarchy process

The MCDA approach used in this thesis is the Analytical Hierarchy Process (AHP). The AHP was developed by Tomas L. Saaty in 1980, and as a multi-criteria decision making analysis arranges the factors in a hierarchic structure. The structure is composed of an overall goal to criteria, sub-criteria, and alternatives. This hierarchic arrangement serves two purposes: first, it provides an overall view of the problems complexity; and secondly it helps decision makers assess whether the issues in each level of the hierarchic structure are in the same order of magnitude, so that decision maker can compare these elements accurately (Saaty, 1990). The advantage of the AHP over other multi-criteria decision support methods is that it takes into account the decision maker's intuitive knowledge into the analytical decision (Saaty, 2000). The idea behind the AHP is to clearly define the decision being made and the criteria that affect it. To use the AHP, one should be familiar with the decision's subject (Saaty, 1990). The decision affecting criteria should be defined by experts on the problem's subject.

The AHP is arranged into two parts, the structure of the problem, and the weighting of the various parts of the problem's structure. Firstly the decision maker must decompose the decision in hierarchical sub-problems easier to understandable (Saaty, 1987). The hierarchy elements can be related with any aspect of the decision's subject, tangible or not, precisely measured or not; in other words, anything associated to the decision. The criteria should be chosen by an expert on the subject, representing reliably the real affecting factors of the decision. When the problem is, what stock to buy, one can ask a student of economics, what is his opinion on the factors affecting the problem. On the other hand, if one asks the student's professor, a better judgment might be obtained. So, the decision maker

has to define what is the focus of the decision, the criteria, and if needed, sub-criteria that affect the decision. Finally the possible alternatives for the decision.

Secondly the decision makers must evaluate the various elements systematically, comparing them to one another, in pairs. This comparison is done using Saaty's fundamental scale of comparison ranging from 1 to 9, see table 1 in annex (Saaty, 1987). This scale of importance defines the value 1 as factors having "equal importance", and 9 defines "extreme importance" of a factor over another factor. The decision maker will have to pairwise compare the criteria and sub-criteria defined. And also the alternatives according to each of the criteria defined. Saaty (1987) gave an easily understandable example and illustration of this process. His example expresses the decision of a high school student wanting to go to college, and does not know what college to choose. So the structure of this decision is illustrated in figure1:

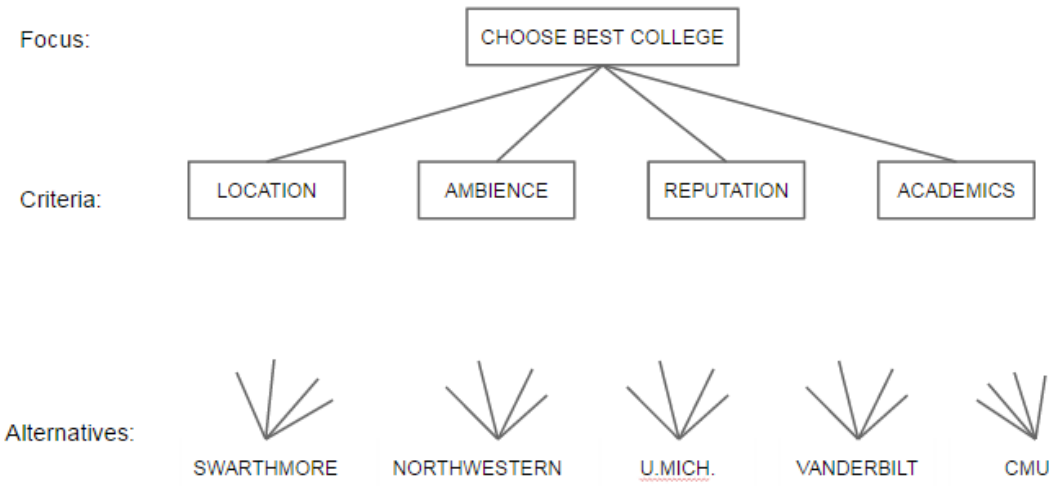

Figure 1 Hierarchy for choosing college example (Saaty, 1987).

As you can see the main goal of the decision is to choose the best college. The criteria that affect this decision are: Location, Ambience, Reputation and Academics. Where Location, is perceived as the farther away from the student's home the better. Ambience, expresses how happy the student felt at the university. Reputation, represents how the university is rated. And Academics representing the personalized attention, small classes over large ones. The student, which in this case is the decision maker, also defined the possible alternatives to the final decision:

Swarthmore College, Northwestern University, the University of Michigan, Vanderbilt University and Carnegie-Mellon University. After structuring the decision into a hierarchy, the next step is to execute the pairwise comparison of the criteria using Saaty's fundamental scale, resulting into a pairwise comparison matrix, in table 1.

Focus	Location	Ambience	Reputation	Academics
Location	1	1/7	1/5	1/5
Ambience	7	1	2	3
Reputation	5	1/2	1	1
Academics	5	1/3	1	1

Table 1 Saaty (1987) pairwise comparison of the criteria college example

When the comparison of all criteria with all criteria is done, the result is a pairwise comparison matrix. This matrix is called reciprocal matrix. This is due to the procedure used to compare the factors, in this case the criteria. The decision maker will compare Ambiance against Location, and it's intuitive, or subjective, judgment is translated into a value of Saaty's fundamental scale of comparison. The value the student decided to input was 7. This means that, the Ambience of a university is strongly favored and its dominance over the Location criterion is demonstrated in practice, see table 1 in annex. In the opposite comparison the reciprocal value is entered. To verify that the values are not entered randomly, a matrix has to respect the consistency condition, $a_{ij}a_{jk} = a_{ik}$. Although the matrix above does not respect this principle its inconsistency can still be acceptable. This is verified with the consistency ratio. The consistency ratio is based on the permissive that although, $a_{12}a_{23} \neq a_{13}$, this difference is compensated by other differences. Some differences are above the expected value and some are below the expected value. If they mitigate each other up to a certain point 0.9 then the matrix can be accepted as input (Malczewski, 1999). This ratio can be calculated, through the consistency index divided by the random index. If the Consistency Ratio (CR) is less than 0.10 the ratio indicates a reasonable level of consistency in the pairwise comparisons,

meaning the matrices are acceptable. With a reasonably consistent comparison matrix the next step is to calculate the eigenvectors of the matrix. The Eigenvectors give the decision maker an overview of his pairwise comparisons in the form of priorities (Saaty, 2000). This priorities of the criteria establish an order of importance magnitude for the decision. The following step for the decision maker is to build the pairwise comparison matrices of the alternatives according to a specific criterion. Following the example, exposed by Saaty (1987), an example of the pairwise comparison matrix of the schools according to locations, in table 2.

Location	Swarth	Northw	U.Mich	Vanderb	CMU	weight
Swarth	1	1/4	1/3	1/3	7	0.115
Northw	4	1	2	3	7	0.402
U.Mich	3	1/2	1	3	6	0.283
Vanderb	3	1/3	1/3	1	4	0.163
CMU	1/7	1/7	1/6	1/4	1	0.037
CR = 0.092						1

Table 2 Saaty (1987) alternatives pairwise comparison matrix college example.

In this example we can see that the alternatives are being compared against each other but according to one of the criteria defined, Location. For example, Northwestern University, according to Location, is 4 times more desirable than Swarthmore College. And reciprocally Swarthmore College has ¼ of the desirable Location than Northwestern University. The consistency ratio and the eigenvectors are calculated for the alternative's matrix, as they were for the criteria matrix. This process is done for all criteria, meaning there will be as much alternative's pairwise comparison matrices as the number of criteria, because for all criteria the alternatives are pairwise compared. This will give the student, the decision maker, an overview of the alternatives' priorities. Obtaining the priorities of the criteria and the priorities of the alternatives considering each criteria, the alternative's priorities, are multiplied by the priorities of the criteria. It should be notice that, the eigenvectors' values are the priorities. This multiplication results into a ranking of

the alternatives or the global priority vector for the alternatives (Saaty, 1990), see equation 1.

$$\begin{bmatrix} 0.115 & 0.034 & 0.521 & 0.564 \\ 0.402 & 0.539 & 0.235 & 0.209 \\ 0.284 & 0.250 & 0.147 & 0.132 \\ 0.163 & 0.121 & 0.038 & 0.040 \\ 0.027 & 0.056 & 0.059 & 0.055 \end{bmatrix} \times \begin{bmatrix} 0.053 \\ 0.491 \\ 0.238 \\ 0.218 \end{bmatrix} = \begin{bmatrix} 0.270 \\ 0.387 \\ 0.201 \\ 0.086 \\ 0.055 \end{bmatrix} \quad (1)$$

With a better understanding of the MCDA used in this research its uses in GIS could give a better insight and understanding of the advantages of this combination this will be discussed in the following section.

2.3. Analytic hierarchy process in GIS

The study of the integration of the MCDA, Analytical Hierarchy Process in GIS has been broadly researched (Malczewski, 2006). The topics being reviewed here are, agriculture (Beigbabayi, Mobaraki, Branch, & Branch, 2012), environmental (Ying et al., 2007) and locational evaluation (Safian & Nawawi, 2012). The last research reviewed will be the implementation of a AHP tool within GIS environment (Marinoni, 2004).

The first research focuses on the agriculture topic, scopes the evaluation of site suitability for Autumn Canola cultivation in the Ardabil province in Iran (Beigbabayi et al., 2012). To assess site suitability, the spatial modeling is one of the strategies that allow its scientific evaluation (Beigbabayi et al., 2012). The researchers state that different factors must be taken into account when evaluating site suitability, requiring for a MCDA. The MCDA chosen was the AHP because of its accuracy. The researchers defined a series of criteria that influence the cultivation of Autumn Canola. From these criteria a series of sub-criteria were defined. For example, for the climate criterion, the frost, precipitation and humidity were defined as some of the sub-criteria. The alternatives were pairwise compared according to each of the sub-criteria. It is of important notice, that the alternatives of the research were not geographic features, but instead, a series of intervals of the

sub-criteria defined. Taking the sub-criterion humidity, the alternatives were defined as follows:

Alternatives	Humidity %
A	70-80
B	65-70
C	55-65
D	40-50
E	>80 or <40

Table 3 Beigbabayi et al. (2012) alternatives table for humidity sub-criteria.

This way of structuring the alternatives is very intersting because it is independent from the alternative's area, or number. The comparisons of the alternatives were based on the best known conditions for the cultivation of Autumn Canola. The comparison matrices of the criteria, sub-criteria, and alternatives were tested for their consistency, and the eigenvectors were calculated. The priorities were obtained for all criteria, sub-criteria and alternatives. After the priorities were defined, these were matched with the corresponding geographic data of the Ardabil province. The province's suitability for the cultivation of the Autumn Canola was devided into five categories. These categories liked to the corresponding geographic data provided the researchers the overview of site suitability for Autumn Canola cultivation. The GIs software used was ArcGIS 9.3.

The next research reviewed mentions the benefits of using AHP in GIS to evaluate the eco-environment quality in the Hunan province of China (Ying et al., 2007). In this research the criteria has four hierarchical levels, where sub-criteria from one level relate to criteria of the level directly above. The main goal is to evaluate the Eco-environmental quality of the 88 counties that make up the Hunan province, this makes it the first level of the hierarchy. The second level is divided in 4 aspects of the environmental quality assessment. The third level is divided into 15 levels that relate to the ones above. The fourth level is dived into 28 sub-sub-criteria that are also directly related to the sub-criteria above. The comparison matrices were built taking into account always the hierarchical level above. Experts' advice was taken

to correctly compare the criteria. The alternatives in this research, contrary to the previous, are the 88 counties that make the Hunan province of China. The alternatives' eco-environmental quality was assessed by using a synthetic index and sub-index. The synthetic evaluation of each county was the sum of the corresponding weight values of all related factors.

It is mentioned in the research that the combination of AHP with GIS is useful for this study, because the AHP has the advantage in multi-indexes evaluation and the GIS is good at spatial analysis.

For this research as for the previous a consistency ratio was calculated to measure the consistency of the pairwise comparison matrices, reassuring the non-randomness. Once again the use of the proprietary software ArcGIS is mentioned.

The next research reviewed uses AHP and GIS on the evaluation of locational characteristics quality for purpose-built offices in Malaysia (Safian & Nawawi, 2012). The term Purpose-Built Office (PBO), meaning an office designed and constructed to serve a particular purpose. The evaluation focused on 5 PBOs within the Kuala-Lumpur Golden triangle. The combination of a GIS with AHP provides an effective evaluation on PBOs and can be profiting.

To overcome the problem of the locational analysis covering spatial and non-spatial data it was proposed the technique of combining AHP with GIS. The research involved two phases of analysis, the AHP and the GIS. The AHP was the first phase. It was used to identify the quality level of locational characteristics on PBOs. The researchers required the help from respondents to assess this characteristics. Five characteristics were determined by presenting a questionnaire form to PBO's tenants. These represent the criteria, they were location of commercial features, availability of transport options, transportation and parking places, vehicle flow and efficiency of property market. The tenants of each PBO, were then asked to weight the importance of the locational characteristics, using Saaty's fundamental scale.

The second phase used a GIS with a network analysis method. A locational quality index was used in order to evaluate the distances, based on the network analysis performed by ArcGIS 9.3. No further details were provided on how this technique was applied. A locational quality index is used to assess the evaluation on locational characteristics of the PBOs.

The final research being reviewed focuses on the implementation of the AHP in ArcGIS software as an extension of the tools provided (Marinoni, 2004). The AHP was implemented using a visual basic programing language in ArcObject. ArcObject is an ESRI software to develop Macros using visual basic programing language for applications. The Macro developed is to be used in ArcGIS environment. The decision to use VBA (visual basic for applications) macros was based on integration capabilities with ArcGIS. Some of them are the use of ArcGIS functionality to its full extent, VBA macros can take advantage of global variables, and the creating, testing and debugging similarity of ArcGIS VB editor to other VB development environments.

The Macro developed accepts all criteria considered relevant as long as it is regionalized and in raster format data set. The user also has to do the necessary transformations to have all raster data sets in the same scale. Each criterion has to be represented with a raster image map. The raster images values need to be in the same scale. Although a standardized scale is not provided the researcher suggests the definition of intervals within the raster values. The values the decision maker considers bad should have the value 0, while better values should be classified with higher values.

The macro developed also helps the decision maker by filling in the reciprocal values in the pairwise comparison matrix of the criteria. The eigenvectors are calculated using an eigenvector calculation routine. Finalizing the process, all classified raster images are multiplied with their corresponding weight and summed up. Each raster cell is calculated according to the classifications done previously. The resulting raster is added to the ArcGIS environment, representing the suitability zones, with respect to the specific land use.

The macro also uses a spatial analyst functionality of ArcGIS. If requested the macro will perform the conversion of the raster image into a polygon vector format. The macro assumes the raster images have the same resolution and extent, meaning the perfect overlap of the criteria raster and the resulting raster. The researcher concludes the implementation of the VBA macro AHP is useful to facilitate land use assessment. The presented VBA macro fills the gap since a commonly used GIS, ArcGIS, does not provide this functionality. The macro is intended to provide

a template for users who are working in the field of land use assessment, and other geosciences where regionalized criteria play a role in the decision making.

3. METHODOLOGY

3.1. Requirements

The guidelines to the development of software were defined as follows: (i) execution in useful time, (ii) process big data sets of alternatives and (iii) a user friendly way of thinking. These arose some constraints that will also be discussed in this section.

Since the application software had a possibility of being implemented on the web later on, the process had to be done in feasible time. The processing was broken into two parts, initially, the processing of the geographic information was done with a GIS, ArcGIS 10.2. This process could take from three to almost eight hours depending on the number of alternatives and criteria that were defined as affecting the problem. And the second part, the AHP, could take almost three more hours to conclude. This were not acceptable for a web based application. This processes needed to be optimized. The geographic processing of the data needed to be automated also due to the time consuming restriction.

Another requirement is the processing of big datasets, or big sets of alternatives. Because the alternatives for the problem had to cover the entire study area, this meant that the number of alternatives could be very large. For example, the study area where the AHP-SDSS was applied is approximately 149 900 km^2, in the case the user wanted a resolution of 2 by 2 km on the output map, the application software needed to process this area divided into 4 km^2 parcels. This resulted into approximately 37 475 parcels, these being the total number of alternatives. Additionally, with the AHP one needs to compare the alternatives in pairs, building the pairwise comparison matrix. This process is done for all criteria. Each comparison matrix would have about 1 404 375 625 elements times the number of criteria. This would be a dreadful task for the decision maker. The time spent comparing alternatives according the criteria would be immense. The application software could not let the user do the pairwise comparisons by hand. Therefore an automatic way of comparing the alternatives had to be implemented.

The last constraint is related to the output of the application software, this required to be a suitability map for the problem proposed, supporting the decision making

process. As the alternatives had to cover the entire area, and the suitability values were to be presented as a percentage, the most adequate data format would be a raster image map, where the cell values are the suitability value of that cell.

The application software as being part of the environmental strategic evaluation of the multi-sectorial plan for land use management of the Zambezi River Valley project had to be able to take as input all the available data layers in the web-based GIS developed.

This guidelines and restrictions affected the application software's development, requiring it to consider the most alternatives possible as input. Requiring furthermore, a flexibility in the input data. Resulting into a more error-proof software, increasing at the same time its complexity.

3.2.System design

The system was designed to have 2 parts, the first organizes the input of the user, and the second processes it into a suitability map. The first part, due to the input flexibility need, was designed to standardize geographic data, formatting all input data into raster format with the same resolution, reference system and extensions. The second part processes the resulting data outputting a suitability raster format map.

The process starts with the user input, followed by a geographic processing of data, which goes through the AHP, resulting into an output raster, in figure 2. This process will be detailed along this chapter.

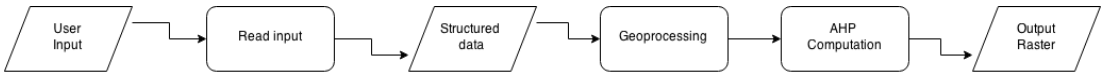

Figure 2 Overview flowchart of the system design.

3.2.1. Geographic processing

The geographic processing of data will from now on be called geoprocess for simplification. This process only produces standardized geographic information for the computation of the AHP. The geoprocess depends strongly on the Geospatial Data Abstraction Library (GDAL, 2015).

The GDAL library is used to read and write geospatial data formats, and is released under the X/MIT style Open Source license. This library comes with a variety of command line utilities for data processing and translation. These are the GDAL functionalities used in the geoprocess: the vector utility program ogr2ogr and the raster utility programs *gdalwarp*, *gdal_rasterize* and *gdal_proximity.py*. These programs will be discussed later in this chapter.

The geoprocess is illustrated in the flowchart in figure 3. This flowchart illustrates the user input being separated into 3 parts, the data, the geographic properties and the grading tables. The data, it's the GI layers available in the web-based GIS, either in raster or in vector format. Each layer is assumed to be a criteria of the AHP. For example, a shapefile of the population density by districts, or a raster file of the slope, are each a different criterion.

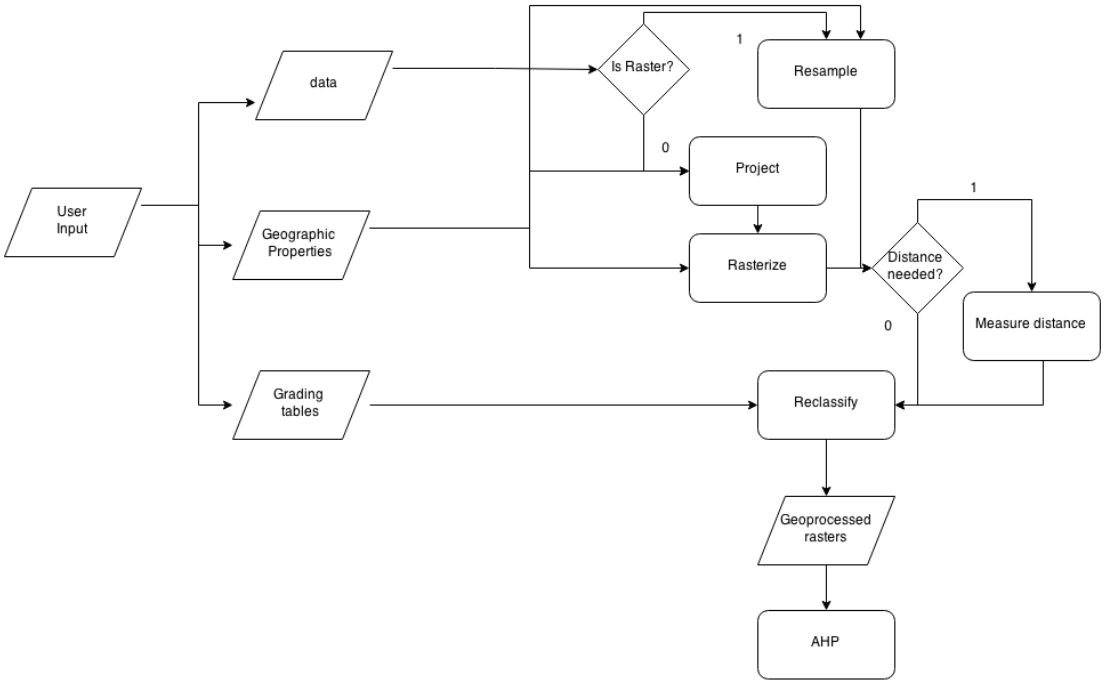

Figure 3 Flowchart of the geographic process

The geographic properties are information needed for the GDAL programs. Like the georeferenced extent and resolution of the output raster file, and the desired spatial reference.

The user will need to define intervals for the geographic data values, and give importance to these intervals, requiring therefore grading tables. Translating the real value of data into a standard scale, where all types of data can be compared. This grading process is used to simplify the problem of pairwise comparison, mentioned before. The grading tables will be the translation between the geographic data values and the grading given by the user to this values. The grading scale will vary only from 1 to 10 in integer numbers (see table 4), where 1 indicates the worst possible values and 10 indicates the best possible values.

Grade value	Explanation
1	The worst possible option for the decision
2	It is an unviable option for the final decision
3	It is a bad option for the final decision
4	It is a dislikable option for the final decision
5	It is indifferent for the final decision
6	It is a decent option for the final decision
7	It is a good option for the final decision
8	It is a very good option for the final decision
9	It is a great option for the final decision
10	The best possible option for the final decision
0	Exclude from output

Table 4 Grading Scale explanation

For example, using slope data, the decision maker might say, for a specific problem, the slope values from 0% to 3% could be the best possible, and give it the value 10 in the grading scale. From 3% to 5%, the slope would still be very good for his final decision and give it a value of 8. On the other hand, if the slope is higher than 5%, it would be the worst possible for his final decision giving it a value of 1 (table 5).

Slope value (%)	Grade (1 - 10)
0% - 3%	10
3% - 5%	8
>5%	1

Table 5 Grading table example.

A fact about this scale is that, the user can choose 2 intervals or categories, to be perfect for the final decision, Giving both the same value in the grading scale. For

example, if one of the criterion that affects the decision is soil type, the user can make the following grading table (table 6).

Soil type	Grade (1 - 10)
Basaltic	10
Sandy	1
Clayey	10
Water	0

Table 6 Grading table example.

Table 6 expresses that, a basaltic soil or a clayey soil, would either be the best possible options for the final decision; and sandy soil type would be the worst option for the final decision. One should notice the exclusion of water soil type from the decision. This exclusion should be used carefully, regarding this soil type will be excluded from the output. When applied, the user should have an idea of what he is excluding from the output, because with this the user might be excluding 50 to 90% of the study area. To exclude 50% of the study region even before the computation of the AHP, might lead to a biased set of alternatives.

Another fact about the grading scale, is that the user doesn't need to use the entire scale, for each grading table, the user might use only 3 intervals as showed in table 5. It is advised that the user still uses the entire range of values or categories of the geographic data, because in case a certain value or category is omitted the AHP process will assume it as excluded from the output.

After the data, the geographic properties and the grading tables are defined the application software is ready to run. The first step is to verify which data are in raster format and which are in vector format. If data is in raster format a resample of the raster image will be performed by using the *gdalwarp* program from the GDAL. On the other hand, if data is in vector format, the application software will use the *ogr2ogr* GDAL program to perform a projection of data. And after the projection is done, it will rasterize the projected vector data using *gdal_rasterize* from the GDAL.

Having all data in raster format, with the same extents and resolution, the application software will verify if it needs to perform a measurement of distances. This is due to the possible need of specifying as a criterion, distances to, or from a certain geographic feature. The distance intervals are graded in the grading tables just as any other criterion. The user must specify if there is a need for the computation of distances. If the distances are required, the application software will resort to the *gdal_proximity.py* program of the GDAL library.

If there is no need for the calculation of distances, or when the distances have been calculated, the application software will perform the reclassification. This is, from the grading tables, for each cell in the raster images, the application software will translate its value into the grade provided by the user. One needs to remember that at these stage all data is in raster format, in the same coordinate system, with the same resolution and extent, implying that the cells of all rasters overlap perfectly. The reclassification method will use the grading tables as a translation reference. The application software will read each cell's value and compare it with the respective grading table. For example, in the soil type example mentioned before, at this point, we have a raster map of, let's say, 2 by 2 km cells, where in each cell its soil type value is defined. And according to table 7, all cells that have the basaltic and clayey value will now have the value 10, the cells with the value sandy will now have the value 1 and the cells with the value water will now have the value 0. And this process is done to every cell of every raster map, each of which represents a criterion.

The *gdalwarp* program is an image reprojection and warping utility. This program is used as a command line program and has 41 possible arguments to be used. The ones being used by the application software developed are 7, them being the source file's name and location, the target file's name and location, the target spatial reference set, the target extent of the image, the destination nodata value, the target resolution of the image and the quiet mode.

The source and target file's name are written as the path in computer's file system. The target spatial reference set is the coordinate system of the resulting geographic data, used to make a geographic transformation or projection if needed. This should be specified as EPSG: followed by the coordinate reference system code of the

region in study. The target image extent argument, takes in the x axis minimum, followed by the y axis minimum, followed by the x axis maximum, followed by the y axis maximum, extents of the raster image. This must be expressed in the coordinate reference system units of the target raster image. The nodata value can be -9999, since this value is not common in data.

The target raster's resolution is the raster cells' size. The size of the cell defines, the resolution of the raster. This argument should be represented in the unit of measurement of the target coordinate reference system and as x axis first and y axis second. The quiet mode, states that while running, unless there is a problem along the process, the command line will not show any messages or progress of completion.

To finalize, the *gdalwarp* program is expected to take in any raster image, and to return a raster image with the resolution, extent and coordinate reference system the user specified. Never changing the original values of the raster image.

The next GDAL program is the *ogr2ogr*. As the *gdalwarp*, this program is also used as command line. Even though with different inputs and outputs. The *ogr2ogr*, is designed for converting simple vector data between file formats, performing various operations during the process such as spatial or attribute selections, reducing the set of attributes, or setting the output coordinate system. The operation that the application software will use is the reprojection. The *ogr2ogr* has 50 possible attributes to perform the above mentioned actions. To perform the reprojection, the following arguments are used, quiet mode, the format name, the target spatial reference set, the source file's name and location and the target file's name and location. The source and target file's location and name, are the same as the one mentioned before, in the *gdalwarp* program. Also the target spatial reference set and quiet mode were mentioned. The format name is the specification of the target file format, which can be "ESRI Shapefile", "Tiger", "MapInfo file", "GML" or "PostgreSQL". The format used is the ESRI shapefile, for simplification. The *ogr2ogr* program is expected to take in a vector type data and to perform transformation or reprojection to the specified coordinate reference system. This step is necessary so the rasterization can be done even when data have no coordinate reference system.

The *gdal_rasterize* program burns vector geometries into a raster image. It has 21 possible arguments, but only 8 were used. These are, the source file's name and location, the target file's name and location, the override spatial reference set, the quiet mode, the burn value, the layer name, the target extent and target resolution. The source and target file's name and location format, were mentioned for the *gdalwarp* and *ogr2ogr* program. The override spatial reference set should be the same as the target spatial reference set mentioned before, specified as EPSG: followed by the coordinate reference system code of the region in study. The quiet mode was also mentioned before. As the target extent and resolution were also mentioned before.

The biggest difference is the layer name and burn value arguments. These are not mentioned in the other programs, because they are programs which the input and output data are in the same data format, vector to vector or raster to raster. In this program it is expected that the input is vector and the output is raster. Vector data can have various attributes, for example a vector file can represent visually the municipalities of the Lisbon district, but have several attributes, like the total population distribution, the female population distribution, the male population distribution, the working population distribution, etc. So it is crucial to refer which attribute (or in this case called layer) will be used to write the raster image. This happens because the raster image represents one attribute only. The burn value argument comes in place when the layer is not specified, a binary raster is assumed to be needed. For example, when the criterion is the distance to main roads, the raster resulting should be a binary raster, where the cells that represent the road have the value 1 and the rest of the cells have value 0. This way, and introducing the next program, the *gdal_proximity.py* can compute the distances from the cells that contain only the value 1.

The expected process of *gdal_proximity.py* is that, it will take as input the "ESRI shapefiles" from the *ogr2ogr* and rasterize those vector format files. These result into raster format files with the extent, resolution and coordinate reference system defined by the user. And matching the previous resampled raster from *gdalwarp*.

This GDAL program produces a raster proximity map. This program has a total of 12 possible arguments from which 4 were used. The common arguments with the

previous programs are, the source and target file's name and location and the quiet mode. These should be specified in the same way as mentioned before. The other argument is the distance units, which indicates whether distances are generated in pixels or in the units of the coordinate reference system. The program takes as input a binary raster file and calculates the distances from every pixel to the closest pixel with value 1. This results in a raster file that has all the cells filled with values of their distance to the nearest cell with value 1. And since the binary cell with value 1 represent geographic features, the output of the *gdal_proximity.py* is a proximity map of the geographic features, whether it be roads, ports or nature reserves.

3.2.2. Analytic Hierarchy Process

The process that will be described here, is mentioned in the first paragraph of the system design chapter, and visible on figure 6. The overview of the flowchart of this process is illustrated in figure 4.

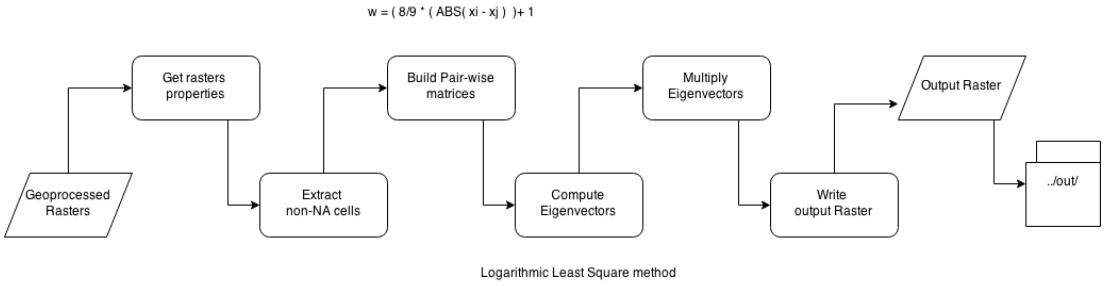

Figure 4 Flowchart of the AHP computation

Remembering that at this point, we have all data in raster format, and with the values of the grading scale, see table 4, all rasters were standardized for the AHP computation. The first step of this process, is to retrieve the raster properties, like the number of rows and columns, the coordinate reference system, the projection and the raster name. Next, all the cells that have nodata values are excluded from the computation. Since they fall out of the study region, there is no need to process these cells. The succeeding step is to compute the pairwise comparison matrices. These matrices are usually filled by the user, as in most cases there is a restricted number of alternatives. In this application we are dealing with possibly around 37 000 alternatives, it becomes almost impossible to pairwise compare all alternatives.

Instead, we use the grading provided by the user, for an automatic computation of these matrices. This will be explained in detail further down. The computation of the eigenvectors is the next step. The eigenvectors of the alternatives' pairwise comparison matrices are computed, as well as, the criteria's pairwise comparison matrix eigenvector. To better understand the process an illustration of the process is given, in figure 5.

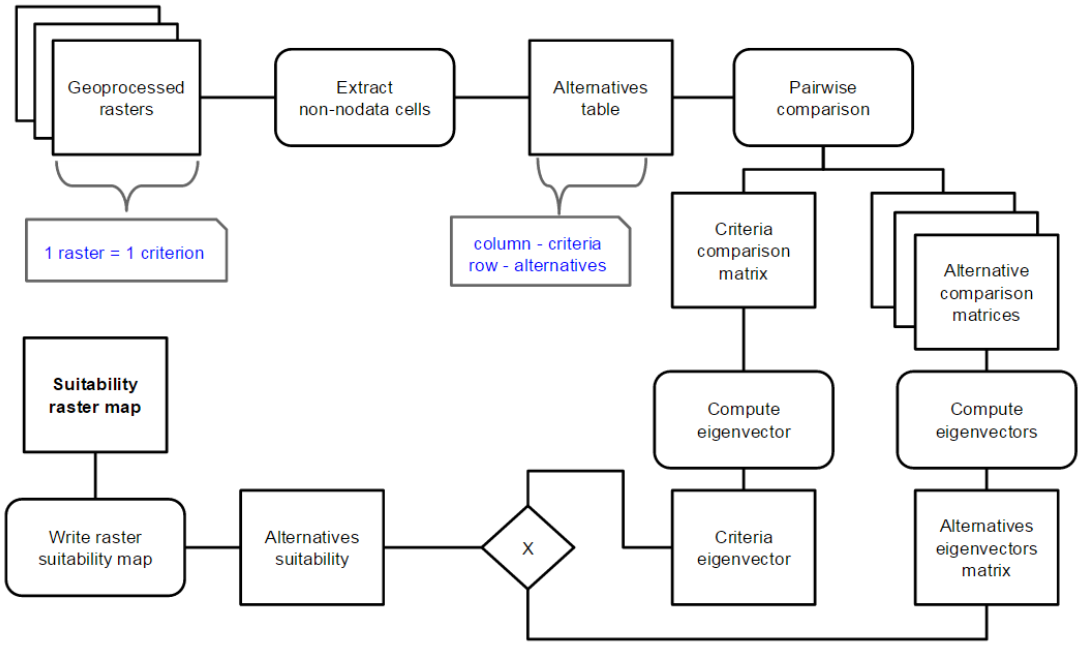

Figure 5 Detailed flowchart of the AHP system design.

After the eigenvectors are calculated these are multiplied by the eigenvector of the criteria, see figure 5, resulting into a ranking of the alternatives, the suitability of each raster cell for the solution of the problem. The ranking is written in a raster image after the normalization, and this is the final output of the application software.

As mentioned in the requirements section, one of these requirements was the need for a big number of alternatives. The idea to solve this was the use of a raster map where each cell is an alternative. When each cell is a possible alternative then it becomes almost impossible to pairwise compare all alternatives, since the number of comparisons is close to the square of the possible alternatives. And we multiply this number by the number of criteria, plus the number of criteria squared, this gives

us the total number of pairwise comparisons. Which for the Lisbon district, for example, with a total area of 2800km^2, and a 1km by 1km cell size, translates into 2800 alternatives. Assuming the criteria were only 3, this turns into 7 840 000 pairwise comparisons for 1 criterion. Multiplied by 3 is 23 520 000 pairwise comparisons of the alternatives, plus the 9 pairwise comparison of the criteria itself. In the end one would have 3 matrices of 2 800 by 2 800 plus one of 3 by 3 to fill in. Could this be done by hand? Maybe, if filled in randomly, where the consistency ratio of the matrices would never be under 0.10, the maximum value accepted (Malczewski, 1999).

To solve this problem the grading scale is used in order to reduce the comparisons done by the user or decision maker. The scale ranging from 1 to 10, see table 4, is used to establish priority of the criteria values. This means that instead of comparing the alternatives with each other according to a specific criteria, the user will order the possible values of each criterion with the grading scale, and the order proposed will always be respected to all alternatives. In case we have 4 alternatives, A, B, C and D, see figure 6, these being raster image cells, and 3 criterion, them being, soil type, slope and population density, the grading tables being table 7, 8 and 9 respectively.

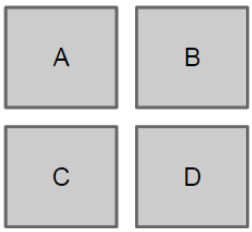

Figure 6 Example alternatives.

Soil type	Grading (1-10)
Basalt	10
Clayey	8
Water	0
Other	1

Table 7 Soil type grading table.

Population dens.	Grading (1-10)
0 - 6	10
6-10	8
10-100	5
>100	1

Table 8 Population density grading table.

Slope (%)	Grading (1-10)
0% - 3%	10
3% - 5%	8
>5%	1

Table 9 Slope grading table.

Now the real values of the alternatives for the 3 criteria are:

- Soil type:
  - A – Sandy
  - B – Basalt
  - C – Clayey
  - D – Water
- Population density:
  - A – 11
  - B – 8
  - C – 2
  - D – 0
- Slope
  - A – 2%
  - B – 3%
  - C – 1%
  - D – 0%

After the reclassification in the geoprocess, the values of A, B, C and D would be standardized to the same scale, the grading scale, as in table 10.

Alternatives	Soil type	Population density	Slope
A	1	5	10
B	10	8	8
C	8	10	10
D	0	10	10

Table 10 Overall grading table of the alternatives.

From this table the pairwise comparison matrices for each criterion are produced using the formula below:

$$s(a_i, a_j) = \frac{8}{9}\left|v(a_i) - v(a_j)\right| + 1 \quad (1)$$

Where,

a_i is one cell of index i, and $a_j$is the cell with index j that $\forall\, i, j \in R, i \neq j$.

$v(a_i)\ and\ v(a_j)$ are the values of the cells a_i or a_j respectively.

This formula will translate the grading scale values of each cell into a pairwise comparison value of Saaty's fundamental scale. Producing the comparison matrices of the alternatives according to each criterion. This process reduces the number of comparisons needed, and no matter the number of cells, the number of grading tables will not change, neither the number of classifications done by the user.

One problem that erupted from the use of the eigenvectors, is that the eigenvectors of matrices of 30 000 by 30 000, take too long to compute making the application software run for hours. So after some research an approximation method was found to overcome this problem, to substitute the eigenvectors calculation by the logarithmic least square method (LLSM) (Saaty & Vargas, 1984), the article discusses the possibility of substituting the eigenvectors by another less complex method that would achieve the same results. This research also suggests that the eigenvalue procedure, is the only one that does not depend on the scale, leads to the measurements of inconsistency, and is the only procedure that preserves ranking of the alternatives under inconsistency conditions.

The problem of inconsistency defines that $a_{ij}a_{jk} = a_{ik}$, and presumably if inconsistency is present in the pairwise comparison matrix, the given comparisons are questionable and new comparisons may be needed (Saaty & Vargas, 1984).

The consistency however might be tolerated up to 0.1(Malczewski, 1999; Saaty, 2000) Using the example above, A has a sandy soil type, B has a basalt soil type and C has a clayey soil type. If the grading scale is the same as in table 5 then, A and all other sandy cells, will always have the value 1 in the grading scale B and all the other basalt cells will always have the value 10 and C and all clayey cells will always have the value 8. With this statement one can apply the equation above to build the comparison matrix of the alternatives A, B and C according to soil type.

$$s(A,B) = \frac{8}{9}|1-10| + 1 = 9 \tag{4}$$

$$s(A,C) = \frac{8}{9}|1-8| + 1 = 7.2 \cong 7 \tag{5}$$

$$s(B,C) = \frac{8}{9}|10-8| + 1 = 2.8 \cong 3 \tag{6}$$

When building the comparison matrix, always a square matrix, we have to compare the alternatives in pairs according to a specific criterion, in this case soil type. The matrix is built with the values from Saaty's fundamental scale, see table 1 in the annex. The main diagonal of the matrix will always contain the value 1, according to Saaty's fundamental scale, the value 1 represents that both activities contribute equally to the objective. Implying, alternative A compared with itself, will always contribute equally in soil type to the objective. Another fact is that the matrices are reciprocal e.g. $z_{ij} = {}^{1}/_{z_{ji}}$(Saaty, 1987). Suggesting, that when one compares A with B (A, B) and defines that A is extremely more important for the objective than B, giving Saaty's fundamental scale corresponding value 9, the opposite comparison (B, A) is 1/9. The matrix below shows the resulting pairwise comparisons of the automatic grading.

$$Z = \begin{bmatrix} 1 & {}^{1}/_{9} & {}^{1}/_{7} \\ 9 & 1 & 3 \\ 7 & 1/3 & 1 \end{bmatrix} \tag{7}$$

One problem arose when using equation 1, the result will always be a value from 1 to 9, considering Saaty's fundamental scale. While sometimes it should be the reciprocal value. Take the comparison of A with B, equation 4, for example, this comparison should result in 1/9, and not 9, because B is the alternative with the best

value for the objective. To overcome this problem, the application software will test which alternative had the highest grading scale value. The one with the highest will get the value of Saaty's fundamental scale, the lowest one will get the reciprocal, as shown in the Z matrix.

Assuming that the columns and rows of the Z matrix are ordered as A, B and C respectively the comparison of A against B was 9, but as B had the highest value in the grading scale the comparison of B against A is the one who gets the value 9 and the comparison A against B will get the reciprocal.

It is said, that for a matrix to be consistent, it should respect the consistency criterion mentioned above, but please note, that $z_{21} * z_{13} = z_{23}$ is $9 * \frac{1}{7} = \frac{9}{7}$. The result should be 3, according to the consistency principle, and 3 is bigger than $\frac{9}{7}$, but then we enter the reciprocal value in z_{32} of $\frac{1}{3}$, which is lower than $\frac{7}{9}$, the expected value according to the consistency criterion. The importance of this observation, is that while one value is less than the corresponding consistency value, the other exceeds it and there is a tendency to compensate. Maintaining the consistency ratio within acceptable values, lower than 0.10 (Saaty, 1987).

The Logarithmic Least Square Method (LLSM)

Saaty based the AHP in the permissive that organizing paired comparisons in matrixes the underlying ratios α could be found, giving a priorities to the factors being compared. The ratios α of a given matrix $A = (a_{ij}),\ \forall (a_{ij}) > 0\ \Delta\ a_{ji} = 1/a_{ij}$, where $(a_{ij}), i, j = 1, \dots, n$ these are estimations of the ratios α. These can be obtained through the eigenvalue, logarithmic least square or least square method (Saaty & Vargas, 1984). The one used is the logarithmic least square method, which estimates α through the vector derived by minimizing (8):

$$\sum_{i,j=1}^{n} \left(\ln a_{ij} - \ln \frac{u_i}{u_j} \right)^2 \qquad (8)$$

The solution to this minimization problem is given by (9):

$$u_i = \left(\prod_{j=1}^{n} a_{ij} u_j \right)^{1/n} \quad , i = 1, 2, \dots, n \tag{9}$$

Imposing the condition that the sum of all ranking values is 1, $\sum_{i=1}^{n} u_i = 1$:

$$u_i = \left(\prod_{j=1}^{n} a_{ij} \right)^{1/n} / \sum_{i=1}^{n} \left(\prod_{j=1}^{n} a_{ij} \right)^{1/n} \quad , i = 1, 2, \dots, n \tag{10}$$

The LLSM as being the most practical, to implement, of the three methods proposed by Saaty & Vargas (1984), was applied in the application software to substitute the eigenvalues computation. This method obtains the u_i, ranking values, using equation 9. Suggesting, the logarithm of base n of the multiplication of all values of the matrix A by row, be divided by the sum of all these.

When consistency is present the logarithmic least square method produces identical solutions to the Eigenvalue method (Saaty & Vargas, 1984).

3.3.Implementation

During the implementation and development of the application software, criteria were chosen and values for the pairwise comparison were given. Although this criteria and values should be defined by experts in the subject of the problem. As I am not an expert on the problems characterized, the spatial outputs of the application software have no relevance to make decisions or assumptions about locational suitability. The purpose of this examples was only to test the most broadly data types and formats possible, ensuring the application software's data flexibility.

The application software was initially designed to run on a Windows operating system, this because it is one of the most commonly used operating systems in the world, and the machine available to develop the application software had this operating system installed.

The first prototype of the program was intended to test the possibility of thousands of alternatives, and test the evidence of Tobler's law, by achieving a spatial continuity. With this said, the first prototype was developed to compute only, the AHP complex calculations, the automatic comparison of the alternatives, the computation of eigenvectors, and their multiplication to attain the final suitability ranking of the alternatives. As mentioned in the system design chapter, to have the possibility of thousands of alternatives, it is almost impossible to do the pairwise comparisons by hand. So the grading scale was introduced from the beginning.

The geoprocess, in the first prototype, was done using ESRI software, ArcGIS 10.2. The result of the geoprocess was a grid-like vector map covering the study region, where the criteria were the attributes, and the alternatives each grid cell. The initial data had to be clipped, vectorised, projected, areas calculated, edited, reclassified and dissolved one by one, using ArcGIS functionalities. This process could take almost a day depending on the cells' size and the number of criteria. In addition, for each criterion a specific process had to be thought and executed. The output was, as mentioned, an ESRI shapefile. A normal shapefile usually has a minimum of 4 files that are read by the GIS, each of this files contain a specific part of the shapefile's information. One of this files is the dbf file, or otherwise known as the database file. This file contains non-geographic data such as, alphanumeric data,

and is structured like a table, where one ID corresponds to one, and only one, feature in the map with its attributes information.

For the first prototype, the software used to develop the AHP computation was R (R, 2015), because of its mathematical and statistical functionalities. R is a scripting language and software for statistical computing and graphics. It can be compiled on windows operating system with ease, just by executing the execution file. It's an Open Source project, meaning it has no costs for the user, under the General Public License (free software license). The R language is easy to learn with some basic knowledge of programming. And has the possibility of efficiently handling and storing, on temporary memory, complex data. It has an extensive library with large variety of extensions for various purposes. This extensions for R software are commonly called R packages, and can be developed by any user, as long as the metadata standard style is filled and respected. This gives the opportunity to users to develop the tools they need, and might not yet be developed. The package used in the first prototype was Foreign, for its function of reading and writing dbf files.

The first R script was a developed to process only the AHP computation, therefore, to start with, the dbf file of the shapefile was read. The columns were extracted from the table, and stored. It is important to notice, that the reclassification of the grid cells was done. Subsequently, the equation 1 mentioned in the system design chapter, is used to make the pair wise comparisons using Saaty's fundamental scale values of 1 to 9 according to table 1, in annex. The method of filling the matrix is the same as described in the system design chapter. After the pairwise comparison matrices of the alternatives are computed, the criteria matrix had to be built. Since the information about the criteria was not in the dbf file, the values were inserted manually in the script to compute the eigenvector of the criteria. Finally the eigenvectors of the alternatives are concatenated into a matrix table, where in the columns each criterion is represented, and as rows each one of the alternatives. The alternatives eigenvector matrix is multiplied by the criteria eigenvector, resulting into a single column of the alternatives' ranking. This column is then written into the dbf file. To visualize the results we fell back to the GIS used to process the geographic data, ArcGIS 10.2.

Initially it was stated that the hypotheses being tested were: the possibility of having thousands of alternatives, and if there was spatial continuity among the alternatives. After observing the results, and some corrections to the initial code, these hypothesis were proven to be true. The program would process around 1 658 alternatives, and since it was a grid map of the study area, the resolution was 10 by 10 kilometers, similar to a raster image map. Also a visible continuity was observed, since neighboring grid cells had similar values in the resulting suitability map.

With the hypotheses tested and verified, it was time to optimize the process, because in this first prototype the time needed to execute all AHP computation was about 3 hours, the criteria had to be individually processed and analyzed to enter the AHP computation. Since the data decision makers use, might not be so well planned and controlled, the application software needed to be more robust. At this point, the program would not take in all types of data, nor criteria, since the alternatives were grid cells, it would not take as input a line or point shapefile.

Dealing first with the time optimization, some "bottle necks" were identified, this is, the steps of the process where most time was being consumed. One of the calculations that took a long time was the eigenvectors calculation. To optimize this part, we decided to reduce the number of loops, performed by the application software in the calculations. Using Saaty's principle of the reciprocal matrix, $a_{ij=1/a_{ji}}$, the number of loops was reduced to half because we could calculate only half of the matrix, and fill in the rest of the matrix with its reciprocal values. After this optimization, the program still took about 2 hours to run.

R programming language is based on C programming language, and C is an efficient programming language for general purpose, therefore better than R. The translation of the R scripts to C, was the next step in the optimization process. Consequence of being a programming language closer to the machine language, which the computer uses to execute actions.

As a result, the automatic pairwise comparison of matrices code, was translated into C programming language for optimization proposes. And the time of execution was reduced from 3 hours to 30 seconds, *ceteris paribus*. Although if the number of criteria and alternatives was increased, the time needed to execute would again increase.

The second “bottle neck” found was in the computation of the eigenvectors, this could increase the execution time exponentially, if the criteria was doubled, or even interrupt the execution of the application software. To overcome this problem, the section of the code responsible for this computation, was also translated into C programming language. Causing a reduction of the execution time, back to 30 seconds or 2 minutes depending on the resolution of the grid cell maps and the number of criteria. This maintained the execution time in an acceptable execution time.

When the resolution of the grid was increased to 2km size cells, to obtain a better and smother spatial distribution, the application software was blocked by the operating system. The application software was using more than 11 gigabytes of memory to execute one comparison matrix. The Windows operating system does not let software such as R, to use more than 11 gigabytes of memory in one process. This is a safety measure to protect the machine from virtual attacks. The solution found, was to use a UNIX based operating system, Linux mint, for its low consumption of memory, and because the limit of memory used by software was higher.

After achieving reasonable execution times for the AHP computation, with no problems for 4 km^2 (2 by 2 km) cell size, for the study area. The next step was to optimize the input data accepted. This due to the application’s software code need to change when a new criterion was added or removed. Reducing the application’s software flexibility in adding or removing criteria. This was a big limitation of the application software’s versatility.

The solution found was to use raster format data as input, this way, the lines and points could be represented as binary raster maps and could be included in the AHP computation. This solution also provided the flexibility needed in adding and removing criteria. With this solution, each raster map would be a criterion, since raster maps do not store multiple attributes for the same cell. One other advantage of using raster maps is the lower need for memory, and the processing of data is computationally less demanding. But, as everything has its benefits, it has its costs, this matter was no exception, the geographic processing of the data became more complex, and time consuming. Mainly because the process of rasterization was

done using a GIS. Whether it was ESRI's ArcGIS, latest version to the moment, 10.2 or Quantum GIS, latest version to the moment, 2.6, the geographic processing of the original data, was not becoming less complex or time consuming. After some research, about raster transformation the Geographic Data Abstraction Library or GDAL was chosen to process the data. Initially through command line separately from the AHP computation. Adding to this required flexibility, all the AHP computation was translated to C language, where there was no need to specify the number of criteria being used, just the location of each raster map.

The GDAL became a very important part of the whole process, since it can rasterize, project, resample, and preform measurements to the layers, very quickly, and with simple command line instructions the geoprocess was optimized. The geographic processing execution time was reduced to a few minutes instead of hours. The programs of the GDAL used, are mentioned, and fully described in the system design chapter.

At this point, the program had a first part, the geoprocess of data the user wants to use, done with GDAL programs mentioned in section system design. And a second part of the program that is written in C programming language, and computes the AHP, resulting into a raster suitability map of the study region.

Two separate processes, can be discouraging to use by decision makers. Consequentially, the application software was integrated into one structured process, which internally, without the end user's knowledge, is separated into the geoprocess of data and the AHP computation. With this change, the reclassification process of the rasters was still done with the help of a GIS, had now do be done internally. As the GDAL library did not offer a program similar to the others, simplifying the implementation and use, it was hard coded in C programming language.

In conclusion, the final version of the application software is structured with a main folder called geoahp, sub-folders, and files illustrated in figure 7:

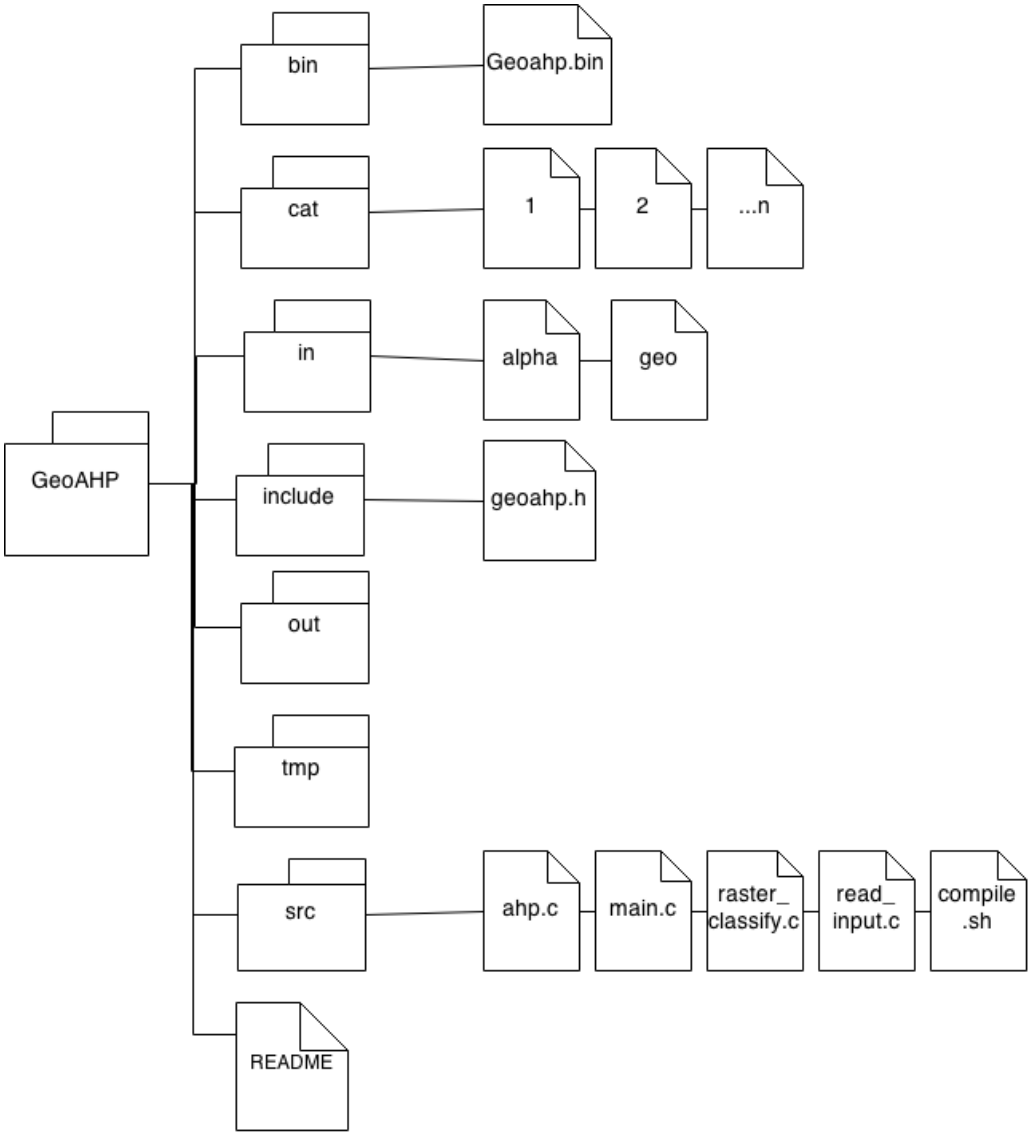

Figure 7 Folder structure of the application software.

- bin, where the binary version of the compiled application software is stored.
- cat, where grading table files are stored.
- in, where the geographic information and data location is stored.
- include, where the header files are stored.
- out, where the output file is stored.
- src, where the hard-coded C files are stored.
- tmp, where the necessary temporary files are stored.
- and a README file with a description of the application software's structure.

The application software's interface is still being developed. The user is supposed to be able to: define the number of criteria, specify where the data files are located for each criterion, specify whether the criterion needs the computation of distances or not, grade each criterion, specify the number of intervals for each criterion, specify the values of each interval and their grade, and finally specify the resolution, extent and reference system of the output file. An illustration of the user interface being developed is shown in figure 8.

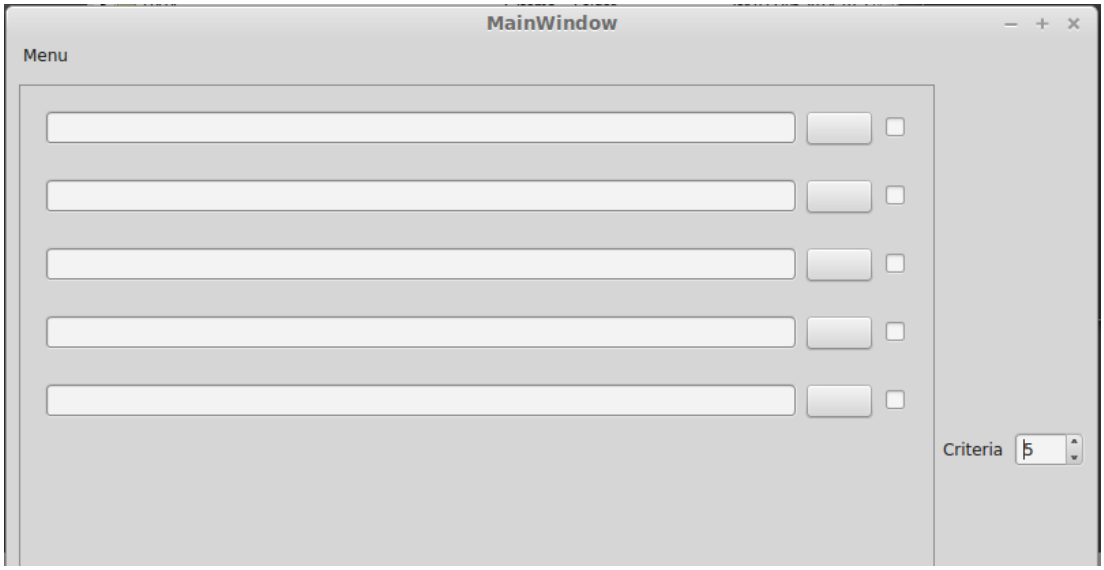

Figure 8 User interface of the application software, in development.

The user interface is being developed using Python programming language and the software QT designer. QT designer is a software to support the development of cross-platform graphical user interface. The use of Python programming language, is for creating the actions within the user interface, like adding or removing criterion insertion lines as we see in figure 8, or starting the application's software execution.

3.4.Verification

By verification, we mean that the application software works, within acceptable time, in different areas, and with different data types. The application software was tested to give a meaningful output, a raster format map. It was also verified if the application software developed could process in feasible time (10 minutes maximum), a possible real problem. It was also verified to process alternatives within the study area, this will be described in the following chapter, and also to the Greater Cairo region in Egypt. The application software was also tested to take as input, raster and vector format data. Within the vector format data, it was also verified to accept point, line and polygon data type.

3.5.Study area and data

The above mention implementation of the application software used as a testing study area the Zambezi river Valley in Mozambique, described in this section. The study area is based on the project, environmental strategic evaluation of the multi-sectorial plan for land use management of the Zambezi river Valley. This area is part of the Zambezi river basin in the Republic of Mozambique, in figure 9. Mozambique is located in the Southeastern Africa, Mozambique's capital is Maputo in the south.

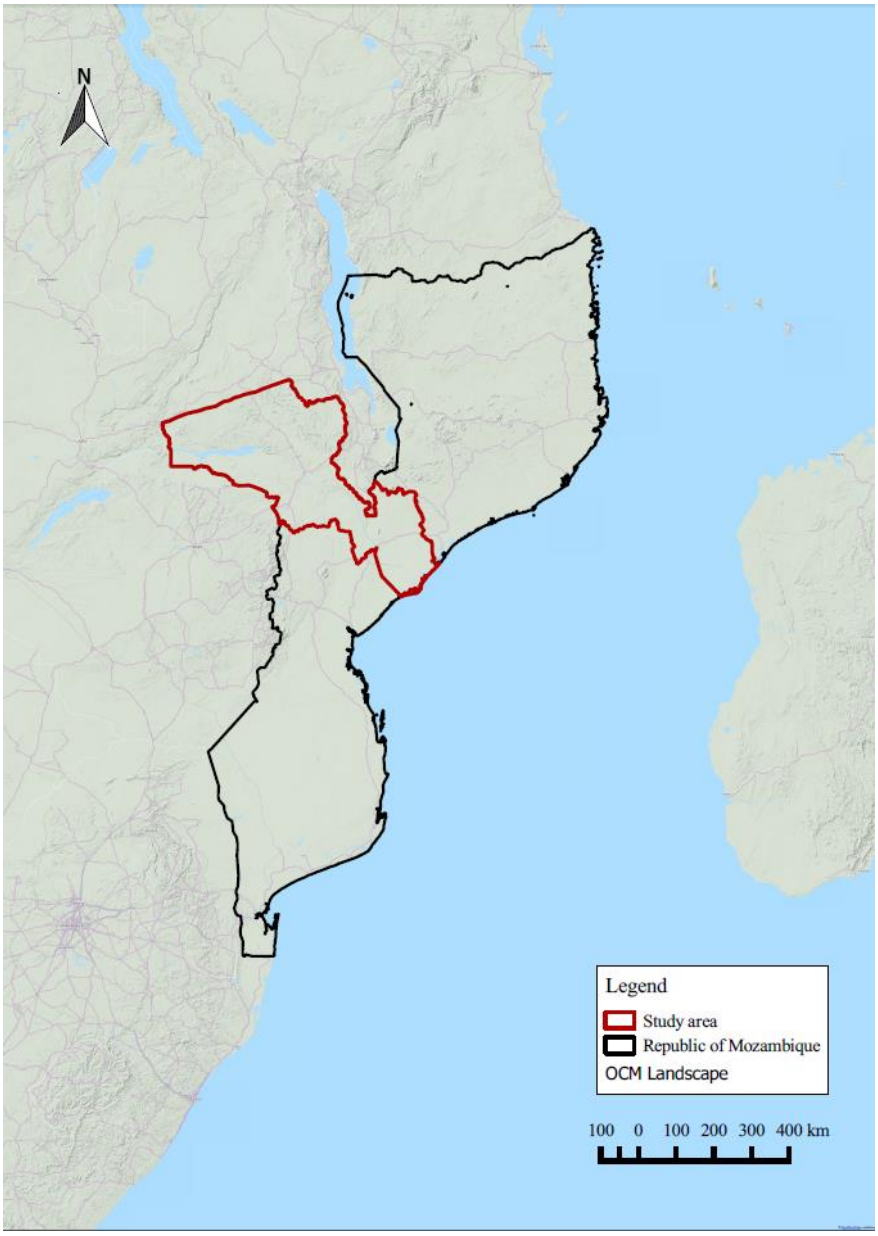

Figure 9 Context image of the study area.

The study area has 58 posts within 27 districts, that belong to 4 provinces, and the total area is 149 903km^2, in figure 10. The study area has one province capital, Tete. The main river in Mozambique is the Zambezi river, that crosses the study area. The Zambezi River is the fourth-longest river in Africa. Its source is located in Zambia, it passes by Angola, and defines the border of Zambia and Zimbabwe until it enters Mozambique. The river flows into the Mozambique Channel, part of the Indic ocean.

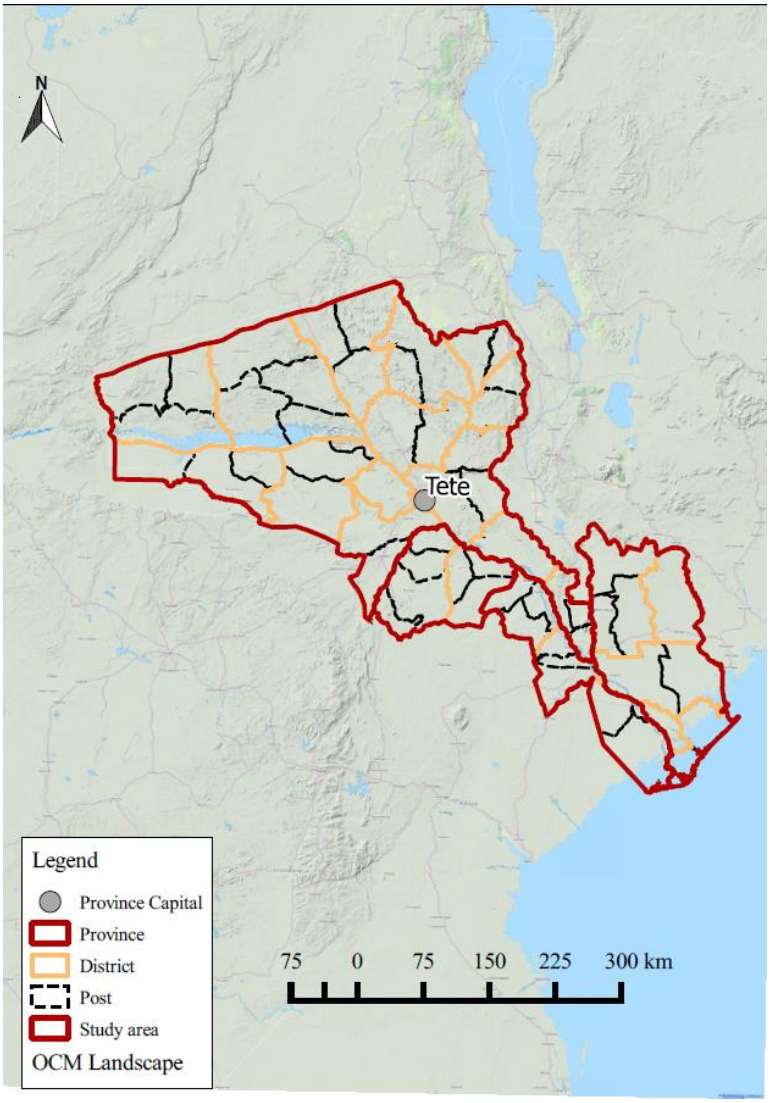

Figure 10 Administrative divisions of the study area.

The study area is characterized by 19 different types land cover. In its majority mixed shrub, around 46 500 km^2. Bare areas are the smallest areas found in the study area, with 0,18 km^2. The water bodies occupy 2996km^2 of the area. The reason for the large occupation of water bodies is the Chahora Bassa reservoir. This reservoir retains water from the Zambezi River, that flows into Mozambique from the Northwest border. To the south of the study area the predominant land cover is deciduous forest, while to the north is mixed shrub, in figure 11.

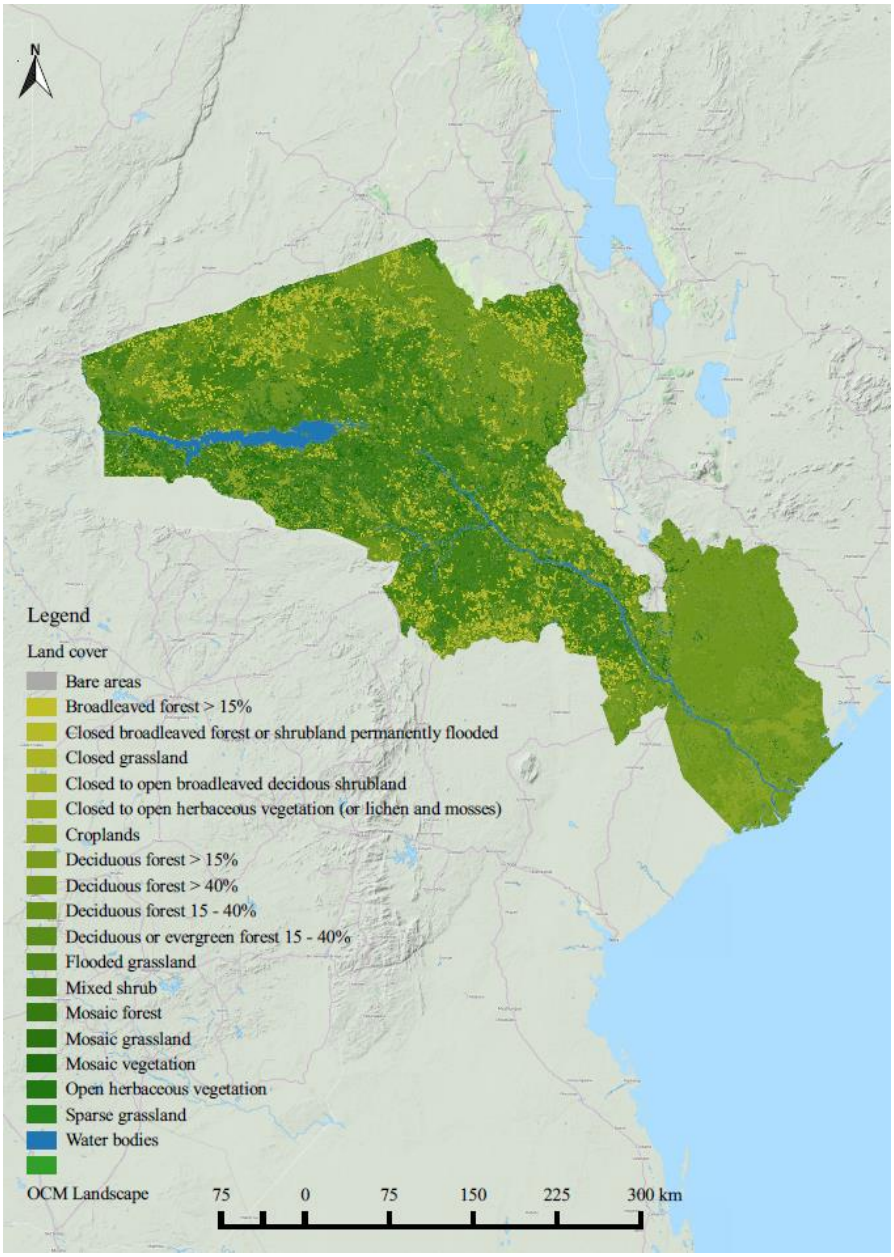

Figure 11 Land cover map of the study area.

The hydrology of the study area is characterized by a dense network of rivers that complement the Zambezi River. The reservoir of Chahora Bassa to the northwest of the region, collecting water from the Zambezi. The flooding areas are majorly located around the Chahora Bassa reservoir and the Zambezi River.

The study area is also characterized by its low slope percentage. The higher slope percentages, 5 to 20% are located mainly to the north of the Zambezi River. In the delta of the Zambezi River the areas are very flat, never reaching the 5%, visible in figure 12.

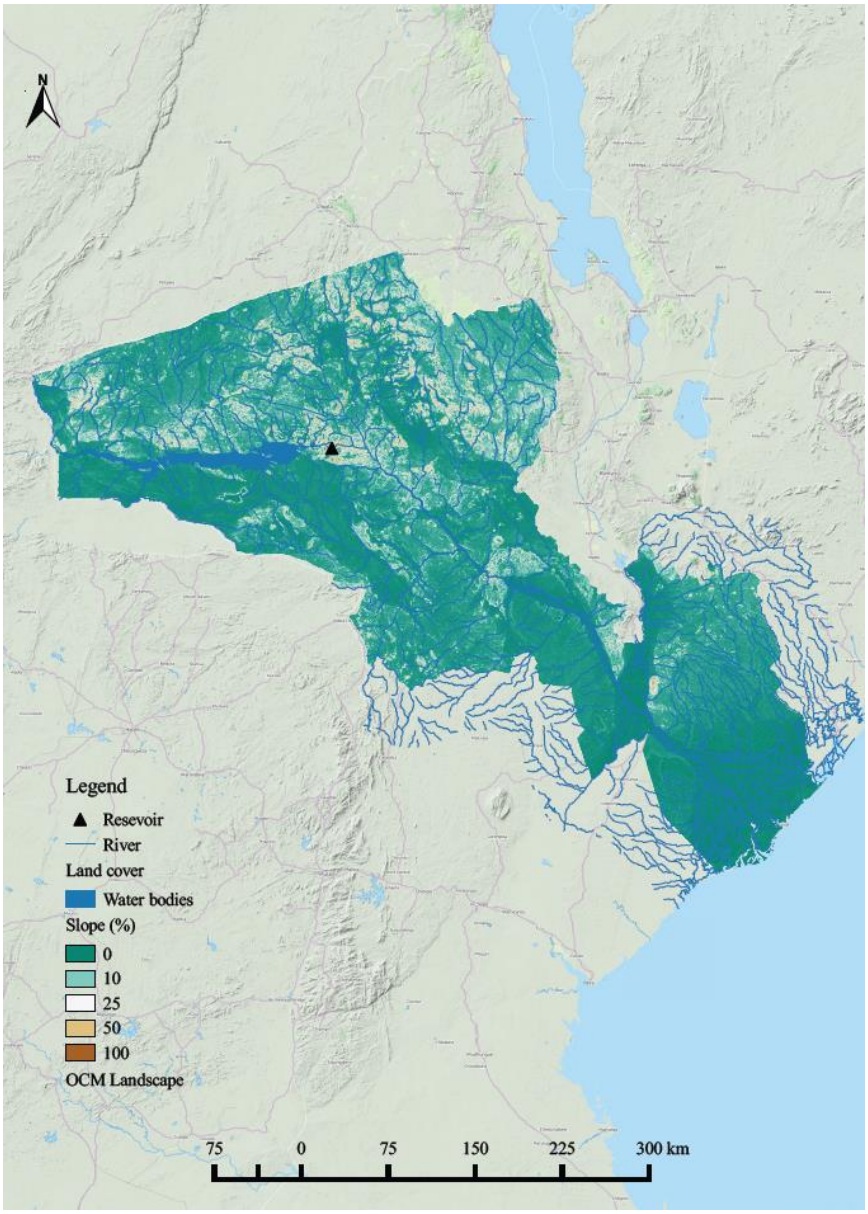

Figure 12 Rivers, reservoir and slope of the study area.

The number of villages in the study area is denser in the northeast and southeast regions. The south has a higher population density than the north. The city of Tete is another point of high population density. The districts with highest population density are Cidade de Tete with 489 inhabitants per km^2, and the second highest is Angonia in the northeast with 85 inhabitants per km^2. Which is the district with more population, with almost 375 000 inhabitants, followed by Morrumbala, with almost 360 000. There are 10 different ethno-linguistic groups in the study area.

The transport network of the study area can be divided into two, rail network and road network. The rail network that passes by the study area connects, Beira, from

the south, to Mutarara to Tete, and leaving Mozambique into Malawi. The roads network has 5 categories, primary, secondary, tertiary, vicinal, and not classified. Primary roads are paved, and they exist in the north of the study area in a cross-like shape, and in the south crossing the study area. This two roads are connected by a secondary road that goes along that section of the Zambezi River, and it is prolonged into the delta of the river.

The data available to develop the application software and the thesis, was provided by the web-based GIS developed for the environmental strategic evaluation of the multi-sectorial plan for land use management of the Zambezi River Valley project. The data cover 17 topics, from which hydrology, communications, geology, demography and socio-economic data, are a part of. The web-based GIS has a total of 130 geographic data layers available to display, within the 17 topics. Since the project is not complete, this number might change. The 130 layers can be separated into two data formats, vector and raster format. The vector format layers available in the web-based GIS, can be classified into point, line or polygon feature type layers, depending on what these represent.

Since the application software developed to integrate the web-based GIS project for the Zambezi River Valley, the data used in the development of the application was retrieved from the web-based GIS. This helped keeping the application's development focused on what data formats and types would be used in it. Although some data was obtained specifically for this project, some data came from other sources, like CENACARTA (CENACARTA, 2015). This institution, as part of the Agriculture department of Mozambique, has a geo-spatial data repository online, available for public download.

The data used in the development of the application software was:

- Land cover.
- Population density.
- Slope.
- Roads network.
- Railways network.
- Province capitals location.

The land cover data is in vector format and is of polygon features type. The slope layer used is in raster format, with a resolution of 90 meter cell, this was derived from the digital elevation model. The layers used from the population topic were vector format only, one polygon feature format and the other point feature format. The polygon feature presented the population distribution by villages, which was a result of a Theissen polygon algorithm on the point feature layer of villages. The point feature layer used also was the province capital where Tete was illustrated. The two other layers used were the roads network layer, and the rail network layer, both vector format and line feature type layers.

1. RESULTS AND DISCUSSION

This section will describe the final version of the application software, and present a practical example of the applications use. The final versions describes as follows. The user inputs for the application software are showed in table 11.

Input name	Description	Example
Data	The data file's name and location.	../../originals/mineral.shp
Criteria grade	The grading value for the criterion (1-10).	10
Type (code)	Type of data (vector/raster and categorical/continuous) and if the distance to objects is needed.	101
Field	If applied, specify the attribute from the shapefile to be used as criterion.	Density
Resolution	The cells resolution, using the unit of the target coordinate reference system.	2000
Extent	The extent of the raster image map in the target coordinate reference system, as x min, y min, x max, y max.	200200.279 7900257.049 906200.279 8452257.049
Target Coordinate Reference System	The EPSG code that the user wants for the output file.	EPSG:32736
Grading table files	The grading table files to assist the automatic pairwise comparison, as min max grade, or code 0 grade.	0 30 9 401 0 7

Table 11 User's input description.

The following description of the input, how it is stored and its integration in the whole process of the application software relates to the folder structure illustrated in figure 7.

The inputs, data, criteria grade, type and field are stored in a file called *alpha*, in the *in* folder. The inputs, resolution, extent and target coordinate system are in the file *geo* also in the *in* folder. The grading table files are numbered according to the order of the criteria specified in the *alpha* file, inside the *cat* folder.

The system design and structure was already discussed in the previous chapter. Still, it might be unclear the process flow of the application software, until the output is obtained. For a better illustration please review the flowcharts of figures 2, 3, 4 and 5, while following the succeeding explanation of a concrete example.

An example of the application's use for the study area, is the hypothetical reallocation of the Tete city. The data chosen, and the grading tables applied in this example, should not be considered expert's advice. Some guidance is given on the problem, the new location of the city should not be too far from the existing city, and mining is a very big part of the employment in this region. Organizing the problem into a hierarchy, the main goal would be to obtain a suitable location for the new Tete within the study area.

The data given by the decision maker are: the population density, the proximity to roads network, the proximity to railway network, the proximity to the existing Tete, the land cover, the slope and mining concessions. This are mentioned in the *alpha* file in the *in* folder (figure 14).

The population density was chosen because, the construction of a city requires large areas, and to reallocate many people is harder than reallocating no people. The proximity to roads network and rail networks, because good accesses to main roads or railways will ease the commute of the inhabitants around the city, bring more commerce to the city, etc. The proximity to the existing Tete, because people living and working in the existing city would not have to find a new place to live while working in the new Tete. The land cover, because we would not want to build the new city on top of water, and want to avoid deforestation. The slope, because of cost-efficiency, to flatten down big areas is time consuming and expensive. The proximity to mining concessions, to provide the population with job opportunities. The alternatives should cover the entire study area. The resolution of the output raster could be $9km^2$ (3km by 3km) because cities occupy large areas. A representation of the hierarchy is given in figure 12.

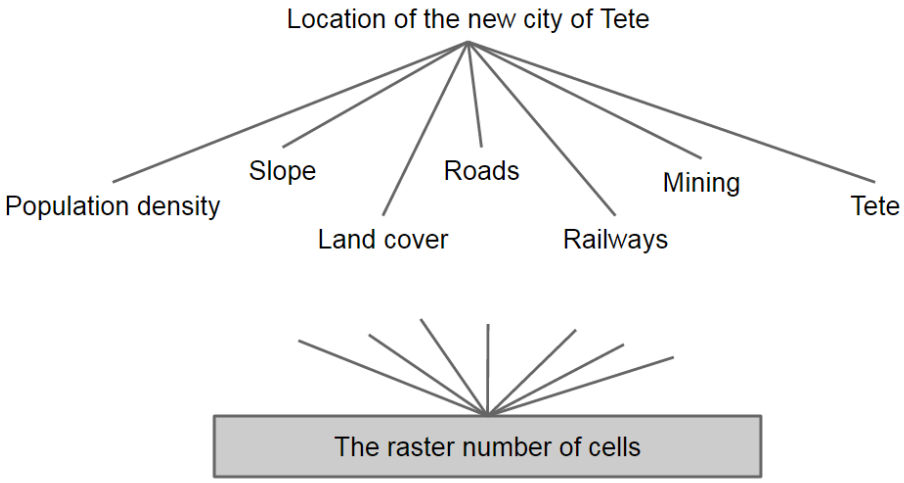

Figure 13 Hierarchic structure of the problem.

This data is read by the application software, and is stored in memory. Then for each of these criteria/maps mentioned in the *alpha* file (figure 14) the application will read if it is in raster or vector format. This distinction is done using the *type code*, a 3 digit binary code. If the data is in raster format, like the slope data, the application will use the GDAL *gdalwarp* program to perform a resample of the raster, according to the resolution, extension and target coordinate reference system mentioned by the user in the input *geo* file (figure 15). The resampled raster will be stored in the *tmp* folder. In case the data is in vector format, like the roads network and population density, it is reprojected according to the given coordinate reference system, using the GDAL *ogr2ogr* program, and also stores these in the *tmp* folder.

alpha

```
1 ../../originals/Rede_ferroviaria.shp Rede_ferroviaria 6 110 NA
2 ../../originals/Concessoes_minerais.shp Concessoes_minerais 8 110 NA
3 ../../originals/Capitais_de_Provincia.shp Capitais_de_Provincia 7 110 NA
4 ../../originals/Rede_viaria.shp Rede_viaria 5 110 NA
5 ../../originals/Habitantes_por_aldeia_thiessen.shp Habitantes_por_aldeia_thiessen 8 111 DENSIDADE
6 ../../originals/Ocupacao_do_solo.shp Ocupacao_do_solo 10 101 OCUP1
7 ../../originals/Declives.tif NA 7 011 NA
```

Figure 14 Alpha file used in the example, reallocation of Tete city.

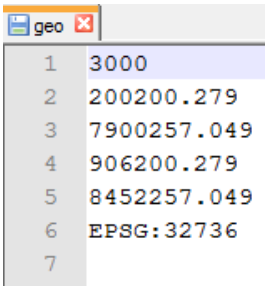

Figure 15 Geo file used in the example, reallocation of Tete city.

In the *geo* file in figure 15, from top to bottom, the cell size of the desired raster output, the x minimum, y minimum, x maximum, y maximum, and the coordinate reference system. The *alpha* file in figure 14, specifies the criterion land cover to be the most important for the decision, giving it a grade of 10, followed by the mining concessions, 8, the distance to Tete and slope of the area are thought to have the same importance, 7, which is still more than railway network and the roads network, 5.

After the reprojection, the application software will use the *gdal_rasterize* program to rasterize the vector format maps according to the *geo* file (figure 15) and the input *field* mentioned in the *alpha* file. The resulting rasters will be store in the *tmp* folder as well. The files in the *tmp* folder no longer needed, are erased to save memory space, and to avoid conflicts between files. With all criteria in raster format, the application software will verify the *type code* again, one by one, to inspect if the user required that any of the criteria depended on distances. In this example, the criteria, roads, railways, and Tete need this. The application software will use the *gdal_proximity.py* program to calculate the distance, and write these distances as the new cell values.

The last inputs needed to start running the application software are the grading tables' files, in the *cat* folder. They are named with the number of the order, from which they were mentioned in the *alpha* file, 1 for the railway, 2 for the mining, and so on. The grading tables are illustrated in table 12 and 13. The criteria represented mentioned on the top, and each table will have 3 columns. The first is the minimum value of the intervals of the criterion, the second is the maximum value of the corresponding interval, and the third is the grading given to that interval. The categorical data uses only the minimum's column for the categories code.

Railway (m)			Mining (m)			Tete (m)			Roads (m)		
0	5000	10	0	3000	10	0	10000	2	0	2000	10
5000	10000	8	3000	3000	8	10000	30000	10	2000	5000	8
10000	20000	7	6000	6000	7	30000	100000	8	5000	10000	6
20000	50000	5	10000	10000	5	100000	150000	7	10000	0	5
50000	0	3	20000	0	2	150000	0	1			

Table 12 Grading tables used in the example, reallocation of Tete city.

Population			Land Cover (gridcode)			Slope (%)		
0	100	10	16	0	8	0	2	10
100	200	7	15	0	8	2	5	8
200	300	6	14	0	8	5	10	3
300	400	3	…	…	…	…	…	…
400	0	1	200	0	10	45	0	1

Table 13 Grading tables used in the example, reallocation of Tete city.

When all raster images are ready for reclassification, the application software will read the corresponding grading table from the *cat* folder, and reclassify the raster values. The reclassified rasters are stored in the *tmp* folder, ready for the AHP computation.

At this point, all raster maps have the values ranging from 1 through 10, or 0 to exclude cells from the process (apply masks). Before the AHP computation, it is important to understand that, if the raster images have the same, cell size, extent and coordinate reference system, then they represent the same area overlaying perfectly each cell. This means the cell's location (x and y position) can be used to identify the cell and assume the same for all raster images, see figure 16.

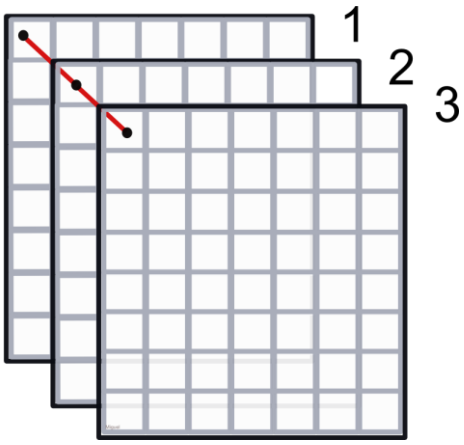

Figure 16 Raster overlay illustration.

The AHP computation starts when all raster images are in the standard format, and the perfect overlay of the rasters is assumed. This principles are crucial for the AHP computation to work. The values of the cells of each raster are organized in a table, where each column is a criterion and each row is the cell's location (and identification).

The AHP computation continues, as mentioned in the system design chapter, creating the pairwise comparison matrices, computing the logarithmic least squares, and obtaining the final ranking of the alternatives. It should be mentioned that, according to Saaty (1990) the pairwise comparison matrices, should follow the consistency condition.

The matrices have to have a consistency ratio lower than 0.1, to be considered acceptable. The Consistency Ratio (CR) is obtained by dividing the Consistency Index (CI) by an Index of Randomness (IR). The CI of a matrix is as shown in equation 10

$$CI = (\lambda max - n)/(n-1) \tag{11}$$

Where *n* is the matrix dimension, and

$$\lambda \max = \sum priority\ row \tag{12}$$

$$priority\ row = \sum column\ values * priority\ vector \tag{13}$$

And the priority vector is the normalized vector obtained with the LLSM for each matrix.

A matrix	1	2	…	*n*	Priority vector (pv)
1	a_{11}	a_{12}	…	a_{1n}	$LLSM_{1j} / \sum LLSM$
2	a_{21}	a_{22}	…	a_{2n}	…
…	…	…	…	…	
n	a_{n1}	a_{n2}	…	a_{nn}	$LLSM_{nj} / \sum LLSM$
$\sum_{j=1} a_{ij}$	$\sum a_{i1}$		…	$\sum a_{ij}$	
Priority row	$\sum a_{i1}$*pv		…	$\sum a_{ij}$*pv	λmax

Table 14 Example matrix with priority vector and row.

The IR is a mean of the CI of 500 random generated matrices for *n* dimensions (Saaty, 1990). Saaty (1987) only generated this index for matrices of *n* =1, 2, … , 10. Since the matrices used in this example have *n* = 36614, we developed a script to generate the 500 random matrices with *n* = 36614. After several tests of the script, the IR's generated were always lower than the ones Saaty (1987) obtained, in table 15 in annex. With lower IR values the CR becomes harsher on the matrices. Even with this increased constraint all matrices generated by the application software were consistent, with the highest CR being 0.011 for the slope criterion.

The output is the ranking of the raster cells. For simplification, these values are normalized to range from 0 to 1. Providing an easier visualization of the output suitability raster map, for the decision maker. After they are normalized, the values are written into a raster map, with the extent, resolution and coordinate reference system the decision maker provided in the *geo* file. The resulting raster map of this example, the reallocation of Tete city is showed in figure 17.

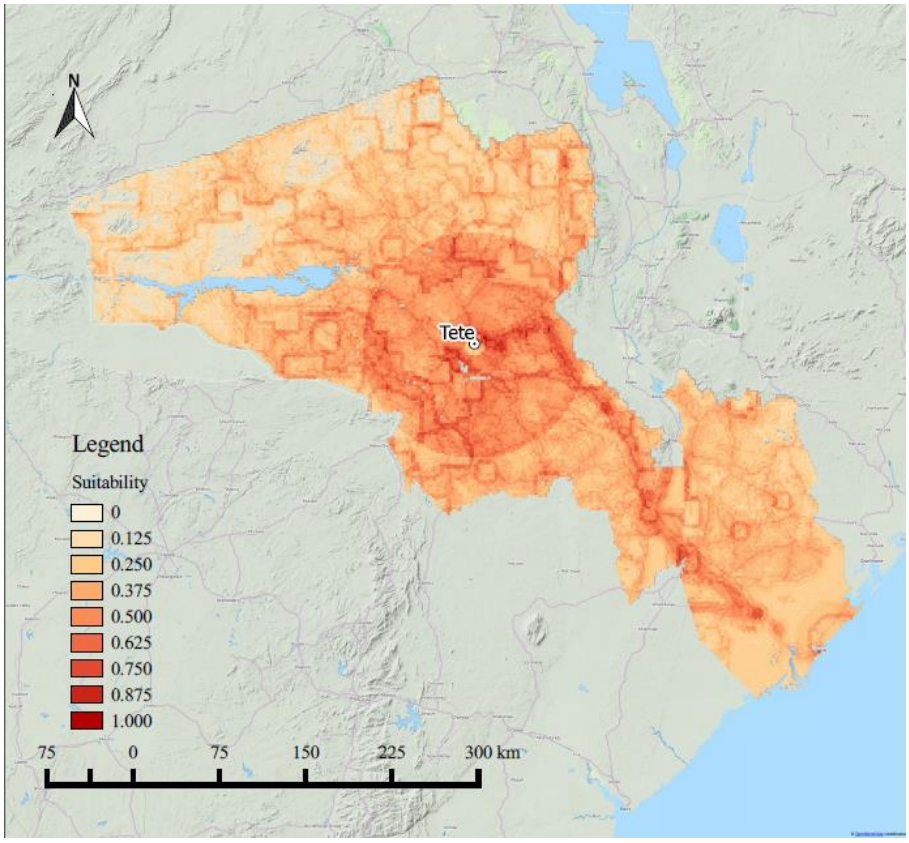

Figure 17 Suitability map of the example, reallocation of Tete city.

In the resulting suitability map, of the application software developed, according to the criteria above mentioned and the grading specified, the most suitable areas for the reallocation of the Tete city are the areas in red color. These areas are located mainly around the railway network, the roads network and within the 100km radius of the existing Tete city. Where all these criteria overlap, East of Tete, is one of the largest clusters of highly suitable areas, for the reallocation of Tete city.

This suitability map is intended to help decision makers select the area for the construction of the new Tete city. It should be noted, the output of the application software is always dependent on the inputs given by the decision maker. Furthermore, this suitability map is a translation of the criteria chosen, and grades given. What was not considered as input will never be denoted in the output.

2. CONCLUSIONS

We can conclude the successful implementation of the AHP in a spatial support decision system for land use management. It can also be conclude the application software's flexibility data format input. Consequence of the geoprocess, the possibility of input raster, as well as, vector format data. Also a consequence of the geoprocess the application's software flexibility on the number of alternatives. The use of thousands of alternatives for the decision being done, is also consequence of the AHP computational optimization. Attached to this implementation, the automatic pairwise comparison method was successfully applied. The user can also define custom made data intervals, exclusion areas and best values for the comparisons, giving the user full control of the decision support system. The alternative to the suggested eigenvalue's methods, the logarithmic least squares methods was also a successful implementation.

The application software developed can until now intake around 50 000 alternatives. This limit is set by the machine's or operating system's limitations. The application software also does not have until now a user interface. This could be implemented to facilitate the user's interaction with the SDSS in a user-friendly way.

Bibliography

ATTI, B. R. (2005). - NOTE - The New Geography : Geographical Knowledge and GIS. *Netcom*, *19*, 119–124.

BEIGBABAYI, B., MOBARAKI, M. A., BRANCH, M., & BRANCH, A. (2012). Using AHP Modeling and GIS to Evaluate the Suitability of Site with Climatic Potential for Cultivation of Autumn Canola in Ardabil Province, *3*(5), 2307–2317.

CARVER, S. J. (1991). Integrating multi-criteria evaluation with geographical information systems. *International Journal of Geographical Information Systems*, *5*(January 2015), 321–339. doi:10.1080/02693799108927858

GOODCHILD, M. F. (1992). Geographical information science. *International Journal of Geographical Information Systems*, *6*(McHarg 1969), 31–45. doi:10.1080/02693799208901893

GOODCHILD, M. F. (1993). The state of GIS for environmental problem-solving. *Environmental Modeling with GIS*.

JANKOWSKI, P., Fraley, G., & Pebesma, E. (2014). An exploratory approach to spatial decision support. *Computers, Environment and Urban Systems*, *45*, 101–113. doi:10.1016/j.compenvurbsys.2014.02.008

LIN, C., CHOY, K. L., HO, G. T. S., LAM, H. Y., PANG, G. K. H., & CHIN, K. S. (2014). A decision support system for optimizing dynamic courier routing operations. *Expert Systems with Applications*, *41*, 6917–6933. doi:10.1016/j.eswa.2014.04.036

LONGLEY, P., GOODCHILD, M., MAGUIRE, D., & RHIND, D. (2010). *Geographic Information Systems and Science* (3rd. ed., p. 539). Wiley.

MALCZEWSKI, J. (1999). *GIS and Multiriteria Decision Analysis* (p. 392). John Wiley & Sons, Inc.

MALCZEWSKI, J. (2006a). GIS- based multicriteria decision analysis: a survey of the literature. *International Journal of Geographical Information Science*, *20*(7), 703–726. doi:10.1080/13658810600661508

MALCZEWSKI, J. (2006b). GIS- based multicriteria decision analysis: a survey of the literature. *International Journal of Geographical Information Science*, *20*(January 2015), 703–726. doi:10.1080/13658810600661508

MARINONI, O. (2004). Implementation of the analytical hierarchy process with VBA in ArcGIS, *30*, 637–646. doi:10.1016/j.cageo.2004.03.010

MASSEI, G., ROCCHI, L., & PAOLOTTI, L. (2013). MCDA-GIS integration: an application in GRASS GIS 6.4. *Journal of Spatial Information Science*, (Dm), 1–24. Retrieved from http://www.josis.org/index.php/josis/article/viewArticle/118

SAATY, R. W. (1987). The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling*, *9*(3-5), 161–176. Retrieved from http://www.sciencedirect.com/science/article/pii/0270025587904738

SAATY, T. L. (1990). How to make a decision: The analytic hierarchy process. *European Journal of Operational Research*, *48*(1), 9–26. doi:10.1016/0377-2217(90)90057-I

SAATY, T. L. (2000). *Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process* (p. 458). RWS Publications. Retrieved from http://cpe.njit.edu/dlnotes/MIS645/MIS648_AFrameWorkfortheDevelopmentofDecisionSupportSystems.pdf

SAATY, T. L., & VARGAS, L. G. (1984). Comparison of eigenvalue, logarithmic least squares and least squares methods in estimating ratios. *Mathematical Modelling*, *5*(5), 309–324. doi:10.1016/0270-0255(84)90008-3

SAFIAN, E. M., & NAWAWI, A. (2012). Combining AHP with GIS in the evaluation of locational characteristics quality for purpose-built offices in Malaysia, (39546). Retrieved from http://ideas.repec.org/p/pra/mprapa/39546.html

SILVA, S., ALÇADA-ALMEIDA, L., & DIAS, L. C. (2014). Development of a Web-based Multi-criteria Spatial Decision Support System for the assessment of environmental sustainability of dairy farms. *COMPUTERS AND ELECTRONICS IN AGRICULTURE*, *108*, 46–57. doi:10.1016/j.compag.2014.06.009

SUGUMARAN, R., & DEGROOTE, J. (2011). *Spatial Decision Support Systems: Principles and Practices* (p. 507). CRC Press.

TOMLINSON, R. F. (1962). An Introduction to the use of Elecronis Computers in the Storage, Compilation and Assessment of Natural and Economic Data for the Evaluation of Marginal Lands.

YING, X., ZENG, G.-M., CHEN, G.-Q., TANG, L., WANG, K.-L., & HUANG, D.-Y. (2007). Combining AHP with GIS in synthetic evaluation of eco-environment quality—A case study of Hunan Province, China. *Ecological Modelling*, *209*(2-4), 97–109. doi:10.1016/j.ecolmodel.2007.06.007

Cenacarta.com, (2015). CENACARTA - Home. [online] Available at: http://www.cenacarta.com/ [Accessed 10 Feb. 2015].

Gdal.org, (2015). GDAL: GDAL - Geospatial Data Abstraction Library. [online] Available at: http://www.gdal.org/ [Accessed 10 Feb. 2015].

R-project.org, (2015). The R Project for Statistical Computing. [online] Available at: http://www.r-project.org/ [Accessed 10 Feb. 2015].

Qt-project.org, (2015). QtWhitepaper | Qt Wiki | Qt Project. [online] Available at: https://qt-project.org/wiki/QtWhitepaper [Accessed 10 Feb. 2015].

Annex

Intensity of importance on an absolute scale	Definition	Explanation
1	Equal importance	Two activities contribute equally to the objective
3	Moderate importance of one over another	Experience and judgment strongly favor one activity over another
5	Essential or strong importance	Experience and judgment strongly favor one activity over another
7	Very strong importance	An activity is strongly favored and its dominance demonstrated in practice
9	Extreme importance	The evidence favoring one activity over another is of the highest possible order of affirmation
2,4,6,8	Intermediate values between the two adjacent judgments	When compromise is needed
Reciprocals	If activity *i* has one of the above numbers assigned to it when compared with activity *j*, then *j* has the reciprocal value when compared with *i*	
Rational	Ratios arising from the scale	If consistency were to be forced by obtaining n numerical values to span the matrix

Table 15 Saaty's (1990) fundamental scale of comparison.

n	Saaty (1987) IR	LLSM IR	Eigenvalue IR
3	0.58	0.57	0.46
4	0.9	0.78	0.82
5	1.12	0.97	1
6	1.24	1.07	1.15
7	1.32	1.15	1.27
8	1.41	1.22	1.29
9	1.45	1.25	1.35
10	1.49	1.32	1.38

Table 16 Index of randomness by Saaty (1987), and generated.

The application software code is as follows, in figure 7 the structure is illustrated.

The *geoahp.h* in the *include* folder.

```
/*
    this is the header that defines the type, order and names of the var
iables in the input files.

 */
#ifndef _geoahp3_h
#define _geoahp3_h

#define MAX_NUM_CHARS       10000

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct input {
    // geo file data
    char *cellsize;     /* the size of the raster cells */
    char *xmin;         /* the xmin of the geographic envelope */
    char *ymin;         /* the ymin of the geographic envelope */
    char *xmax;         /* the xmax of the geographic envelope */
    char *ymax;         /* the ymax of the geographic envelope */
    char *epsg;         /* the EPSG code of the study area */
    // alpha file data
    int ncrit;          /* the number of criteria in the alpha file */
    struct {
        char *name;     /* the path and name of the layer */
        char *lyr;      /* the layer's name */
        int weight;     /* the weight from 1-10 of this layer */
        int type;       /* the type of this layer according to README fi
le */
        char *field;    /* the field to be used for rasterization */
    }  *layer;
    // grading tables files information
    struct {
        int nrows;      /* the number of rows in the catfile */
        int *cat;       /* the content of the catfile nrows x 3 */
    } *catfile;
};


/*
    Name: read_input
    Description: it defines the structure of the input files (in/alpha,
geo)
    Parameters:
        . alpha is the file where the alphanumeric information of each l
ayer is;
        . geo is the file where the geographic information of each layer
 is;
    Require: alpha and geo files must conform the specifications on the
README file.
    Returns: a pointer to a structure input or NULL if memory is not ava
ilable.
 */
struct input * read_input(const char *geo,const char *alpha);

/*
    Name: free_input
```

```
52.     Description: frees the memory used in the read_input
53.     Parameters:
54.         . input structure
55.     Returns: no return.
56.  */
57. void free_input(struct input *x);
58.
59. /*
60.     Name: syscall
61.     Description: system call to call system functions.
62.     Parameters:
63.         . argc integer number;
64.         . argv character value;
65.     Returns: calls for functions of the system like the command line.
66.  */
67. int syscall(int argc, char *argv[]);
68.
69. /*
70.     Name: israster
71.     Description: verifies if the data is in raster format using the type
     input
72.     in the alpha file
73.     Parameters:
74.         . infile attributes of the input data files;
75.         . i integer number;
76.     Returns: 1 if the type code specifies data is in raster format.
77.  */
78. int israster(struct input *infile, int i);
79.
80. /*
81.     Name: distreq
82.     Description: verifies if the data needs the calculation of proximity
     using the
83.     type input in the alpha file
84.     Parameters:
85.         . infile attributes of the input data files;
86.         . i integer number;
87.     Returns: 1 if the type input is specified the need to calculate prox
    imity.
88.  */
89. int distreq(struct input *infile, int i);
90.
91. /*
92.     Name: iscontinuous
93.     Description: verifies if the data is continuous or categorical.
94.     Parameters:
95.         . infile attributes of the input data files;
96.         . i integer number;
97.     Returns: 1 if in the type input is specified as continuous.
98.  */
99. int iscontinuous(struct input *infile, int i);
100.
101.       /*
102.           Name: reclassify
103.           Description: reclassifies the raster cells according to the r
    espective catfile
104.           files
105.           Parameters:
106.               . infile attributes of the input data files;
107.               . idx the index of each cell of the rasters;
108.           Returns: returns a raster with values varying between 1 and 1
    0 according to
109.           the respective catfile.
```

```
110.        */
111.       void reclassify(struct input *infile, int idx);
112.
113.       /*
114.           Name: ahp
115.           Description: computes the pairwise comparison matrix and it's
    priorities
116.           Parameters:
117.               . idx the index of the non-nodata cells;
118.               . tdt the 1 to 10 value of each cell before comparison;
119.               . nidx the number of non-nodata cells (length of idx);
120.               . ncrit the number of criteria;
121.           Returns: the ranking of the alternatives or criteria.
122.        */
123.       float * ahp(int *idx, float *tdt, int nidx, int ncrit);
124.
125.       /*
126.           Name: geoahp
127.           Description: opens each raster from the geoprocess, extracts
   the non-nodata cells,
128.           calls the ahp function, multiplies the priorities of the alte
   rnatives with the
129.           criteria priority vector, normalizes the result, and writes t
   he output raster file.
130.           Parameters:
131.               . infile attributes of the input data files;
132.           Returns: the suitability raster file.
133.        */
134.       void geoahp(struct input *infile);
135.
136.       #endif
```

The *ahp.c* file, in the *src* folder:

```
1.  #include <math.h>
2.  #include <gdal.h>
3.  #include <cpl_conv.h>
4.  #include "geoahp.h"
5.
6.
7.
8.  void geoahp(struct input *infile) {
9.
10.     GDALAllRegister();
11.
12.     int nrows, ncols;
13.     double geotransf[6];    // Geotransformation parameters according to
     GDAL specifications (http://www.gdal.org/gdal_datamodel.html)
14.     char rastername[100];
15.     const char *proj;       // WKT according to GDAL specifications (htt
    p://www.gdal.org/gdal_datamodel.html)
16.     float *parray;
17.     float *tdt;             // Table holding raster data (nrows*ncols x
    ncrit)
18.
19.     int ncrit = infile->ncrit;
20.
21.     // For each criterion
22.     for (int i=0; i<ncrit; ++i) {
23.
24.         // Open raster file
25.         rastername[0] = '\0';
26.         sprintf(rastername, "../tmp/%d.tif", i+1);
27.         GDALDatasetH hDataset = GDALOpen(rastername,GA_ReadOnly);
28.
29.         // Assuming that each file has the same geo properties
30.         if (i == 0) {
31.             ncols = GDALGetRasterXSize(hDataset);
32.             nrows = GDALGetRasterYSize(hDataset);
33.             proj = GDALGetProjectionRef(hDataset);
34.             GDALGetGeoTransform(hDataset, geotransf);
35.             parray = CPLMalloc(ncols*nrows*sizeof*parray);
36.             tdt = malloc(nrows*ncols*ncrit*sizeof*tdt);
37.         }
38.         // Read raster data
39.         GDALRasterBandH hBand = GDALGetRasterBand(hDataset,1);
40.         GDALRasterIO(hBand, GF_Read, 0, 0, ncols, nrows, parray, ncols,
    nrows, GDT_Float32, 0, 0);
41.
42.         // Compute raster table
43.         for (int j=0; j<nrows; ++j) {
44.             for (int k=0; k<ncols; ++k) {
45.                 int l = j*ncols+k; // pixel's linear index
46.                 tdt[l*ncrit+i] = parray[l];
47.             }
48.         }
49.         GDALClose(hDataset);
50.     }
51.
52.     CPLFree(parray);
53.
54.     // Get the non-nodata pixels indices
55.     int nidx = 0;
56.     int *idx = malloc(nrows*ncols*sizeof*idx);
```

```
57.     for (int i=0; i<ncols*nrows; ++i) {
58.         int stay = 1;
59.         for (int j=0; j<ncrit && stay; ++j)
60.             if (tdt[i*ncrit+j] == 0) stay = 0;
61.         if (stay) idx[nidx++] = i;
62.     }
63.
64.     // Calculate the alternatives ranking
65.     float *rnk = ahp(idx,tdt,nidx,nrows*ncols,ncrit);
66.
67.     free(tdt);
68.
69.     // Calculate the criteria ranking
70.     int *cidx = malloc(ncrit*sizeof*cidx);
71.     tdt = malloc(ncrit*sizeof*tdt);
72.     for (int i=0; i<ncrit; ++i) {
73.         cidx[i] = i;
74.         tdt[i] = (float)infile->layer[i].weight;
75.     }
76.
77.     float *crnk = ahp(cidx,tdt,ncrit,ncrit,1);
78.
79.     free(cidx);
80.     free(tdt);
81.
82.     // Matrix x vector product
83.     float *suit = malloc(nidx*sizeof*suit);
84.     for (int i=0; i<nidx; ++i) {
85.         suit[i] = 0.0;
86.         for (int j=0; j<ncrit; ++j)
87.             suit[i] += rnk[i*ncrit+j] * crnk[j];
88.     }
89.
90.     // Normalization
91.     float smin = suit[0];
92.     float smax = suit[0];
93.     for (int i=1; i<nidx; ++i){
94.         if (suit[i] < smin) smin = suit[i];
95.         if (suit[i] > smax) smax = suit[i];
96.     }
97.
98.     for (int i=1; i<nidx; ++i){
99.         suit[i] = (suit[i]-smin) / (smax-smin);
100.        }
101.
102.        free(rnk);
103.        free(crnk);
104.
105.        // Compose output
106.        float *out = calloc(nrows*ncols,sizeof*out);
107.        for (int i=0; i<nidx; ++i) {
108.            out[idx[i]] = suit[i];
109.        }
110.
111.        free(suit);
112.        free(idx);
113.
114.        // Write output
115.        rastername[0] = '\0';
116.        strcpy(rastername, "../out/out.tif");
117.
118.        GDALDriverH driver = GDALGetDriverByName("GTiff");
```

```
119.            GDALDatasetH ds = GDALCreate(driver, rastername, ncols, nrows
    , 1, GDT_Float32, 0);
120.
121.            GDALSetProjection(ds, proj);
122.            GDALSetGeoTransform(ds, geotransf);
123.
124.            GDALRasterBandH hband = GDALGetRasterBand(ds,1);
125.            GDALRasterIO(hband, GF_Write, 0, 0, ncols, nrows, out, ncols,
     nrows, GDT_Float32, 0, 0);
126.
127.            GDALClose(ds);
128.
129.
130.            free(out);
131.        }
132.
133.        float * ahp(int *idx, float *tdt, int nidx, int ncrit) {
134.
135.            char filename[100];
136.
137.            // Calculate ranking for each field (criterion)
138.            float *mat = malloc(nidx*nidx*sizeof*mat);
139.
140.            // The output
141.            float *rnk = malloc(nidx*ncrit*sizeof*rnk);
142.
143.            // For each criterion
144.            for (int k=0; k<ncrit; ++k) {
145.                // Pairwise matrix relation between alternatives
146.                for (int i=0; i<nidx; ++i) {
147.                    for (int j=i; j<nidx; ++j) {
148.                        if (i != j) {
149.                            float xi = tdt[idx[i]*ncrit+k];
150.                            float xj = tdt[idx[j]*ncrit+k];
151.                            float s = 0.888888889 * ABS(xi-xj) + 1;
152.                            if (xi > xj) {
153.                                mat[i*nidx+j] = s;
154.                                mat[j*nidx+i] = 1/s;
155.                            } else {
156.                                mat[i*nidx+j] = 1/s;
157.                                mat[j*nidx+i] = s;
158.                            }
159.                        } else {
160.                            mat[i*nidx+j] = 1.0;
161.                        }
162.                    }
163.                }
164.                // Ranking computing
165.                float sum = 0.0;
166.                for (int i=0; i<nidx; ++i) {
167.                    float s = 0.0;
168.                    for (int j=0; j<nidx; ++j)
169.                        s += log(mat[i*nidx+j]);
170.                    rnk[i*ncrit+k] = exp(s/nidx);
171.                    sum += rnk[i*ncrit+k];
172.                }
173.                // Normalization
174.                for (int i=0; i<nidx; ++i)
175.                    rnk[i*ncrit+k] /= sum;
176.            }
177.
178.            free(mat);
179.            return rnk;
```

```
180.        }
```

The *main.c* file in the *src* folder:

```
1.  #include "geoahp.h"
2.
3.
4.  int main(){
5.
6.      char str[MAX_NUM_CHARS];
7.
8.      // Read the input from the user
9.      struct input *infile = read_input("../in/geo","../in/alpha");
10.
11.     // For each criterion in the alpha file
12.     for (int i=0; i<infile->ncrit; ++i){
13.
14.         // Is the i-th file a Raster?
15.         if (israster(infile,i)) {
16.             printf("Resampling Raster\n");
17.             // Resample:
18.             int argc = 18;
19.             sprintf(str, "../tmp/t%d.tif", i+1);
20.             char *argv[] = {"gdalwarp -q -t_srs ", infile->epsg, " -
    te ", infile->xmin, " ", infile->ymin, " ", infile->xmax, " ", infile-
    >ymax,
21.                             " -dstnodata -9999 -tr ", infile-
    >cellsize, " ", infile->cellsize,"  ", infile-
    >layer[i].name, " ", str};
22.             syscall(argc, argv);
23.         }
24.         else {
25.             printf("Projecting layer\n");
26.             // Project data
27.             int argc = 6;
28.             strcpy(str, "../tmp/a");
29.             char *argv[] = {"ogr2ogr -q -f \"ESRI shapefile\" -
    t_srs ", infile->epsg," ", str, " ", infile->layer[i].name};
30.             syscall(argc, argv);
31.
32.             // Does the criteria needs to measure distances?
33.             if (distreq(infile,i)) {
34.                 printf("Rasterizing layer\n");
35.                 // Rasterize and Clip
36.                 char auxs[100];
37.                 int argc = 18;
38.                 sprintf(auxs, "../tmp/tt%d", i+1);
39.                 char *argv[] = {"gdal_rasterize -burn 1 -q -
    te ", infile->xmin, " ", infile->ymin, " ", infile->xmax, " ", infile-
    >ymax,
40.                                 " -tr ", infile->cellsize, " ", infile-
    >cellsize," -a_srs ", infile->epsg, " -l ", infile-
    >layer[i].lyr, " ../tmp/a/*.shp ", auxs};
41.                 syscall(argc, argv);
42.                 printf("Calculating distances for raster\n");
43.                 // Calculate distances
44.                 char auxz[100];
45.                 sprintf(auxz, "../tmp/t%d.tif", i+1);
46.                 int argk = 5;
47.                 char *argx[] = {"gdal_proximity.py ", auxs, " ", auxz, "
     -distunits GEO -q"};
48.                 syscall(argk, argx);
```

```
49.                 system("rm -r ../tmp/tt*");
50.             }
51.             else {
52.                 printf("Rasterizing layer\n");
53.                 // Rasterize
54.                 char auxs[100];
55.                 int argc = 20;
56.                 strcpy(str, "../tmp/a/*.shp");
57.                 sprintf(auxs, "../tmp/t%d.tif", i+1);
58.                 char *argv[] = {"gdal_rasterize -q -a ", infile->layer[i].field, " -te ", infile->xmin, " ", infile->ymin, " ", infile->xmax, " ",
59.                                 infile->ymax, " -tr ", infile->cellsize, " ", infile->cellsize, " -a_srs ", infile->epsg, " -l ", infile->layer[i].lyr, " ../tmp/a/*.shp ", auxs};
60.                 syscall(argc, argv);
61.             }
62.
63.             // Delete folder a in tmp
64.             system("rm -r ../tmp/a");
65.         }
66.
67.         // Reclassify
68.         printf("Reclassifying rasters\n");
69.         reclassify(infile, i);
70.
71.         // Delete tmp files
72.         system("rm ../tmp/t*");
73.     }
74.
75.     // Calculate AHP
76.     printf("Computing AHP\n");
77.     geoahp(infile);
78.
79.     free_input(infile);
80.
81.     // Clean up tmp folder
82.     system("rm ../tmp/*");
83.     printf("Done! Thank you.\n");
84.     return 0;
85. }
86.
87. int syscall(int argc, char *argv[]) {
88.
89.     char cmd[MAX_NUM_CHARS];
90.
91.     cmd[0]='\0';
92.     for (int i=0; i<argc; ++i)
93.         strcat(cmd, argv[i]);
94.
95.     return system(cmd);
96. }
97.
98. int israster(struct input *infile, int i){
99.     return infile->layer[i].type/100 == 0;
100.       }
101.
102.       int distreq(struct input *infile, int i) {
103.           return infile->layer[i].type%10 == 0;
104.       }
```

The *raster_classify.c* in the *src* folder:

```
#include <gdal.h>
#include <cpl_conv.h>
#include "geoahp.h"

void reclassify(struct input *infile, int idx) {

    GDALAllRegister();

    char rastername[100];
    sprintf(rastername, "../tmp/t%d.tif", idx+1);

    GDALDatasetH hDataset = GDALOpen(rastername,GA_ReadOnly);

    // Number of columns, rows and bands
    int ncols = GDALGetRasterXSize(hDataset);
    int nrows = GDALGetRasterYSize(hDataset);

    // Projection
    const char*proj = GDALGetProjectionRef(hDataset);

    // Info for tfw
    double geotransf[6];
    GDALGetGeoTransform(hDataset, geotransf);

    // Read data
    float *parray = CPLMalloc(ncols*nrows*sizeof*parray);
    GDALRasterBandH hBand = GDALGetRasterBand(hDataset,1);
    GDALRasterIO(hBand, GF_Read, 0, 0, ncols, nrows, parray, ncols, nrow
s, GDT_Float32, 0, 0);

    // New array to hold reclassification data
    float *reclass = CPLMalloc(ncols*nrows*sizeof*reclass);

    // Reclassification
    int catnrows = infile->catfile[idx].nrows;
    for (int i=0; i<nrows; ++i) {
        for (int j=0; j<ncols; ++j) {
            float x = parray[i*ncols+j] ;
            if (x == -9999) continue;
            if (iscontinuous(infile, idx)) {
                int found = 0;
                for (int k=0; k<catnrows-1 && !found; ++k) {
                    int catmin = infile->catfile[idx].cat[3*k+0];
                    int catmax = infile->catfile[idx].cat[3*k+1];
                    if (catmin <= x && x < catmax) {
                        reclass[i*ncols+j] = (float) infile-
>catfile[idx].cat[3*k+2];
                        found = 1;
                    }
                }
                if (!found)
                    reclass[i*ncols+j] = (float) infile-
>catfile[idx].cat[3*(catnrows-1)+2];
            }
            else {
                int found = 0;
                for (int k=0; k<catnrows-1; ++k) {
                    int catmin = infile->catfile[idx].cat[3*k+0];
                    if (catmin == x) {
```

```
                        reclass[i*ncols+j] = (float) infile->catfile[idx].cat[3*k+2];
                        found = 1;
                    }
                }
            }
        }
    }

    // Save file
    sprintf(rastername, "../tmp/%d.tif", idx+1);

    GDALDriverH driver = GDALGetDriverByName("GTiff");
    GDALDatasetH ds = GDALCreate(driver, rastername, ncols, nrows, 1, GDT_Float32, 0);

    GDALSetProjection(ds, proj);
    GDALSetGeoTransform(ds, geotransf);

    GDALRasterBandH hband = GDALGetRasterBand(ds,1);
    GDALRasterIO(hband, GF_Write, 0, 0, ncols, nrows, reclass, ncols, nrows, GDT_Float32, 0, 0);

    GDALClose(ds);
    GDALClose(hDataset);

    CPLFree(parray);
    CPLFree(reclass);
}

int iscontinuous(struct input *infile, int i){
    return (infile->layer[i].type%100)/10 == 1;
}
```

The *read_input.c* file in the *src* folder:

```c
#include "geoahp.h"

// Concatenation of two strings
static char *mystrcat(const char *s1, const char *s2) {
    int ls1 = strlen(s1);
    int ls2 = strlen(s2);
    char *scat = malloc(ls1 + ls2 + 1);
    strcpy(scat, s1);
    strcat(scat, s2);
    return scat;
}

struct input * read_input(const char *geo,const char *alpha) {

    int slen;
    char string[MAX_NUM_CHARS];

    struct input *x = malloc(sizeof *x);
    if (x == NULL) return NULL;

    FILE *fid = fopen(geo,"r");
    if (fid == NULL) {
        free(x); return NULL;
    }

    // Reading cellsize
    fscanf(fid, "%s", string);
    slen = strlen(string);
    x->cellsize = malloc(slen);
    strcpy(x->cellsize, string);

    // Reading xmin
    fscanf(fid, "%s", string);
    slen = strlen(string);
    x->xmin = malloc(slen);
    strcpy(x->xmin, string);

    // Reading ymin
    fscanf(fid, "%s", string);
    slen = strlen(string);
    x->ymin = malloc(slen);
    strcpy(x->ymin, string);

    // Reading xmax
    fscanf(fid, "%s", string);
    slen = strlen(string);
    x->xmax = malloc(slen);
    strcpy(x->xmax, string);

    // Reading ymax
    fscanf(fid, "%s", string);
    slen = strlen(string);
    x->ymax = malloc(slen);
    strcpy(x->ymax, string);

    // Reading epsg
    fscanf(fid, "%s", string);
    slen = strlen(string);
    x->epsg = malloc(slen);
    strcpy(x->epsg, string);
```

```
61.
62.     fclose(fid);
63.
64.     fid = fopen(alpha, "r");
65.     if (fid == NULL) {
66.         free(x); return NULL;
67.     }
68.
69.     // Read alpha file
70.     for (x->ncrit=0; !feof(fid); )
71.         if (fgetc(fid) == '\n') ++x->ncrit;
72.
73.     x->layer = malloc(x->ncrit*sizeof*x->layer);
74.     if (x->layer == NULL) {
75.         free(x); return NULL;
76.     }
77.
78.     rewind(fid);
79.
80.     for (int i=0; i<x->ncrit; ++i) {
81.
82.         // Read file name
83.         fscanf(fid, "%s", string);
84.         slen = strlen(string);
85.         x->layer[i].name = malloc(slen);
86.         strcpy(x->layer[i].name, string);
87.
88.         // Read layer name
89.         fscanf(fid, "%s", string);
90.         slen = strlen(string);
91.         x->layer[i].lyr = malloc(slen);
92.         strcpy(x->layer[i].lyr, string);
93.
94.         // Read weight and type
95.         fscanf(fid, "%d %d", &x->layer[i].weight,&x->layer[i].type);
96.
97.         // Read the field to be burnt
98.         fscanf(fid, "%s", string);
99.         slen = strlen(string);
100.                x->layer[i].field = malloc(slen);
101.                strcpy(x->layer[i].field, string);
102.            }
103.
104.            fclose(fid);
105.
106.            // Reading cat files
107.            x->catfile = malloc(x->ncrit*sizeof*x->catfile);
108.            for (int i=0; i<x->ncrit; ++i) {
109.                sprintf(string,"%d",i+1);
110.                char *catfilename = mystrcat("../cat/",string);
111.                fid = fopen(catfilename, "r");
112.                int nrows = 0;
113.                while (!feof(fid))
114.                    if (fgetc(fid) == '\n') ++nrows;
115.                rewind(fid);
116.                x->catfile[i].nrows = nrows;
117.                x->catfile[i].cat = malloc(3*nrows*sizeof*x-
    >catfile[i].cat);
118.                for (int j=0; j<x->catfile[i].nrows; ++j)
119.                    fscanf(fid, "%d %d %d", &x-
    >catfile[i].cat[3*j+0], &x->catfile[i].cat[3*j+1], &x-
    >catfile[i].cat[3*j+2]);
120.                fclose(fid);
```

```
121.                free(catfilename);
122.            }
123.
124.            return x;
125.        }
126.
127.        void free_input(struct input *x) {
128.            for (int i=0; i<x->ncrit; ++i)
129.                free(x->catfile[i].cat);
130.            free(x->catfile);
131.            for (int i=0; i<x->ncrit; ++i) {
132.                free(x->layer[i].name);
133.                free(x->layer[i].lyr);
134.                free(x->layer[i].field);
135.            }
136.            free(x->layer);
137.            free(x->epsg);
138.            free(x->ymax);
139.            free(x->xmax);
140.            free(x->ymin);
141.            free(x->xmin);
142.            free(x->cellsize);
143.            free(x);
144.        }
```