Decision Making Support Tool Using Fast and Frugal Heuristics

Daniel Messenger

Bachelor of Science in Computer Science with Honours
The University of Bath
May 2010
This dissertation may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signed:
Decision Making Support Tool Using Fast and Frugal Heuristics

Submitted By: Daniel Messenger

COPYRIGHT

Attention is drawn to the fact that copyright of this dissertation rests with its author. The Intellectual Property Rights of the products produced as part of the project belong to the University of Bath (see http://www.bath.ac.uk/ordinances/#intelprop). This copy of the dissertation has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the dissertation and no information derived from it may be published without the prior written consent of the author.

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed:
Abstract

Decision making is a task we carry out on a daily basis. The types of decisions we make can sometimes be very simplistic which don’t require much reasoning, such as, what clothes to wear or what to watch on TV. Other decisions may be very complex and require lots of cognition and reasoning techniques in order to find the best and most appropriate alternative out of a set of alternatives. A real life decision people will be faced with at some point in their lives will be finding the best house or flat to buy within a given budget, or similarly the best car within a given budget.
# Contents

1. Introduction ......................................................................................... 1  
   Research focus .................................................................................. 1  
   Aims and Objectives ......................................................................... 1  
   Value of Research ............................................................................ 1  
   Outline structure .............................................................................. 2  
2. Literature Review ............................................................................ 3  
   Theories behind decision making .................................................. 3  
   Bounded Rationality ........................................................................ 4  
   Fast and Frugal ................................................................................ 4  
   Dominance ....................................................................................... 4  
   Additive Linear ................................................................................ 5  
   Additive Difference .......................................................................... 5  
   Satisficing (Conjunctive) ............................................................... 5  
   Disjunctive ....................................................................................... 5  
   Elimination by Aspects .................................................................... 5  
   Recognition Heuristic ....................................................................... 5  
   Take-The-Best (TTB) Heuristic ....................................................... 6  
   Decision Making Process ................................................................ 8  
   Outcome Oriented Approach ........................................................ 8  
   Process Oriented Approach ............................................................ 9  
   Pre-decision Stage .......................................................................... 9  
   Partial Decision Stage ...................................................................... 9  
   Final Decision Stage ........................................................................ 9  
   Post-decision Stage .......................................................................... 9  
   Decision Making Process Diagram ............................................... 10  
   The Analytic Hierarchy Process ...................................................... 11  
   Data Representation ......................................................................... 13  
   The Attribute Explorer .................................................................... 14  
   Data Visualisation Background ...................................................... 15  
   Data Visualisation .......................................................................... 17  
3. Requirements ................................................................................... 18
User Documentation

Disjun

Satisficing (Conjunctive)

Elimination by Aspects

Lexicographic (Take

Final

Phase 3

Phase 2

Phase 1

Final interface

interface

Designing the main table for the spreadsheet

Design Reuse

UML Class Diagrams

Pseudo code for setting cut off value thresholds

Pseudo code for setting up the interface

makeFrame() ........................ 24

makeMenu() ................................ 24

makeToolBar() ................................ 25

Implementation .................................................. 26

Phase 1 .................................................. 26

Phase 2 .................................................. 29

Phase 3 .................................................. 31

What does the program do? .................................. 55

Opening the program ........................................ 55

Results .......................................................... 35

Lexicographic (Take-The-Best) .................................. 35

Elimination by Aspects ........................................ 36

Satisficing (Conjunctive) ...................................... 38

Disjunctive ..................................................... 39

Conclusion ....................................................... 41

Bibliography ...................................................... 43

Appendix A ...................................................... 45

Appendix B ...................................................... 48

Appendix C ...................................................... 50

Appendix D ...................................................... 55

Functional Requirements ........................................ 18

Non-Functional Requirements ................................. 18

4. Design .......................................................... 20

Interface sketches ............................................... 20

Displaying data when a cut off threshold has been set ........... 20

Designing the main table for the spreadsheet ..................... 21

Code Reuse ....................................................... 22

Pseudo code for setting up the interface .......................... 24

makeFrame() .................................................... 24

makeMenu() ..................................................... 24

makeToolBar() .................................................. 25

5. Implementation ................................................ 26

Phase 1 .......................................................... 26

Phase 2 .......................................................... 29

Phase 3 .......................................................... 31

Final interface .................................................... 32

6. Testing ............................................................ 33

7. Results ............................................................ 35

Lexicographic (Take-The-Best) .................................. 35

Elimination by Aspects ........................................ 36

Satisficing (Conjunctive) ...................................... 38

Disjunctive ....................................................... 39

8. Conclusion ....................................................... 41

9. Bibliography ...................................................... 43

10. Appendix A ...................................................... 45

11. Appendix B ...................................................... 48

12. Appendix C ...................................................... 50

13. Appendix D ...................................................... 55

User Documentation ............................................. 55

Opening the program ............................................ 55
What is a cut off value? ........................................................................................................ 55
How do I retrieve my data once it’s been removed from the table after setting a cut
off value? .......................................................................................................................... 55
How can I move columns? ................................................................................................. 55
Why is the cut off value not being set for a particular cell? ............................................. 55
How do I sort a column ascending or descending? ......................................................... 56
How can I edit the header text of a column? ................................................................. 56
How do I close the spreadsheet? ..................................................................................... 56
14. Appendix E .................................................................................................................. 57
Table of Figures

Figure 1: Flow diagram of the Take-The-Best (TTB) algorithm ........................................ 7
Figure 2: Discrimination rule between two alternatives. If one alternatives cue ............... 7
Figure 3: A simple example of how Take The Best works. We are assuming you walk to
the local shops and the weather conditions are poor. Which set of clothes do you wear?.. 8
Figure 4: Top half of the decision making process tree, taken from (9) ........................... 10
Figure 5: Bottom half of the decision making process tree, taken from (9) ...................... 11
Figure 6: Decomposition of a problem into a Analytical Hierarchical Process ............... 12
Figure 7: The Attribute Explorer tool interactive histogram, taken from Tweedie and
Spence - The Attribute Explorer (11) ............................................................................. 14
Figure 8: The Attribute Explorer tool interactive histogram, taken from Tweedie and
Spence - The Attribute Explorer (11) ............................................................................. 15
Figure 9: 10th century graph displaying the movement of planets as described by
Funkhouser[4] ..................................................................................................................... 16
Figure 10: Series of graphs displaying how many children like a specific ice cream ...... 17
Figure 11: JTable structure taken from the official Java website and JTable API ........... 22
Figure 12: HTA diagram on performing a cut off value to a specific cell .................... 48
Figure 13: HTA diagram on performing a cut off value to a specific cell v.2 ............... 48
Figure 14: UML Class diagram ...................................................................................... 49
Figure 15: Final interface with simple cell values ....................................................... 50
Figure 16: Interface after cut off value has been set to “less than” on currently selected
cell ................................................................................................................................ 50
Figure 17: Interface after cut off value has been set to “greater than” on currently selected
cell ................................................................................................................................ 50
Figure 18: Test 1 of the test plan ..................................................................................... 51
Figure 19: Test 2 of the test plan ..................................................................................... 51
Figure 20: Test 3 of the test plan ..................................................................................... 51
Figure 21: Test 4 of the test plan ..................................................................................... 51
Figure 22: Test 5 of the test plan ..................................................................................... 52
Figure 23: Test 6 of the test plan ..................................................................................... 52
Figure 24: Test 7 of the test plan ..................................................................................... 52
Figure 25: Test 8 of the test plan ..................................................................................... 52
Figure 26: Test 15 of the test plan .................................................................................. 53
Figure 27: Test 16 of the test plan .................................................................................. 53
Figure 28: Spreadsheet in use with a real world example of choosing the most appropriate
house to buy .................................................................................................................. 54
Figure 29: Price column sorted ascending ................................................................. 56
Figure 30: Price column sorted descending ............................................................... 56
1. Introduction

My project will be mainly research based on the topic of decision making using fast and frugal heuristics. I will begin by performing an in depth literature survey in order to get a greater understanding of the subject. The topic is completely new to me so the literature survey will be of great benefit and will have a huge contribution to the success of this project.

Research focus

Most of my time will go into in depth research into the psychology of how decisions are made, the process of deriving a decision and understanding what drives humans to a decision. Additionally, my main research and focus area will be investigating and analyzing the different types of heuristics which contribute toward decision making; these heuristics are known as fast and frugal. It’s absolutely imperative to have a full understanding and be able to clearly distinguish between each of the different types of heuristics which are available through fast and frugal. Having a full understanding of these fast and frugal heuristics will make it a much easier task of implementing them into a spreadsheet which will support decision making.

Aims and Objectives

The aim of the project is to have a greater understanding of how decisions are derived and the factors behind what drives a decision. I intend on designing, implementing, testing and documenting a spreadsheet which fully integrates as many fast and frugal heuristics as possible. My main objectives are:

- Understanding how decisions are made
- Understand what drives a human to a decision
- Research what fast and frugal heuristics exist
- Understand these fast and frugal heuristics
- Implement a spreadsheet integrating fast and frugal heuristics
- Get a much broader understanding on the topic in general

Much of my time will be spent on researching and investigating all of the things listed above. I estimate that 2 to 4 months of my overall project will be spent researching and gathering a greater understanding of the topic.

Value of Research

Research is valuable no matter what you are doing. Having a greater knowledge about something you perhaps didn’t previously know is always an advantage and benefit to have. In terms of my project I have a very small knowledgebase and
understanding of the topic so every piece of research I do and every single paper I read is a huge benefit and will be of great significance to the success of this project. I intend on researching as much as possible to get as much understanding as I can on the topic which will make the project a lot more interesting and motivate me to push for success. Additionally, the more research I undertake the better quality of software will be produced at the end of this project.

Outline structure

I will begin by performing an in depth literature survey covering lots of relevant information, followed by:

- A set of functional and non-functional requirements that the program is required to do.
- Designing the spreadsheet including sketches, class diagrams, pseudo code etc…
- The different phases and iterations of the implementation process.
- Explaining the final design of the interface and why certain things were done the way they were.
- Testing the program to ensure that there are no major bugs or inconsistencies
- Applying the spreadsheet to real world dilemmas with the use of fast and frugal heuristics
- Concluding how the project went, what could of gone better and future work.
2. Literature Review

There is much needed research in order to implement a tool which supports group decision making. Both the psychological and human computer interaction approaches to group decision making need to be fully understood in order to develop such a support tool. There are several critical heuristics which need to be analysed and evaluated.

Another critical region which needs to be fully evaluated is how information is input and more importantly how it is displayed to the end user. This region is where more traditional human computer interaction techniques apply, such as, cognitive walkthroughs, usability principles, and hierarchical task analysis and heuristic evaluations.

I will begin by identifying the psychological theories behind decision making, as well as, the different types of decision making heuristics and analysing the most important ones which are appropriate to group decision making support. There are several very interesting papers from Tweedie and Spence which focus on how information is displayed to users, whether it's in the form of a graph, highlighted cells in a spreadsheet or other complex techniques. Obviously, a vast amount of research will need to be put into the more psychological and theoretical papers which describe and analyse decision making theories, including how decisions are made and why.

Theories behind decision making

Decision making is a task we carry out on a daily basis, what we choose to have for breakfast, what clothes we choose to wear and how we travel to work in the morning, whether it's cycling, driving, public transport or walking. According to classical decision making (1) the most effective way to derive a decision from a set of particular parameters is to assign a weight for each parameter, or dimension. By assigning a weight to each parameter which affect the way we come to a conclusion causes trade-offs, which are known as "compensatory". These values compensate for values on different parameters. For example, if we wanted to walk to work but the weather was bad, the "walking" parameter within the decision would essentially be traded off because the weather parameter would carry a very small weight. This would result in finding an alternative way of travelling to work. This decision making approach is very time consuming and tedious so generally human decisions are made using a "fast and frugal" approach.
Bounded Rationality

Bounded rationality is a study of how humans, and even animals make inferences on what to do in a given scenario or circumstance. Essentially, this means that a conclusion is derived by applying rules of logic and semantics. These inferences are made with limited knowledge and understanding about something, in addition to limited amount of time (2). A decision has to be made quickly. This is where Fast and Frugal techniques can be applied. There are specific heuristics which deal with analysing a situation quickly and coming up with a decision on what to do.

Fast and Frugal

Fast and Frugal are a set of heuristics that meet the criteria of Goldstein and Gigerenzer (3). Fast and Frugal heuristics compose of a set of algorithms (TTB, Integration, Recognition) (4) which attempt to reduce time and memory constraint on human decision makers allowing us to come to a decision quickly and without the use of complicated, time consuming statistical methods. Humans are limited by their cognitive abilities and how much information can be processed and stored at a given time. This limitation can make some larger decisions very difficult. The Fast and Frugal heuristic is fast because it can solve the problem in a very short period of time, and it’s frugal because there is very small information required for the heuristic to work (5). There are a number of Fast and Frugal heuristics including:

- Dominance
- Additive Linear
- Additive Difference
- Satisficing(Conjunctive)
- Disjunction
- Lexicographic(Take-The-Best)
- Elimination By Aspects
- Recognition

All of the heuristics taken above are listed in the book by (6).

Dominance

This heuristic searches for an alternative that is as good as every other alternative on all high priority attributes and chooses the alternative or simply removes all other worse alternatives from the selection choice.
Additive Linear

Every attributed is sorted by importance and then each alternative is considered one at a time and a global value is calculated and weighed up when all attributes have a global value.

Additive Difference

Two alternatives are considered at a time and compared attribute by attribute. The difference between each attribute is then calculated. Once the difference of all attributes within the two alternatives are calculated the global difference can be calculated. This process is repeated until all pairs of alternatives have been evaluated and the best alternative is the one which won all pair-wise comparisons.

Satisficing (Conjunctive)

A cut off value is set on all important attributes and the first alternative which is in range of the cut off value is evaluated. All alternatives above the cut off points are considered for further evaluation.

Disjunctive

A cut off value is set and the first alternative that is at least as good as the cut off threshold on any attribute is evaluated. Or a strategy is used to select a set of alternatives that are very good on at least one dimension for further consideration.

Elimination by Aspects

A cut off threshold is set and all attributes outside of this threshold is thrown away and not considered for further evaluation. All attributes inside the threshold remain and the process is repeated on the next attribute. This is repeated until one alternative is left.

Recognition Heuristic

The recognition heuristic is one of the heuristics within Fast and Frugal. This particular heuristic essentially places a higher weight the most recognised parameter.

"Recognition heuristic: If one of two objects is recognized and the other is not, then infer that the recognized object has the higher value with respect to the criterion."

G. Gigerenzer
(5) Puts forward an example of a $1 million question in a game show, "Which city has more inhabitants, San Diego or San Antonio?". The article claims that two thirds of American undergraduates answered the question correctly. Whereas, 100% of Germans answered the question correctly. This is because of the recognition heuristic. The German people who answered the question undertook some very simple, yet effective reasoning to derive their answer. Germans had only heard of San Diego and could use the recognition heuristic and apply it to their answer. The Americans however couldn't use the heuristic since they had heard of both cities. (7) Performed an experiment on 50 Turkish students and 54 British students. The experiment was a simple question, which football team would win between Manchester United v Shrewsbury Town? Obviously, the British students knew more about the two teams than the Turkish students but the Turkish results were almost identical to the British results. 63% of the Turkish students predicted Manchester United, whereas, 66% of British students predicted Manchester United. These predictions were consistent 95% of the time.

**Take-The-Best (TTB) Heuristic**

Take The Best, otherwise known as TTB is a "Simple Heuristic" (8) within Fast and Frugal that chooses between two alternatives and decides which will hold the greatest weight for a given criterion. Each alternative will have particular parameters/cues/attributes where they will be compared against each other in order to compute the most relevant option for the given scenario. TTB can be used in any situation where two alternatives share the same parameters and the human applying the heuristic has a goal. For example, you have two different sets of clothes you want to wear but it's raining outdoors. The goal would be to get from A to B while staying warm and dry. Each set of clothes would be given weighted parameters on how dry you would be and how warm you will be.
Figure 1: Flow diagram of the Take-The-Best (TTB) algorithm.

Figure 1, taken from (4) illustrates how the Take The Best (TTB) algorithm operates and the steps involved in determining which "cue" or parameter is the best suited for a given criterion. By following the TTB algorithm in Figure 1 you will choose the correct alternative by carrying out simple reasoning techniques.

Figure 2: Discrimination rule between two alternatives. If one alternatives cue weighs more than the other alternatives cue then this is said to be a discrimination rule. All 4 cases of the discrimination rule are shaded in grey.

The algorithm works in 5 steps (4).

1. Recognition rule: You _rstly identify the two possible objects. If you
recognise one of the two objects then take the recognised object. If no object is recognised then choose randomly between the two. Otherwise, if both objects are recognised then go ahead to step 2.

2. Identify the cues from memory.

3. Apply the discrimination rule in Figure 2 to decide whether a cue discriminates the other.

4. If a cue discrimination is found then stop identifying cues. If a cue does not discriminate then iterate back to step 2 until a cue is found that does discriminate.

5. Choose the object with the highest cue values. Choose randomly if no cue discriminates the other.

The example shown in Figure 3 is a very simple, and basic decision to make in the given criterion. However, you can clearly see that the "best" set of clothes is set 1. The answer to this may appear obvious but this is a simple illustration of how TTB works and the reasoning behind how the decisions were derived.

<table>
<thead>
<tr>
<th>Cues</th>
<th>Clothes (Set 1)</th>
<th>Clothes (Set 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep dry</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Keep warm</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Comfortable</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

Figure 3: A simple example of how Take The Best works. We are assuming you walk to the local shops and the weather conditions are poor. Which set of clothes do you wear?

Decision Making Process

(9) Explains there are two very basic approaches in order to derive a decision from a given problem. The first approach is the outcome-oriented approach, the second approach being process-oriented. The decision making process is split into 4 categories. The pre-decision stage, Partial decisions, Final decision stage and the Post-decision stage.

Outcome Oriented Approach

One of two basic decision making processes is the outcome-oriented approach. The outcome-oriented approach essentially says that if you can correctly predict the outcome of a decision then you understand the decision making process for that given scenario. This approach is widely based around whether you can correctly predict the outcome of a decision (9).
Process Oriented Approach

The second basic decision making process is the process-oriented approach. This approach is described as, if you understand the decision making process then you can accurately predict the results of a given choice in a given scenario (9).

Pre-decision Stage

The initial stage of making a decision is going through different possibilities and carrying out reasoning techniques to understand whether or not the decision is feasible. Given a certain criteria, the human decision maker goes through many alternatives examining and analysing if the decision can indeed be followed through and whether it's the best possible decision in the given circumstances. Finding a conflict or non-feasible decision the human quickly moves onto another alternative weighing up and analysing whether this new decision is feasible. The whole pre-decision stage essentially derives a set of alternative decisions, analyses them, evaluates them and comes to a separate decision of whether or not the alternatives are feasible (9).

Partial Decision Stage

Partial decisions are derived from the pre-decision stage. A set of feasible decisions are generated and stored in memory. It is these pre-generated, feasible decisions that are used and tweaked in the pre-decision stage. Previous decisions that were considered non-feasible are added once again to the list of alternatives but this time removing or changing criteria in order to make the decision more feasible.

Final Decision Stage

The final decision is a process of iterating through all possible decisions carried over from the partial decision stage and eliminating the less attractive ones one-by-one. As the list of alternatives become smaller and smaller, the elimination process becomes more and more difficult. When the list of alternatives is small the decision maker becomes more attracted to specific options within the alternatives. From this point onwards, the decision maker can derive a suitable decision based on the most attractive options available. As the alternatives get smaller and smaller the decision maker becomes more and more committed to specific options. Once the level of commitment reaches a certain level the final decision is derived.

Post-decision Stage

Once a final decision has been derived the decision maker enhances or tweaks the attractiveness of the appropriate options within the decision. The decision maker re-evaluates and reassesses the final decision and performs a cognitive tune up (9). The transition from the pre-decision stage to the post-decision stage removes all
negative elements from the decision so the decision maker is confident that the
decision is not only the correct one, but the best it can possibly be.

**Decision Making Process Diagram**

Figure 4 and 5 illustrates a very detailed decision making process, based on the
Pre-decision, partial decision, final decision and the post-decision stages.
The Analytic Hierarchy Process

A hierarchy is a representation of a top-down model approach. Everything is arranged in a descending order with the most important element at the top of the hierarchy descending down to the least important. There can be a multiple number of levels between the top level and bottom level. A decision making hierarchy is no different, a top level exists which is the goal of the decision and it descends all the way down to the bottom level including every element, parameter or stakeholder that plays a role in reaching the goal level (10). A hierarchy in the case of decision making is used to clearly identify what factors, attributes, stakeholders etc... affect
the decision and the steps taken to reach the decision. The use of a hierarchy breaks down a complex problem into small sub-problems on different levels or dimensions. Each sub-problem can be weighed up and assessed so it can be placed on the relevant level of the hierarchy. Each element on a specific level can be compared against other elements in order to check the importance of that element and/or other elements. A simple example of an analytic hierarchy process (AHP):

You want to buy a car out of a possibility of three. The goal of the hierarchy would be; how satisfied am I with the car? The three candidate cars will be on the very lower tier of the hierarchy and the different elements which contribute toward the goal are on the second tier of the hierarchy. In my example, there are very few elements on the second tier. There could be many, many more elements which could potentially be used, such as, the tax bracket in which the car is, Engine size, Boot space, ABS, Power Assisted Steering etc... My example simply has 5 different elements:

1. Colour - colour preference of the car.
2. Fuel Type - Diesel or Petrol.
3. Cost - Cost of the car.
4. Age - Age of the car.
5. No. Doors - Number of doors on the car, 3 or 5.

![Diagram of Analytical Hierarchical Process]

Once the hierarchy has been established with all of the possible candidates and elements which affect the decision, a list of priority needs to be established in terms of which elements on tier 1 are the most important.
<table>
<thead>
<tr>
<th>Importance</th>
<th>Definition</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 1</td>
<td>No importance</td>
<td>Less important than another</td>
</tr>
<tr>
<td>1</td>
<td>Equal importance</td>
<td>Two elements contribute evenly</td>
</tr>
<tr>
<td>3</td>
<td>Moderate importance over another</td>
<td>Judgment favors one over another</td>
</tr>
<tr>
<td>5</td>
<td>High importance over another</td>
<td>Judgment favors one over another</td>
</tr>
<tr>
<td>2,4</td>
<td>When compromises are needed</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Scale of importance

<table>
<thead>
<tr>
<th>Element</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Priority Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0.5</td>
<td>5.2</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>0.3</td>
<td>0.5</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>4.9</td>
</tr>
</tbody>
</table>

Table 2: Compute the priority list for each element on tier 1 of the hierarchy

Now a priority list has been established we can compute a matrix to verify the order in which elements consist in terms of priority. List of priority:

1. Cost.
2. Age.
3. Fuel Type.
5. Colour.

The list of priority is now determined and you can clearly see that cost is the most important; age is the second highest on importance etc... You can make a decision on which car would be the most appropriate knowing your priority list.

**Data Representation**

A major issue with any problem is the way it's visually displayed to the end user. The data being displayed must be easy to read and have clear mappings between necessary labels to reduce the cognitive strain on the viewer. Tweedie and Spence (11) describe visual data as being too static. Hardware and software have evolved rapidly over the last decade (12) which essentially means that we can perform more complex ways of displaying data to the user and make it a lot more interactive and visually easier to understand. Because of the evolution of hardware we can move away from very one-dimensional ways of representing data and start representing it in a more readable and interactive form.
The Attribute Explorer

Tweedie and Spence implemented a project called "The Attribute Explorer" (11). This project was a study into how data were represented within a graphical interactive tool which visualised multi-dimensional parameters within data sets. The tool is used to assist users who have a number of possible candidate elements that share a set of common parameters. The tool identifies which parameter(s) are the highest on priority based on the weight of a given parameter within an element. The tool would break the problem down into a single suggestion in terms of which candidate would be the most appropriate to the given scenario. Tweedie and Spence put forward an innovative way of representing data with the use of an interactive histogram. Using an interactive histogram allow users of the tool to interactively change the weight of a given parameter within an element by sliding the weight of an element to the appropriate value. The tool could then determine which elements were of greater importance and categorise these elements and perform calculations on what data the user selected. By changing the weight of one element all other histograms were automatically updated and represented to the user in a separate histogram with a critical path form of visually displaying the data. Figure 7 illustrates the interactive histogram in which users slide each elements parameter to assign it's weight value.

The elements within the histogram are the four different house candidates, and the parameters are:

1. Type of House
2. Price (1000)
3. No. bed rooms
4. Garden (feet)
5. Dist. from st. (mins)

Each parameter is given a weight value by selecting a certain number of houses within the histogram. Once every parameter has been assigned its weight value the second histogram is updated in the form of a critical path leading to the most appropriate element.

![Figure 2: Once a sub-population has been identified the individual “house-lines” can be examined.](image)

Figure 8: The Attribute Explorer tool interactive histogram, taken from Tweedie and Spence - The Attribute Explorer (11)

**Data Visualisation Background**

Data visualisation techniques date all the way back to the 10th century. A 10th century graph described by Funkhouser (13) and reproduced by Tufte (14), as seen in Figure 9. There is a wide selection of 16th, 17th, 18th century, up to present graphs which are heavily detailed in "The visual display of quantitative Information" (14).
Figure 9: 10th century graph displaying the movement of planets as described by Funkhouser[4].

Data representation has been used for many, many centuries which prove that it's a very effective technique of displaying raw data and translating the data into something more meaningful. Even the simpler tables of data are interpreted better and easier as a graph. Take this simple example in table 3 of how many children like a specific flavor of ice cream. Understandably, the information is easy to read since it's a very simple example:

<table>
<thead>
<tr>
<th>Ice Cream Flavour</th>
<th>Chocolate</th>
<th>Strawberry</th>
<th>Vanilla</th>
<th>Raspberry</th>
<th>Fudge</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Children</td>
<td>50</td>
<td>30</td>
<td>10</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 3: Table illustrating the number of children who enjoy different types of ice cream.

If the table is translated into a series of graphs then the information will be even simpler to read. Figure 10 shows how table 3 is translated and visually represented as a simple graph. It's more appealing to the eye which makes it easier to understand and read the data, thus making it more understandable and meaningful.
Data Visualisation

Data visualisation is a huge research area in trying to understand the best ways in which data can be represented in its most readable form (15). Humans try to perceive data on a screen in front of them as a mental image (16). Processing lots of text as an image makes the data more readable and generally easier to understand and identify patterns (17) and sequences. Visualising something is a mental, cognitive process that humans must carry out and not something a computer must do for them. A computer can try to illustrate the data as best as possible but it’s down to the human to generate a mental model of this data in such a way that it’s easy to understand for that particular user. Humans are unique and generate their own mental models which imply that everybody may come up with different mental models. Although a computer cannot generate a human’s mental model it can indeed influence the way the mental model is generated. This is done by breaking a large amount of data down into a readable graph, which can then be broken down again into a graph with possible sub graphs.
3. Requirements

Before implementation and design can begin I must prepare and have a well defined set of requirements in order for the piece of software to be of maximum benefit for the user. All requirements are derived from the literature survey. This project is mainly research based rather than software development and some requirements may appear very high level. However, I feel some form of list of requirements is essential for the project to run smoothly.

Functional Requirements

1. All columns in the spreadsheet must be fully editable during runtime
   1.1. Must be able to change the text within column headers
   1.2. Must be able to move all columns in order to specify a priority
   1.3. All columns must be sortable
      1.3.1. Ascending
      1.3.2. Descending
2. Must be able to select and edit every cell
3. The spreadsheet window must follow standard operating system standards
   3.1. Have a title
   3.2. Standard windows icons and functions
      3.2.1. Minimise
      3.2.2. Maximise
      3.2.3. Restore
      3.2.4. Close
      3.2.5. Resizable
   3.3. Have a menu bar with an alternative way to close the window
      3.3.1. File -> Exit
4. Must be able to set a cut off value threshold
   4.1. Set threshold “less than” value in current selected cell
   4.2. Set threshold “greater than” value in current selected cell
   4.3. Values which are not within the threshold should be:
      4.3.1. Deleted from the spreadsheet
      4.3.2. Or grey out the cells in the row

Non-Functional Requirements

1. The spreadsheet must be implemented in a language which works on cross-platforms.
   1.1. Windows
   1.2. Linux
   1.3. OSX
   1.4. Solaris
   1.5. …
2. The spreadsheet must integrate as many decision making heuristics as possible
   2.1. Take-The-Best
   2.2. Dominance
   2.3. Additive Linear
   2.4. Additive Difference
   2.5. Satisficing
   2.6. Disjunctive
   2.7. Elimination By Aspects

Now there are some concrete and explicit requirements in place I can move onto planning and designing how the piece of software will come together in terms of interface design and sketches, the types of components that will appear on the interface and where they will be placed, as well as, the classes needed for the spreadsheet to function properly.
4. Design

Interface sketches

The chosen language to implement the piece of software was Java. I felt it was essential for the first phase or iteration of the design process was to create a handful of sketches of potential interface designs as seen in Appendix A. Not only do these sketches give a good idea of how the interface will look, it shows the sort of components that will be used within the JFrame and how they are positioned.

Figure 1 of Appendix A illustrates my initial design. The first thoughts of the interface were that it seemed very simplistic and ideal. However, there appeared to be a slight problem when setting a cut off value. There were too many steps involved in setting a cut off value as illustrated in the Hierarchical Task Analysis diagram in Figure 12 of Appendix B.

In order to set the cut off value the user was required to:
1. Select an initial cell within a column
2. Click “set cut off value” from the menu level
3. Within this menu there are sub menus:
   3.1. Less than
   3.2. Greater than
4. Clicking one of these sub menus brought up a dialog box as seen in Figure 2 of Appendix A.
5. The user would type a value into the dialog box and click “OK”.
6. The cut off value would then be set and rows were removed accordingly.

The sketches clearly show the problem I am discussing. Therefore, the interface sketch was changed to the one as shown in Figure 3 of Appendix A. The menu level which existed previously has been removed and replaced by a toolbar which can be:
1. Drag and dropped into a different position of the window
2. Removed from the toolbar so it acts as an independent window

By cutting out this extra level of menus it greatly speeds up the way in which a cut off value is set. Looking at the Hierarchical Task Analysis diagram in Figure 13 of Appendix B there is a clear distinction between how many steps are involved in setting the cut off value.

Displaying data when a cut off threshold has been set

When a cut off value threshold has been set the cells which are out of range of the threshold should be dealt with accordingly. The sketch in Figure 4 of Appendix A is one possibility of dealing with data outside of the threshold range. Graying out cells and essentially disabling these cells ensure that the user cannot modify nor do anything with these redundant cells. By graying out the cells out of range of the
threshold the user can easily identify rows/cells which are still within the threshold and are still present for further inspection and breakdown.

Another possibility proposed in Figure 4 and Figure 5 of Appendix A is to completely remove cells/rows from the spreadsheet so that there is no way of the user being confused with the cells still being present. This introduces another problem of when the cells are deleted from the table the user cannot retrieve the data that has been removed. One possible way of overcoming this problem is illustrated in Figure 3 of Appendix A. Having a reset button will allow the user to revert back to the previous state of their table and essentially retrieve their data.

When a user inputs various data into the spreadsheet for support with decision making the current state of the spreadsheet is saved into a separate array of data once a cut off value has been set. For example, the sketch suggests that if the user entered some data such as:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 4: User entered data

Assuming the grey cell is selected and the user clicks the “less than” button on the toolbar, every row in column 2 is then evaluated to find a value less than 5. If a value less than 5 is found then that row will be removed from the spreadsheet as illustrated in the table below.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 5: Data in the spreadsheet after the “less than” threshold has been selected

The light blue cell in the table above is the row which has been removed from the table because it’s less than the value 5 which was selected as the initial value. Because the user clicks a cut off button (less than or greater than) the data from Table 4 will be stored in a separate array and will be re-entered into the spreadsheet when the user clicks the “reset” button as illustrated in Figure 3 of Appendix A. Once the “reset” button has been clicked the data in Table 5 will be repopulated with the initial data within Table 4.

**Designing the main table for the spreadsheet**

In order to implement a spreadsheet within Java the JTable class must be used. This class deals with what you see on the interface as well as the functionality behind the table. Each cell is rendered as a JLabel and processed in a way such that it essentially becomes one cell within the table. A JTable doesn’t simply give you a spreadsheet interface to work from you need to specify a table model which affects the way a JTable behaves within java. Implementing your own table model allows you to have greater control over the functionality of the table and override the
default table model provided by Java in order to perform much more sophisticated operations. The structure of a JTable appears like this:

The table object is the physical table which appears on the interface and behind the physical layer is a table model which controls how the table behaves in terms of what to do when a cell is selected, or when a value is set in a specific cell. The table model object even controls things such as the headers within the table and listens for changes so that it can fire triggers to update the table contents. Behind the table model is the physical, raw data that is stored in each cell. The data is stored as an Object within Java and the toString() method has to be invoked in order to convert the contents to a string, double, integer or float etc...

**Code Reuse**

Code reuse is a big thing and definitely makes sense when implementing something which has already been done by somebody with a greater understanding on the subject. Therefore, I chose to reuse an open source table data model which deals with all of the necessary methods that I would of used, in addition to allowing table headers to be fully customizable during runtime. The table data model I reused already allowed users to edit table headers on the fly rather than implementing it from scratch.

Although the table data model was already implemented I had to modify the code in such a way that column headers could be moved and rearranged during runtime as well as allowing columns to be sortable by ascending and descending.
UML Class Diagrams

Figure 14 illustrates my design plan of implementing the spreadsheet, including the classes involved and their attributes and methods with the necessary arguments each method will take. EditableHeader, EditableHeaderUI and EditableHeaderTableColumn are all from the open source code which deal with the back end of the spreadsheet and the EditableHeaderExe class is where the main method is placed, along with the necessary methods to create the user interface, set the window size, add every component to a container so it can be displayed to the user and deal with setting cut off value thresholds.

Pseudo code for setting cut off value thresholds

Below is a sequence of pseudo code that I plan on using in order to set cut off values for the spreadsheet rows.

\[
\begin{align*}
\text{Int row} &= \text{table.getSelectedRow}(); \\
\text{Int col} &= \text{table.getSelectedColumn}(); \\
\text{Double cutoffval} &= \text{convert to double(table.getValueAt(row, col))}; \\
\text{Double value}; \\
\text{Loop through rows 0-8} & \{ \\
& \quad \text{Value} = \text{convert to double(table.getValueAt(row, col))}; \\
& \quad \text{If(} \text{operation} = \text{less than and value < cutoffval} \text{)} & \{ \\
& \quad & \quad \text{Loop through column 0-9} & \{ \\
& \quad & \quad & \quad \text{Remove data using setValueAt(...);} \\
& \quad & \quad \} \\
& \quad & \text{\}} \text{ else if(} \text{operation} = \text{greater than and value > cutoffval} \text{)} & \{ \\
& \quad & \quad \text{Loop through column 0-9} & \{ \\
& \quad & \quad & \quad \text{Remove data using setValueAt(...);} \\
& \quad & \quad \} \\
& \quad \} \\
\end{align*}
\]

The plan is to set the \textit{cutoffval} variable to be the value of the currently selected cell. Once the \textit{cutoffval} variable has been set the method will move onto iterating through each row finding values which are less than or greater than the cut off value in the given column. If a value is identified as being out of range of the cut off threshold then the method will go onto iterating through each column of the row in which this out of range value is placed and replacing the value with a \textit{null} object. This will ensure that that the row is completely removed from the spreadsheet data.

Of course the data isn’t completely lost because the first time the user clicks “less than” or “greater than” the current state of the table will be stored within an independent object array which can later be restored by clicking the “reset” button.
Pseudo code for setting up the interface

Below is a block of pseudo code that I will follow in order to implement the main interface. I will have three methods which all contribute toward making the interface:

- makeFrame()
- makeMenu()
- makeToolBar()

**makeFrame()**

```java
public void makeFrame() {
    2Darray containing header names = { ... };
    2Darray containing default cell values = { ... };

    Add JFrame component
    Add JTable component with header names and cell values

    Add JScrollpane component
    Add JTable to scrollpane

    Invoke makeMenu() method
    Invoke makeToolbar() method

    Pack the frame
    Set the default close operation
    Set the default frame size
    Set the frame to be visible
}
```

**makeMenu()**

```java
public void makeMenu(JFrame frame) {
    Add JMenu component
    Set the menubar in the frame

    Add a File menu
    Add the file menu to the menu frame

    Add an Exit menu component
    Add the Exit menu component to the file menu structure
}
```
makeToolBar()

public void makeToolBar() {
    Add JToolBar and set its orientation to horizontal
    Add the less than button component
    Add a tooltip to the button
    Set the action command for the button
    Add the button to the toolbar

    Add the greater than button component
    Add a tooltip to the button
    Set the action command for the button
    Add the button to the toolbar

    Add the rest button component
    Add a tooltip to the button
    Set the action command for the button
    Add the button to the toolbar

    Add the toolbar to the frame
}
5. Implementation

Now the design and planning phase is in place I can begin implementing the piece of software paying particular attention to the requirements set in the requirements phase.

Phase 1

My chosen design from the sketches is the one which appears in Figure 3 of Appendix A. I feel this is the better of the two because of what I discussed in the Interface sketches section of the design plan. To reiterate what I was saying, the chosen interface requires a lot less steps in order to set a cut off value in comparison the interface in Figure 1 of Appendix A. Rather than being required to go through the process of entering menus and sub-menus and having dialog boxes pop up where the cut off value is to be set. I feel it’s a lot more viable and simpler to select a cell and the value in the selected cell will be used as the cut off value instead of entering it manually in the dialog box.

Now the interface design has been chosen I can start putting the components in place of the JFrame. The class diagrams of Figure 14 in Appendix B shows that I will be using three methods in the EditableHeaderExe class, which is where the main method will be placed:

```java
public static void main(String[] args) {
    ...
}
```

The three methods which create the interface are:

```java
public void makeFrame(...) {
    ...
}
p
public void makeMenuBar(...) {
    ...
}
p
public void makeToolBar(...) {
    ...
}
```

makeFrame(...) is the method which sets up the main window, or JFrame and places a JTable called table into a JScrollPane. Firstly, I set up an array of type String containing the default header names for each column within the JTable.

```java
String[] columnNames = {"Alternatives",
"1",
"2",
"3",
"4",
"5",
"6",
"7",
"8",
"9",
```
I then setup another array which contained the default values for each cell within the JTable.

```
Object[][] data = {
    { "Option 1", null, null, null, null, null, null, null, null, null, null },
    { "Option 2", null, null, null, null, null, null, null, null, null, null },
    { "Option 3", null, null, null, null, null, null, null, null, null, null },
    { "Option 4", null, null, null, null, null, null, null, null, null, null },
    { "Option 5", null, null, null, null, null, null, null, null, null, null },
    { "Option 6", null, null, null, null, null, null, null, null, null, null },
    { "Option 7", null, null, null, null, null, null, null, null, null, null },
    { "Option 8", null, null, null, null, null, null, null, null, null, null },
    { "Option 9", null, null, null, null, null, null, null, null, null, null }
};
```

Rather than explicitly typing each value of each element of the array I could of used a for loop and an inner for loop to set the values. However, I chose to do it the way illustrated above because it was easier to visualize each cell within the table and allow me to easily enter default values for cells for testing and debugging purposes.

Now there are two independent arrays setup containing column headers and cell values I can make the window or JFrame and start adding components to a container. I firstly setup a JFrame called frame and set the layout to be BorderLayout since this layout manager is easy to work with.

```
frame = new JFrame("Decision Support Tool");
frame.setLayout(new BorderLayout());
```

With the frame in place I could start implementing and setting up the JTable with the default values and column names using the method:

```
table = new JTable(data, columnNames);
```

The most important thing to do next were to put the table into a scrollpane in case the window was resized smaller than the physical table. This would ensure that the user could still view and edit all cells within the table inside a smaller window than expected.

```
JScrollPane pane = new JScrollPane(table);
```

After this stage the window is in place and viewable once some simple methods have been invoked to pack all of the components within the container and make the window viewable as well as setting a default window size and close operation.

```
frame.pack();
```
frame.setDefaultCloseOperation(EXIT_ON_CLOSE);
frame.setSize(1000, 305);
frame.setVisible(true);

The line highlighted in yellow is very important because it essentially tells the
window what to do when the window has been closed. Initially, I didn’t have this
attribute set and therefore every time I ran the program the spreadsheet process
remained running and it eventually caused my computer to run out of RAM.
Setting the default close operation ensures that the process is terminated and the
program closes properly and safely.

After the frame has been set and the table added to the frame I set up a very simple
toolbar which contains buttons to set the cut off threshold to be “less than” or
“greater than” the value in the selected cell of the table. A “reset” button will also
be present on the toolbar so the user can easily revert back to the tables previous
stage before a cut off value were set.

// make tool bar
JToolBar toolbar = new JToolBar("Toolbar", JToolBar.HORIZONTAL);

The toolbar is now setup and its default orientation is horizontal and at the top of
the window just below the menu bar. I can now add components to the toolbar,
such as, buttons, images, radio buttons etc…

// less than button and add to toolbar
JButton lessthan = new JButton("Less Than");
lessthan.setActionCommand("Less Than");
lessthan.setToolTipText("Set cut off value less than selected cell");
lessthan.addActionListener(this);
toolbar.add(lessthan);

The code above simply creates a JButton, adds a tooltip to the button and finally
adds the button to the toolbar. This code was used three times to setup the three
different buttons on the toolbar. A thing to note is the setActionCommand(…) method. This method sits there and waits for components to be clicked on the menu
bar or toolbar and fires the necessary methods once these components have been
clicked. For example, if the “less than” button were clicked on the toolbar then the
setActionCommand(…) method would have a string value of “less than” sent into
it and then the setCutOffValue method would be invoked with the operation being
less than. Similarly, if File -> Exit were invoked then the setActionCommand(…) would have the string value of “exit” sent into it and the method would know to
close the program because of the string value sent into the method. The code below
shows what the action listener method does.

public void actionPerformed(ActionEvent event)
{
    System.out.println("Item: "+ event.getActionCommand());
}
String eventPressed = event.getActionCommand();
if(eventPressed == "Exit") {
    System.exit(0);
} else if(eventPressed == "Less Than") {
    setCutOff(0);
} else if(eventPressed == "Greater Than") {
    setCutOff(1);
} else if(eventPressed == "Reset") {
    reset();
}

Phase 1 is now complete. The window is in place with all of the necessary components:

- Menu structure – File > Exit
- Toolbar – Less Than | Greater Than | Reset
- Spreadsheet table
- Scroll bars

I can move onto phase 2 and implement some functionality to set cut off thresholds and remove redundant cells.

**Phase 2**

With the main interface working as intended and all of the necessary components in place I can begin phase 2. Phase 2 is one method which takes an operation as its parameter of type integer.

private void setCutOff(int op) { ... }

It has 2 possible arguments that it will receive:

- 0 being the less than notation
- 1 being the greater than notation

The parameter is decided by the action listener method. If the “less than” button is clicked then the action listener fires the event to proceed with the cut off value method passing it the argument “0”.

Once the setCutOff(...) method has been invoked the initial thing to do is to set the cut off value to be the value in the currently selected cell.

try {
    cutoffval = Double.parseDouble(table.getValueAt(row, col).toString());
} catch (NumberFormatException nfe) {
    System.out.println("Invalid value: " + nfe);
}
This block of code gets the value in the selected cell and converts it to a string using the toString() method. The value in cells are always stored as Objects and therefore toString() needs to be invoked before the value can be parsed to be of a double data type. Obviously not everything in the cells are going to be number values and I’ve implemented a try, catch statement in the case of trying to convert strings or anything else illegal to a double value.

Now a cut off value has been set it’s possible to iterate through each row in the selected column and remove the rows which are outside of the cut off threshold. The pseudo code on page 23 was a good groundwork for implementing the algorithm which deals with this.

```java
// remove all values that do not meet the cut off value
for(row = 0 ; row < 9; row++) {
    // System.out.println("row: "+ row + "col: "+
    table.getSelectedColumn());
    // get values in the table and convert them to doubles for evaluating
    try {
        val = Double.parseDouble(table.getValueAt(row,col).toString());
        // if op = 0 then check if val < cutoffval else if op = 1 check if val > cutoffval
        if(op == 0 && val < cutoffval) {
            // remove row where val < cutoffval
            for(j = 0 ; j < 10 ; j++) {
                table.setValueAt(null, row, j);
            }
        } else if(op == 1 && val > cutoffval) {
            // remove row where val > cutoffval
            for(j = 0 ; j < 10 ; j++) {
                table.setValueAt(null, row, j);
            }
        }
    } catch (NumberFormatException nfe) {
        System.out.println("Incorrect format: "+ nfe);
    } catch (Exception e) {
        System.out.println("Invalid value: " + e);
    }
}
```

The code above will remove rows from the spreadsheet which are outside of the cut off threshold. Ideally, I would have liked to keep the rows which are outside of the
cut off threshold within the spreadsheet but simply grey out the rows instead of entirely removing these rows from the table. I feel graying out the rows would have been a better idea because they are still visible to user in case the user wishes to see previous alternatives which are outside of the cut off threshold. Removing the rows completely doesn’t allow this.

In order to grey out cells within the spreadsheet a new class had to be implemented which overrides the default table cell renderer.

Public CellRenderer extends JLabel implements TableCellRendererComponent { ... }

Overriding the cell renderer class theoretically allows you to format cells within the spreadsheet in terms of specifying cell colour, text colour, size, components within a cell etc… However, I had problems implementing such as class and decided to stay with the alternative way of removing the redundant rows from the table but allowing the user to revert back to the table state with the use of a reset button.

Phase 3

Since I wasn’t able to implement the table in such a way that cells are greyed out which are not in range of the cut off threshold I was forced to implement some mechanism which allows the user to revert back to the original state of the table. I chose to have a simple “reset” button on the toolbar that allows the user to go back to the original state.

Firstly, I set the reset button to be disabled or inactive when the program initially starts up. This is because there is no state to save. The state can only be saved once a cut off value has been set. As soon as the user clicks the “less than” or “greater than” buttons on the toolbar the state of the spreadsheet is instantly saved inside an independent array which can be reverted back to at any point. When the program starts up the line of code below is executed:

reset.setEnabled(false);

Once the cut off value method has been invoked the code below is executed which sets the reset button to be enable and saves the current state of the table into the independent array.

// enable reset button
reset.setEnabled(true);

// save current table state
if(count == 0) {
    for(i = 0; i < 9; i++) {
        for(j = 0; j < 10; j++) {
            d[i][j] = table.getValueAt(i,j);
        }
    }
}
I have a “count” variable in place so that the table state is only backed up only once. If the count variable is not equal to 0 then the block of code is skipped and the table state is not stored in the backup array. Otherwise, if the count variable value is equal to 0 then a loop and inner loop are present to store every value of each cell into the relevant array location, for example, cell (0,0) will be stored inside array location (0,0) and the same applies to every cell within the spreadsheet table.

Once the reset button method has been invoked the original table state is restored with all values placed in the relevant column and in the relevant cells.

```java
public void reset() {
    // populate table with saved state
    int i, j;
    for(i = 0; i < 9; i++) {
        for(j = 0; j < 10; j++) {
            table.setValueAt(d[i][j], i, j);
        }
    }

    // set global count var to 0 to denote the table has been reset
    count = 0;

    // disable reset button because its just been pressed
    reset.setEnabled(false);
}
```

The block of code above iterates through the backup array and transfers the data within each location back into each cell and reset the count variable to 0 in case the user changes any of the values before setting a cut off value and again this will back up the original state of the spreadsheet table. Additionally, the reset button is set back to its original state of being disabled until the user sets a new cut off value.

**Final interface**

Now the spreadsheet is implemented and has some functionality the final interface can be seen in Appendix C. I feel that the interface design has gone very well in terms of how the final interface looks with regards to the interface design sketches in Appendix A.
6. Testing

Below is a test plan of the spreadsheet in order to ensure everything works as intended and ensure there are no major bugs or inconsistencies. Some tests will have an accompanied screenshot which will appear in Appendix C.

<table>
<thead>
<tr>
<th>#</th>
<th>Testing</th>
<th>Description</th>
<th>Expected</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Value entry</td>
<td>Enter a value into any cell of the spreadsheet</td>
<td>Will work</td>
<td>Cell takes value</td>
</tr>
<tr>
<td>2</td>
<td>Edit column header</td>
<td>Double click any column header and enter new value</td>
<td>Will work</td>
<td>Header takes new value</td>
</tr>
<tr>
<td>3</td>
<td>Sort ascending</td>
<td>Enter values 1-10 in rows 9-1 of col 2</td>
<td>Will sort ascending</td>
<td>Sorts ascending</td>
</tr>
<tr>
<td>4</td>
<td>Sort descending</td>
<td>Enter values 1-10 in rows 1-9 of col 1</td>
<td>Will sort descending</td>
<td>Sorts descending</td>
</tr>
<tr>
<td>5</td>
<td>Move column</td>
<td>Move column 4 into column 2 position</td>
<td>Will move column</td>
<td>Column moved</td>
</tr>
<tr>
<td>6</td>
<td>Cut off value less than</td>
<td>Set a cut off value to be less than 4</td>
<td>Rows less than 4 will be deleted</td>
<td>Rows deleted</td>
</tr>
<tr>
<td>7</td>
<td>Cut off value greater than</td>
<td>Set cut off value to be greater than 4</td>
<td>Rows greater than 4 will be deleted</td>
<td>Rows deleted</td>
</tr>
<tr>
<td>8</td>
<td>Reset button</td>
<td>Reset the spreadsheet to its original state</td>
<td>Original state will be restored</td>
<td>Original state restored</td>
</tr>
<tr>
<td>9</td>
<td>File menu</td>
<td>Click file &gt; exit</td>
<td>Program will close</td>
<td>Program closes</td>
</tr>
<tr>
<td>10</td>
<td>Cell value</td>
<td>Enter “hello” into column 2 row 4 and click “less than”</td>
<td>Numberformat- exception will be caught</td>
<td>Exception handled</td>
</tr>
<tr>
<td>11</td>
<td>Resizing</td>
<td>Resize window so its smaller than the default size</td>
<td>Table will appear inside a scrollpane</td>
<td>Scrollpane appears</td>
</tr>
<tr>
<td>12</td>
<td>Cut off value</td>
<td>Select an empty cell and click “less than”</td>
<td>Numberformat- exception will be handled</td>
<td>Exception handled</td>
</tr>
<tr>
<td>13</td>
<td>Cut off value</td>
<td>Select an empty cell and click “greater than”</td>
<td>Numberformat- exception will be handled</td>
<td>Exception handled</td>
</tr>
<tr>
<td>14</td>
<td>Reset button</td>
<td>Ensure reset button appears inactive once pressed</td>
<td>Reset button will be inactive once pressed</td>
<td>Reset button inactive</td>
</tr>
<tr>
<td>15</td>
<td>Toolbar</td>
<td>Drag the toolbar off of the spreadsheet window</td>
<td>Toolbar will appear as an independent window</td>
<td>toolbar floats</td>
</tr>
<tr>
<td>16</td>
<td>Toolbar</td>
<td>Drag the toolbar onto the left side of window</td>
<td>Toolbar will dock on left side</td>
<td>Toolbar docks</td>
</tr>
</tbody>
</table>
As expected, there is a bug which doesn’t place cell values in the correct column once a cut off value has been set and a column has been shifted to another location. This is because the state of the spreadsheet is stored inside a separate array and only the cell values are reverted back instead of column headers as well. A simple fix to this would simply be resetting table column headers as well as cell values.

Test 22 within the test plan is a problem I was faced with and unable to overcome using the default table models way of sorting data within the table. Unfortunately, when a column is sorted it takes white space into consideration. For example, if the spreadsheet had value 1 in location (column 2, row 1) and value 2 in location (column 2, row 2) and then sorted descending the values 2 and 1 would be at the very bottom of the spreadsheet in column 2 and lots of white space at the top in each of the other columns.

A way I could fix this is to implement my own sorting buttons and place them on the toolbar rather than clicking the column header. This would make sorting be much more sophisticated. There are many different sorting algorithms, such as, bubble sort and quick sort.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Resize columns</td>
<td>Resize any column in the spreadsheet</td>
<td>Any column will resize</td>
<td>Column resizes</td>
</tr>
<tr>
<td>18</td>
<td>Toolbar</td>
<td>Drag the toolbar onto the bottom of window</td>
<td>Toolbar will dock on the bottom</td>
<td>Toolbar docks</td>
</tr>
<tr>
<td>19</td>
<td>Toolbar</td>
<td>Drag the toolbar onto the right of window</td>
<td>Toolbar will dock on the right</td>
<td>Toolbar docks</td>
</tr>
<tr>
<td>20</td>
<td>Toolbar</td>
<td>Drag the toolbar to the top of window</td>
<td>Toolbar will dock in default position</td>
<td>Toolbar docks</td>
</tr>
<tr>
<td>21</td>
<td>Cell values</td>
<td>Set a cut off value, move a column and reset the table</td>
<td>Values will appear in the incorrect columns</td>
<td>Values appear in wrong col</td>
</tr>
<tr>
<td>22</td>
<td>Sorting</td>
<td>Enter different values in different rows of the same column and sort descending</td>
<td>Column will sort ascending but take white space into consideration</td>
<td>Columns sort with white space</td>
</tr>
</tbody>
</table>
7. Results

The program is now fully implemented and tested so I’m able to apply some real world examples to the spreadsheet using different Fast and Frugal Heuristics. I will use an example such as buying a new house with attributes such as, price, number of bedrooms, garden size, and distance from town centre. All data will be real data and taken from an estate agents website. I will then apply different heuristics and identify and analyse the outcome.

I will be using the data as it appears in the table in Figure 28 in Appendix C and applying different heuristics to find the most appropriate house to buy. The data populated within the table was taken from Strakers estate agents back in January of 2010. All data within the table is real data and will be analysed with different heuristics.

Lexicographic (Take-The-Best)

During my literature survey I researched many different fast and frugal heuristics. Take The Best is a heuristic which essentially “takes the best” attribute and chooses this alternative as the most appropriate. If two or more alternatives have the same value for its attribute then the next attribute is taken and considered, again taking the best value. This process is repeated until one alternative wins with the “best” valued attribute.

The first stage the heuristic requires is to rank each attribute in terms of priority. This means that each column should be placed from start to end with the most important attribute first, second most attribute second etc... until all attributes are ranked. For the purpose of the project I will assume that living room space is the biggest factor, followed by price, followed by distance from town centre.

With the data in the table above there is a winner with the “best” living room space. If another properly had 24.5 meters squared of living room space then the price attribute would be taken into consideration.

Another example using the take-the-best heuristic is to find the property with the “best” number of rooms and if there are two or more attributes with the
same value then take the price of the house into consideration where the “best” price is the lowest price.

There are two properties with 4 bedrooms so the next highest priority attribute will be taken for further consideration, which in this example is Price. The price column will then be sorted ascending and property 2 and 3 will be observed for the value which is lowest.

Property 2 is the winner since it has the same number as bedrooms as property 3 but is priced at £156,500 and property 3 is priced at £193,995.

**Elimination by Aspects**

As described in my literature survey, this type of heuristic makes use of cut off values and sets a threshold in which alternatives must be inside of otherwise they are thrown out of consideration. Again, each attribute is ranked in priority. For the purpose of this example I will set number of bedrooms to be the highest priority, followed by price, followed by living room space and finally distance from town centre. I will use the spreadsheet to find the ideal house with 3 or more bedrooms, less than £156,500 and have a living room space more than 20 square meters and close to the town centre.
The screenshot above is the initial data view with attributes ranked in terms of importance. The next step is to set a cut off value of “less than” 3 bedrooms. Simply click a property with 3 bedrooms and click “less than”. This will remove all values outside of the cut off threshold of 3 or more.

Once the threshold has been set for the number of bedrooms you can move onto the price column since its ranked 2nd on importance. Click a property which is the most you can afford. In this case the maximum value is £156,500. Click the cell containing this value and press “greater than”; this will remove all properties which cost more than the cut off threshold.

Now we are left with three properties. The next most important attribute is living room space which has to be greater than 20 square meters. Clicking the closest value to greater than 20 is 24.37. Clicking the cell with this value and pressing “less than” will set a cut off threshold which will remove all alternatives with living room space less than 24.37 meters squared.
The next most important attribute is distance from town centre. The property has to be as close as possible, there is a clear winner with 2.9 so simply click the cell containing the value 2.9 and click “greater than” and this will leave you with one property which is considered the winner and most appropriate property based on criterion.

This heuristic is very mechanical in terms of a process you follow every time to be left with one single alternative. You iterate through each attribute and set a cut off value until a single alternative is left over and this is the winning alternative which meets a specific criterion.

**Satisficing (Conjunctive)**

This type of heuristic sets cut off thresholds on all important attributes and throws out all alternatives outside of this threshold. A strategy is then used to select good enough alternatives for further consideration. Essentially, this heuristic throws out all alternatives outside the cut off value and leaves a set of alternatives that are “good enough” to be used for selection.

For the purpose of this example we will set the important attributes to be this order:

| Price | Drive way | No. Bedrooms | … | … |
The attribute order has been set and we can select a cut off value for price, which we will set to greater than £171,000. All properties greater than £171,000 will be removed from the spreadsheet. The second cut off threshold will be placed on Drive way. The property must have a drive way so the threshold will be “less than” 1 (1 = drive way exists, 0 = drive way doesn’t exist). The final important attribute is number of bedrooms. We will set the threshold to be less than three. Every property with less than 3 bedrooms will be thrown out of consideration. We will be left with a table which looks like the one below once all cut off values have been set.

Property 1, 2 and 7 are all considered “good enough” properties to purchase because of what the rules of the heuristic tells us. All important attributes had cut off thresholds set and we are left with a set of alternatives which are considered good enough to purchase.

**Disjunctive**

This heuristic is very similar to conjunctive; however, it has a slight difference. Instead of leaving a set of “good enough” alternatives you pick a set of alternatives that are very good on one attribute and then consider each alternative further. This heuristic appears to integrate some form of conjunctive as well as lexicographic heuristics. The final step is sorting an attribute to find the best alternative with the set of alternatives which are left over from setting acceptable cut off values. If we perform the exact same steps as we did in the conjunctive heuristic and be left with data in the table on the next page we can then apply the final stage of this heuristic to find an appropriate alternative.
Instead of leaving it here as we did with the conjunctive heuristic we need to choose an attribute that we consider to be very high on importance, such as, number of bedrooms. If we sort this column descending then we are left with a set of alternatives ranked from best to worst based on the acceptable cut off values.

Now the “No. Bedrooms” column is ranked we can take each alternative for further consideration bearing in mind that property 2 is the best and most appropriate property, with property 7 in second place and property 1 in 3rd place.
8. Conclusion

I feel the project went rather well considering I had absolutely no knowledge about the subject before I undertook it. Once the research phase began I got more and more interested in the subject which motivated me to read more papers and want to learn more about the topic. My knowledge on the subject of decision making has definitely improved and being able to implement a piece of software which supports decision making is indeed a personal accomplishment. My objectives listed in the introduction section states that I would like a greater understanding of how decisions are made and the factors involved which drives humans to make these decisions, in addition to implementing a piece of software which supports a list of heuristics and will help with supporting a decision at hand. I feel what I’ve done is something I can be proud of and take away the experience of undertaking a successful personal project.

In terms of time scale I feel the project went fairly well. Obviously, milestones overran somewhat which is to be expected. As I stated before, the topic is very new to me and the initial research phase took longer than anticipated in order to get a much broader understanding of what factors affected how decisions are made and how humans came to a final conclusion about a specific decision at hand.

Implementing the spreadsheet took a while longer than anticipated too. I came across some slight problems when implementing the functionality to set cut off values and the way in which the data were displayed once a cut off threshold had been set. My initial plan were to grey out cells which were out of range of the cut off threshold, however, this didn’t go to plan. I understand there is a class within Java called TableCellRendererComponent which deals with formatting and dealing with single cells within a spreadsheet and applying colours, text fonts, text colours and styles etc… to each cell. I attempted to implement a class which used the TableCellRendererComponent class to grey out specific cells and rows out of range of the cut off value but with no success. I opted to simply remove the rows completely from the spreadsheet rather than grey them out or strikethrough some text within a cell. I agree it’s not the most sophisticated way of completing the task at hand but it’s a work around nonetheless. Spending so much time on trying to implement a class to grey out cells within the table put me a little behind schedule which is why I chose to carry out a much simpler technique of dealing with redundant cells when they were out of range of the cut off threshold.

Another issue I had was with the JTable API itself. I had never worked with a JTable component in Java and found it very difficult to implement a class which dealt with allowing a column header to be editable so I chose to find an open source class which had been implemented already by somebody with a greater knowledge than me on the subject. Obviously, I tweaked the class somewhat to allow columns to be sortable ascending and descending as well as being able to drag and drop columns into different locations on the spreadsheet table.

Designing sketches of the spreadsheet interface was a huge help and advantage. Rather than guessing where to place components and how they would be displayed I had a clear plan to follow. Additionally, sketching the interface makes you think
about what will work and what won’t work. For example, my initial design sketch required the user to go through an unnecessary process of setting a cut off value. Opening a dialog box and manually entering a value instead of simply getting the value in the current cell of the spreadsheet seemed a bit unnecessary and the user would spend much more time on setting a cut off threshold than they would on my second interface sketch. Without careful planning and consideration for each component within the spreadsheet the project can take much more time than intended and can potentially run over or completely fail. The design phase of the project is very valuable and should be performed thoroughly in order to get the most of your piece of software.

With more time I would have liked to implement the TableCellRendererComponent class to deal with graying out and making rows outside of the cut off value threshold inactive. I feel this is a much better way in terms of interface design and HCI rather than completely removing the rows from the table. If a user wished to change a particular attribute of a row which has been removed from the spreadsheet then it isn’t possible. However, if they cell were simply greyed out then it is still possible to view and edit the cell and possibly reset the cut off threshold.

Another feature I would have liked to implement into the spreadsheet is the ability to save and open files rather than having to input the data into each cell every time the spreadsheet is opened. There is a lot of functionality missing from the spreadsheet such as being able to:

- Format text.
  - Bold, italic, underlined, font colours, font style.
- Format cells.
  - Cell colour, merge cells, set cell types such as number, strings, currency etc...
- Save and open files.
- Integrate more heuristics within the spreadsheet since a couple are missing.
- Print the data within spreadsheet.
- Cut, copy and paste
- Sub sorting

Although lots of functionality is missing from the spreadsheet the foundations are in place and the main thing is that the spreadsheet is working as intended and the missing features are purely interface enhancements to make it visually more appealing to the user.
9. Bibliography


10. Appendix A

**Figure 1:** Main spreadsheet interface

**Figure 2:** Dialog box to set a cut off value for a selected column.
**Figure 3: Main Spreadsheet Interface.**

<table>
<thead>
<tr>
<th>ALTERNATIVES</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTION 1</td>
<td>1</td>
<td>7</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>OPTION 2</td>
<td>2</td>
<td>8</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>OPTION 3</td>
<td>3</td>
<td>9</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>OPTION 4</td>
<td>4</td>
<td>10</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>OPTION 5</td>
<td>5</td>
<td>11</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>OPTION 6</td>
<td>6</td>
<td>12</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4: Displaying data after cut off value has been set to 'Less than' selected cell.**

**Selected cell = (3,2)**

Grey out cells 'Less than' or 'Greater than' cut off value.
<table>
<thead>
<tr>
<th>ALTERNATIVUS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTION 1</td>
<td>1</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>OPTION 2</td>
<td>2</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>OPTION 3</td>
<td>3</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>OPTION 4</td>
<td>4</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>OPTION 5</td>
<td>5</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>OPTION 6</td>
<td>6</td>
<td>12</td>
<td>18</td>
</tr>
</tbody>
</table>

**SELECTED CELL = (3,2)**

**FIGURE 5**: DATA BEFORE CUT OFF VALUE HAS BEEN SET.

<table>
<thead>
<tr>
<th>ALTERNATIVUS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTION 1</td>
<td></td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>OPTION 2</td>
<td>2</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>OPTION 3</td>
<td>3</td>
<td>9</td>
<td>15</td>
</tr>
</tbody>
</table>

**FIGURE 6**: DATA AFTER CUT OFF VALUE HAS BEEN SET TO 'GREATER THAN' FROM FIGURE 5.

**ROWS 4, 5, 6 HAVE DISAPPEARED BECAUSE THEY WERE OUTSIDE THE CUT OFF VALUE THRESHOLD.**
11. Appendix B

Figure 12: HTA diagram on performing a cut off value to a specific cell

Figure 13: HTA diagram on performing a cut off value to a specific cell v.2
Figure 14: UML Class diagram
12. Appendix C

Figure 15: Final interface with simple cell values

Figure 16: Interface after cut off value has been set to “less than” on currently selected cell

Figure 17: Interface after cut off value has been set to “greater than” on currently selected cell
Figure 18: Test 1 of the test plan

Figure 19: Test 2 of the test plan

Figure 20: Test 3 of the test plan

Figure 21: Test 4 of the test plan
Figure 22: Test 5 of the test plan

<table>
<thead>
<tr>
<th>Alternatives</th>
<th>Less Than</th>
<th>Greater Than</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 23: Test 6 of the test plan

<table>
<thead>
<tr>
<th>Alternatives</th>
<th>Less Than</th>
<th>Greater Than</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 24: Test 7 of the test plan

<table>
<thead>
<tr>
<th>Alternatives</th>
<th>Less Than</th>
<th>Greater Than</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 25: Test 8 of the test plan

<table>
<thead>
<tr>
<th>Option 1</th>
<th>Option 2</th>
<th>Option 3</th>
<th>Option 4</th>
<th>Option 5</th>
<th>Option 6</th>
<th>Option 7</th>
<th>Option 8</th>
<th>Option 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>24</td>
<td>26</td>
</tr>
<tr>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td>29</td>
<td>31</td>
<td>33</td>
<td>35</td>
</tr>
</tbody>
</table>
Figure 26: Test 15 of the test plan

Figure 27: Test 16 of the test plan
Figure 28: Spreadsheet in use with a real world example of choosing the most appropriate house to buy

<table>
<thead>
<tr>
<th>Alternatives</th>
<th>Living room size (m sq)</th>
<th>Distance from town (km)</th>
<th>Price</th>
<th>No. Bedrooms</th>
<th>Drive Way</th>
<th>Front Garden</th>
<th>Back Garden</th>
<th>Garage</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property 1</td>
<td>24.5</td>
<td>4.1</td>
<td>149905</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Property 2</td>
<td>24.37</td>
<td>2.8</td>
<td>156000</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Property 3</td>
<td>17.28</td>
<td>4.2</td>
<td>193905</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Property 4</td>
<td>15.45</td>
<td>4.3</td>
<td>134905</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Property 5</td>
<td>16.4</td>
<td>3.5</td>
<td>139905</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Property 6</td>
<td>15.11</td>
<td>4.3</td>
<td>149905</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Property 7</td>
<td>15.78</td>
<td>3</td>
<td>171000</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Property 8</td>
<td>15.75</td>
<td>3.5</td>
<td>134905</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Property 9</td>
<td>11.72</td>
<td>4.2</td>
<td>109900</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
13. Appendix D

User Documentation

Opening the program

Simply double click the executable JAR file called Spreadsheet.jar. Once double clicked the program will open and you can start entering data into cells.

What does the program do?

The program will assist with any sort of decision you are faced with. Simply input the different choices in the first column and each attribute associated with the alternatives in the column headers. You can then enter values in cells and set cut off values for alternatives in order to remove alternatives which do not meet the cut off threshold.

What is a cut off value?

A cut off value is a value which isn’t in range of the value in the selected cell. For example, if the value in the selected cell is 5 and the cut off value is set to “less than”, then all rows with a value less than 5 in that given column will be removed from the spreadsheet.

How do I retrieve my data once it’s been removed from the table after setting a cut off value?

There is a button on the toolbar called “reset”, clicking this button will simply revert the table back to its initial state before a cut off value was set.

How can I move columns?

Moving columns couldn’t be simpler. Simply drag and drop columns into their respected position and they will be relocated accordingly within the spreadsheet.

Why is the cut off value not being set for a particular cell?

A cut off value can only be set if the value in the selected cell is considered a number value. Text cannot be evaluated using the “less than” or “greater than” operations.
How do I sort a column ascending or descending?

Simply clicking the column header will sort that particular column ascending or descending. There is a triangular icon which appears on the column header to display how the column is sorted.

![Figure 29: Price column sorted ascending](image)

![Figure 30: Price column sorted descending](image)

How can I edit the header text of a column?

Double clicking the header name of a column will allow text editing within the selected column header. Once the text has been set, simply press the “enter” button on the keyboard to finish off setting the header name for the column.

How do I close the spreadsheet?

Simply close the window using the traditional way of left clicking the “x” in the top corner of the dialog box (depending on your operating system the “x” is generally on the top right). Alternatively, you can close the program in the menu provided called File and then click Exit.
import java.awt.*;

public class EditableHeaderEx extends JFrame implements ActionListener {

    private JFrame frame;
    JTable table;
    double cutOffval;
    JButton reset;
    int count = 0;
    Object[][] d = {
EditableReaderEx.java

private void setCutoff(int op) {
    int row = table.getSelectedRow(), col = table.getSelectedColumn(), i, j;
    double val;

    // enable reset button
    reset.setEnabled(true);

    // save current table state
    if(count == 0) {
        for(i = 0; i < 9; i++) {
            for(j = 0; j < 10; j++) {
                d[i][j] = table.getValueAt(i, j);
            }
        }
        // set count to 1 to denote that the initial table state has been saved
        count = 1;
    }

    // set cut off val to be the value in the selected cell
    try {
        cutoffval = Double.parseDouble(table.getValueAt(row, col).toString());
    } catch (NumberFormatException nfe) {
        System.out.println("Invalid value: " + nfe);
    } catch (Exception e) {
        System.out.println("Invalid value: " + e);
    }

    // remove all values that do not meet the cut off value
    for(row = 0; row < 9; row++) {
        // System.out.println("row: " + row + " col: " + table.getSelectedColumn()));
        // get values in the table and convert them to doubles for evaluating
        try {
            val = Double.parseDouble(table.getValueAt(row, col).toString());
            // if op = 0 then check if val < cutoffval else if op = 1 check if val > cutoffval
            if(op == 0 && val < cutoffval) {
                // remove row where val < cutoffval
                for(j = 0; j < 10; j++) {
                    table.setValueAt(null, row, j);
                }
            } else if(op == 1 && val > cutoffval) {
                // remove row where val > cutoffval
                for(j = 0; j < 10; j++) {
                    table.setValueAt(null, row, j);
                }
            }
        } catch (NumberFormatException nfe) {
            System.out.println("Incorrect format: " + nfe);
        } catch (Exception e) {
            
        }
    }

    // debugging purposes
    System.out.println("cutoff = " + cutoffval);
    System.out.println("col = " + table.getSelectedColumn());
private void makeFrame() {
    // set col names
    String[] columnNames = {"Alternatives", "m", "m^2", "m^3", "m^4", "m^5", "m^6", "m^7", "m^8", "m^9", "m^10"};
    // set 10 rows of empty cells
    Object[][] data = {
        ["Option 1", 1, 10, 19, null, null, null, null, null, null, null],
        ["Option 2", 2, 11, 20, null, null, null, null, null, null, null],
        ["Option 3", 3, 12, 21, null, null, null, null, null, null, null],
        ["Option 4", 4, 13, 22, null, null, null, null, null, null, null],
        ["Option 5", 5, 14, 23, null, null, null, null, null, null, null],
        ["Option 6", 6, 15, 24, null, null, null, null, null, null, null],
        ["Option 7", 7, 16, 25, null, null, null, null, null, null, null],
        ["Option 8", 8, 17, 26, null, null, null, null, null, null, null],
        ["Option 9", 9, 18, 27, null, null, null, null, null, null, null]
    };

    frame = new JFrame("Decision Support Tool");
    frame.setLayout(new BorderLayout());
    // set up data table
    table = new JTable(data, columnNames);
    DefaultTableModel columnModel = table.getColumnModel();
    table.setRowSorter(new TableRowSorter(columnModel));
    table.setRowHeight(20); // set row height to 20px
    table.getColumnModel().setPreferredWidth(new Dimension(20, 25)); // set column
    // set and add components to scrollpane
    JScrollPane pane = new JScrollPane(table);
    pane.addBallon(pane, BorderLayout.CENTER);
    // add menu and toolbar
    makeMenuBar(frame);
    makeToolBar(frame);
    // display window
    frame.pack();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(1000, 300);
    frame.setVisible(true);
}

public void makeMenuBar(JFrame frame) {
    // main menu bar
    JMenuBar menuBar = new JMenuBar();
```java
EditableReaderEx.java

frame.setJMenuBar(menuBar);

// file menu
JMenu fileMenu = new JMenu("File");
menuBar.add(fileMenu);

// file > exit
JMenuItem exitItem = new JMenuItem("Exit");
extItem.addActionListener(this);
fileMenu.add(exitItem);

public void makeToolBar(JFrame frame)
{
    // make toolbar
    JToolBar toolbar = new JToolBar("Toolbar", JToolBar.HORIZONTAL);

    //Label label = new JLabel("Set cut off value ");
    //toolbar.add(label);

    // less than button and add to toolbar
    JButton lessThan = new JButton("Less Than");
    lessThan.setActionCommand("Less Than");
    lessThan.setToolTipText("Set cut off value less than selected cell");
    toolbar.add(lessThan);

    // greater than button and add to toolbar
    JButton greaterThan = new JButton("Greater Than");
    greaterThan.setActionCommand("Greater Than");
    greaterThan.setToolTipText("Set cut off value greater than selected cell");
    toolbar.add(greaterThan);

    // separator
    toolbar.addSeparator();

    // reset button
    JButton reset = new JButton("Reset");
    reset.setActionCommand("Reset");
    reset.setToolTipText("Reset table");
    reset.addActionListener(this);
    toolbar.add(reset);

    // add to content pane
    frame.add(toolbar, BorderLayout.NORTH);
}
```
import java.util.*;

public class EditableHeader extends JTableHeader
    implements CellEditorListener
{
    public int HEADER_ROW = -1;
    transient protected int editingColumn;
    transient protected TableCellEditor cellEditor;
    transient protected Component editorComp;

    public EditableHeader(TableModel columnModel) {
        super(columnModel);
        setReorderingAllowed(true);
        cellEditor = null;
        recreateTableColumn(columnModel);
    }

    public void updateUI() {
        setUI(new EditableHeaderUI());
        repaint();
        invalidate();
    }

    protected void recreateTableColumn(TableModel columnModel) {
        TableModel[] oldCols = new TableModel[columnModel.getColumnCount()];
        TableModel[] newCols = new EditableHeaderTableColumn[columnModel.getColumnCount()];
        for (int i = 0; i < columnModel.getColumnCount(); i++) {
            oldCols[i] = columnModel.getColumn(i);
            newCols[i] = new EditableHeaderTableColumn(i);
            newCols[i].copyValues(oldCols[i]);
        }
        columnModel.removeColumn(oldCols);
        columnModel.addColumn(newCols);
    }

    public boolean editCellAt(int index) {
        return editCellAt(index);
    }

    public boolean editCellAt(int index, EventObject e) {
        if (cellEditor != null && cellEditor.stopCellEditing()) {
            return false;
        }
        if (getCellEditor(index) != null) {
            return false;
        }
        TableCellEditor editor = getCellEditor(index);
        if (editor != null && editor.isCellEditable(e)) {
public boolean isCellEditable(int index) {  
    if (getCellReorderingAllowed()) {  
        return true;  
    }  
    int columnIndex = columnModel.getColumnIndex(index, getModelIndex());  
    EditableHeaderTableColumn col = (EditableHeaderTableColumn) columnModel.getColumn(columnIndex);  
    return col.isHeaderEditable();  
}  

public TableCellEditor getCellEditor(int index) {  
    int columnIndex = columnModel.getColumnIndex(index, getModelIndex());  
    EditableHeaderTableColumn col = (EditableHeaderTableColumn) columnModel.getColumn(columnIndex);  
    return col.getCellEditor();  
}  

public void setCellEditor(TableCellEditor newEditor) {  
    TableCellEditor oldEditor = cellEditor;  
    cellEditor = newEditor;  
    // firePropertyChange  
    if (oldEditor != null && oldEditor instanceof TableCellEditor) {  
        ((TableCellEditor)oldEditor).removeCellEditorListener((CellEditorListener) this);  
    }  
    if (newEditor != null && newEditor instanceof TableCellEditor) {  
        ((TableCellEditor)newEditor).addCellEditorListener((CellEditorListener) this);  
    }  
}  

public Component prepareEditor(TableCellEditor editor, int index) {  
    Object value = columnModel.getColumn(index).getHeaderValue();  
    boolean isDefaulted = true;  
    int row = HEADER_ROW;  
    JTable table = getTable();  
    Component comp = editor.getTableCellEditorComponent(table, value, isDefaulted, row, index);  
    if (comp instanceof JComponent) {  
        ((JComponent)comp).setNextFocusableComponent(this);  
    }  
    return comp;  
}  

public TableCellEditor getCellEditor() {  
    return cellEditor;  
}
```java
public Component getEditorComponent() {
    return editorComp;
}

public void setEditingColumn(int aColumn) {
    editingColumn = aColumn;
}

public int getEditingColumn() {
    return editingColumn;
}

public void removeEditor() {
    TableCellEditor editor = getCellEditor();
    if (editor != null) {
        editor.removeCellEditorListener(this);
        requestFocus();
        remove(editorComp);
        int index = getEditingColumn();
        Rectangle cellRect = getHeaderRect(index);
        setCellEditor(null);
        setEditingColumn(-1);
        editorComp = null;
        repaint(cellRect);
    }
}

public boolean isEditing() {
    return (cellEditor == null) ? false : true;
}

// CellEditorListener
//
public void editingStopped(ChangeEvent e) {
    TableCellEditor editor = getCellEditor();
    if (editor != null) {
        Object value = editor.getCellEditorValue();
        int index = getEditingColumn();
        columnModel.getColumn(index).setHeaderValue(value);
        removeEditor();
    }
}

public void editingCanceled(ChangeEvent e) {
    removeEditor();
}

// public void setReorderingAllowed(boolean b) {
//    reorderingAllowed = b;
//    //
```
package EditableHeaderTableCell;

import javax.swing.*;

public class EditableHeaderTableCell extends TableCell {
    protected TableCell headerEditor;
    protected boolean isHeaderEditable;

    public EditableHeaderTableCell() {
        setHeaderEditor(createDefaultHeaderEditor());
        isHeaderEditable = true;
    }

    public void setHeaderEditor(TableCell headerEditor) {
        this.headerEditor = headerEditor;
    }

    public TableCell getHeaderEditor() {
        return headerEditor;
    }

    public void setHeaderEditable(boolean isEditable) {
        isHeaderEditable = isEditable;
    }

    public boolean isHeaderEditable() {
        return isHeaderEditable;
    }

    public void copyValues(TableCell base) {
        modelIndex = base.getModelIndex();
        identifier = base.getIdentifier();
        width = base.getWidth();
        minWidth = base.getMinWidth();
        setPreferredSize(base.getPreferredSize());
        maxWidth = base.getMaxWidth();
        headerRenderer = base.getHeaderRenderer();
        headerValue = base.getHeaderValue();
        cellRenderer = base.getCellRenderer();
        cellEditor = base.getCellEditor();
        isResizable = base.isResizable();
    }

    protected TableCell createDefaultHeaderEditor() {
        return new DefaultCellEditor(new JTextField());
    }
}
import java.awt.;

public class EditableHeaderUI extends BasicTableHeaderUI {
    protected MouseInputListener createMouseInputListener() {
        return new MouseInputHandler((EditableHeader)header);
    }

    public class MouseInputHandler extends BasicTableHeaderUI.MouseInputHandler {
        private Component dispatchComponent;
        protected EditableHeader header;

        public MouseInputHandler(EditableHeader header) {
            this.header = header;
        }

        private void setDispatchComponent(MouseEvent e) {
            Component editorComponent = header.getEditorComponent();
            Point p = e.getPoint();
            Point p2 = SwingUtilities.convertPoint(header, p, editorComponent);
            dispatchComponent = SwingUtilities.getDeepestComponentAt(editorComponent, p2.x, p2.y);
        }

        private boolean repostEvent(MouseEvent e) {
            if (dispatchComponent == null) {
                return false;
            }
            MouseEvent e2 = SwingUtilities.convertMouseEvent(header, e, dispatchComponent);
            dispatchComponent.dispatchEvent(e2);
            return true;
        }

        public void mousePressed(MouseEvent e) {
            if (!SwingUtilities.isLeftMouseButton(e)) {
                return;
            }
            super.mousePressed(e);

            if (header.getResizingColumn() == null) {
                Point p = e.getPoint();
                TableColumnModel columnModel = header.getColumnModel();
                int index = columnModel.getColumnIndexAtX(p.x);
                if (index != -1) {
                    if (header.editCellAt(index, e)) {
                        setDispatchComponent(e);
                        repostEvent(e);
                    }
                }
            }
        }

        public void mouseReleased(MouseEvent e) {
            super.mouseReleased(e);
            if (!SwingUtilities.isLeftMouseButton(e)) {
                return;
            }
            repostEvent(e);
            dispatchComponent = null;
        }
    }
}