2015-01-01

# Multi-expert Multi-criteria Decision Making

Joel Henderson
*University of Texas at El Paso*, henderjoel@gmail.com

MULTI-EXPERT MULTI-CRITERIA DECISION MAKING

JOEL HENDERSON

Department of Computer Science

APPROVED:

Martine Ceberio, Chair, Ph.D.

Vladik Kreinovich, Ph.D.

Stefano Bistarelli, Ph.D.

Luciana Garbayo, Ph.D.

Charles Ambler, Ph.D.
Dean of the Graduate School

*In the loving memory of*

*Gershon Ettinger*

# MULTI-EXPERT MULTI-CRITERIA DECISION MAKING

by

JOEL HENDERSON

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

July 2015

# Abstract

Expert analysis and decisions are highly valued assets in a wide variety of fields, from social services to grant funding committees. However, the use of experts can be prohibitive due to either lack of availability or cost. As such, it is desirable to be able to replicate such decisions. However, there are many obstacles that impede an accurate simulation of expert decisions. For example, despite looking at the same information, two experts may disagree on the decisions. In addition, a single expert may make inconsistent decisions across similar scenarios.

In this work, we focus on multi-criteria decision making and in particular, in the case of multiple experts (ME-MCDM). We examine how multi-criteria decision making techniques can address the multi-experts dimension of the problem, as well as how argumentation networks can inform us about how to aggregate the multiple experts' decisions.

Questions that we consider include: (1) How do we determine which expert(s) we should listen to in the event of a disagreement? (2) How do we detect inconsistencies in expert decisions, and (3) How do those inconsistencies impact who we should listen to?

We look at experts' decision data in the area of software quality assessment, and we analyze automated decisions that results from using non-discriminatory techniques (techniques that take all decisions – even conflicting – into account with the same importance). We reconsider these data, explore the use of argumentation networks, and reflect on the relevance of such approach. We report the results of our preliminary observations and we propose directions for future work.

# Table of Contents

# Chapter 1

# Introduction

Expert decisions play a very important role in many fields, including, but not limited to, software quality assessment. In software quality assessment for instance, a number of experts will judge a software according to a variety of criteria/features. These features can include understandability, flexibility, and effectiveness. **Each of these experts judges the software as a whole, and then a final judgement on the software is made based on these experts' determinations of the quality.**

For example, to decide if a student is academically at-risk or may pose a violent threat, a student risk assessment is performed. A student risk assessment is based on evaluating numerous criteria, including their personality, behavior, their relationship with their peers, their home life, and recent life events. In addition, when a student risk assessment is performed, it is usually done by an interdisciplinary threat-assessment team. This team can include people such as school-based personnel, social workers, and faith leaders [1]. However, because these people come from wildly different professional backgrounds, they are likely to disagree on various matters which might lead them to making very different recommendations for the at-risk student, or disagree if the student is even at-risk.

There are several major issues that make the expert decision-making process difficult. One of these is the fact that experts may be unable to come to a consensus due to major disagreements about the item they are assessing. We wish to be able to resolve these disagreements in such a way that is fair to all experts involved. Another major problem is that experts can often make wildly inconsistent judgements on multiple items, and we would like to take this inconsistency into account in the final decision-making process. Another major issue with expert decision-making involves predicting expert decisions in

advance. Convening a group of experts can often-times be very time consuming and costly. Having a machine that can predict, or even automate, expert decisions could potentially increase efficiency in these decision-making processes drastically.

Previously, these issues were addressed by identifying weights to be associated to each criterion or feature [17] or to groups of criteria or features, and by then aggregating these weights. This could be done for instance by extracting data known as fuzzy measures [9] from a non-additive measure known as the Choquet integral. With this process, the weights of criteria, as well as the weights of the intersection of these criteria were extracted from previously-assessed items. However, this process has a number of flaws. For one, the number of variables in the fuzzy measure that must be extracted grows exponentially with the number of criteria, and is therefore not scalable. Another issue is that this process weighs all expert evaluations equally. As stated before, experts can often contradict themselves and we wish to take this into account when evaluating their decisions in determining a final result. To resolve these issues, we attempt to solve this problem using a new semantic, the abstract argumentation framework.

An abstract argumentation framework [11] is a set of positions, or arguments, and a set of binary relations between these arguments in the form of attacks. This semantic is used to help formalize disagreements and conflicts. In addition to arguments and attacks, argumentation frameworks have subsets of arguments that meet certain criteria, known as extensions. For example, a conflict-free set is a set of arguments such that no argument in the set is attacked by another in the set. These extensions are useful because they help us find which arguments we should listen to, and which ones we should ignore. To help put these frameworks in a form that can be understood by a computer, and for the computer to help us find the various extensions of these frameworks, we use a special program known as the ConArg tool [7]. We use this tool extensively in our research to determine the quality of our model of this problem.

In this work, we attempt to apply abstract argumentation frameworks to multi-expert multi-criteria decision-making by creating a model that we believe covers the important

aspects of the decision-making process. These aspects include arguments that cover criteria assessment, total quality assessment, and the final judgement; as well as attacks that cover expert disagreements and contradictions. We then create sample data using this model, run it using the ConArg tool – an argumentation framework dedicated solver, and compare the results to that of fuzzy measure extraction. Finally, we analyze what advantages our model has over previous work – namely fuzzy-measure-based MEMCDM, and how this model can be improved in future work.

# Chapter 2

# Background

## 2.1 Multi-Criteria Decision Making (MCDM)

There are many cases where an expert or group of experts need to make a decision on something based on a set of given criteria. For example, a panel at a funding agency or at a scientific journal may decide to fund a project or to publish a paper depending on several factors, such as the value of the work, how detailed the information is, how well sourced it is, and how clear the writing is.

The situation identified above is called Multi-Criteria Decision-Making (MCDM) [24]. MCDM involves selecting one of several different alternatives, based on the individual assessment of a set of criteria that describe the alternatives. However, there are numerous problems that make comparing these alternatives difficult. The major issue is that various criteria may be non-comparable [24]. For example, when choosing between different cars, two different criteria may be price and mileage. How do you compare these two criteria? There exist several methods for addressing these issues.

### 2.1.1 Approaches to MCDM

If we are to predict and check the consistency of expert decisions, we need an algorithm that will give us the total quality decision based on the decisions of the individual criteria. The simplest way to do this is to represent the satisfaction level of the total quality as a weighted sum of the satisfaction levels of the individual criteria [13].

**Definition 1** *A* weighted sum *is a sum* $c_1 \times k_1 + \ldots + c_n \times k_n$ *where* $k_i$ *is the* $i$*th weighted*

*criterion and $c_i$ is the weight of $k_i$ such that $c_i \in [0, 1]$ and $c_1 + \ldots + c_n = 1$. The sum give us the total value of the item in question.*

This method is not only simple, but allows for each individual decision maker to alter the satisfaction levels and weights to match their personal preferences. However, this method assumes that all criteria are independent of one another, which is often not the case [8]. Therefore, we need another method that takes criteria dependence into consideration.

Fortunately, weighted sums aren't the only way to compare alternatives. Other methods exist, such as the Analytic Hierarchy Process (AHP) [21] and the Elimination et Choice Translating Reality method (ELECTRE) [20]. In this paper, we will focus on non-additive measures, or Fuzzy measures [9]. Fuzzy measures use variables that not only take each criterion into account, but the intersection of the criteria, as well. These variables are then aggregated with a function known as the Choquet Integral [15].

### 2.1.2 Fuzzy Measures

Since criteria dependency is something that needs to be considered, we need a way to measure the satisfaction levels of criteria that takes these dependencies into account. Fuzzy measures gives us such a way by measuring the satisfaction levels of unions of criteria as separate from the satisfaction levels of the individual criteria.

**Definition 2** *[9] Let $A$ be a finite set and $P(A)$ be the power set of $A$. A* fuzzy measure *is a set function $\mu : P(A) \to [0, 1]$ satisfying the following conditions.*

1. $\mu(\emptyset) = 0$
2. $\mu(A) = 1$
3. *if $X, Y \subseteq A$ and $X \subseteq Y$, then $\mu(X) \leq \mu(Y)$*

The idea is that rather than only giving each criteria an individual weight, we also give each combination of criteria an individual weight. However, the weights of two disjoint sets

do not necessarily add up to the weight of their union ($\mu(A) + \mu(B) \neq \mu(A \cup B)$). Hence, fuzzy measures are also known as non-additive measures.

The question then arises as to how one would use fuzzy measures to evaluate alternatives and make a judgement? Since they're non-additive, we can't use the weighted sum approach. The solution is to use what is known as the Choquet Integral.

**Definition 3** *Let $u_i(x_i)$ be the level of satisfaction associated with criterion $x_i$, sorted such that $u_1(x_1) \leq u_2(x_2) \ldots \leq u_n(x_n)$. Let $m(A_i)$ be the fuzzy measure on $A_i = [i, i+1, \ldots, n]$. Then the* Choquet integral *[15] on the set $X$ is given by: $\sum_{i=1}^{n} u_i(x_i) \cdot (m(A_i) - m(A_{i+1}))$*

However, fuzzy measures, due to being mapped to a power set, have exponential growth with the number of criteria we have. Can we find a better way to judge something based on multiple criteria?

## 2.2 Multi-Experts MCDM

In many real-world cases, such as software analysis or a thesis panel, it is not only a single expert evaluating criteria and making judgements. Often, there are multiple experts tasked with evaluation of the topic in question. In this case, each expert is going to evaluate it individually, and must come to a consensus in regards to the quality of the software, or other subject of discussion. This adds a whole new layer of complexity to the problem at hand.

What happens when we have a group of decision makers that must come to some sort of consensus? For example, a panel of experts making a decision on the quality of a piece of software. This is known as multi-expert multi-criteria decision making (MEMCDM) [23]. In these cases, we have several new problems that we must address. One problem is prediction of expert decisions. Gathering multiple experts to evaluate an item and come to a decision is very time-consuming and expensive. We wish to be able to predict expert decisions in order to quickly and efficiently determine their evaluations. Another problem is how to

handle expert disagreements and therefore come to a consensus in the first place. As stated before, different decision makers are going to have different preferences, and therefore come to different conclusions based on the same set of alternatives. We therefore must come up with a system to decide which experts to listen to in which situations. For this purpose we decide to use argumentation frameworks to model the problem.

### 2.2.1 Definition of the Problem

Multi-expert multi-criteria decision making is when a set of $n$ experts $\{E_1 \dots E_n\}$ must evaluate one or more items based upon a set of $i$ criteria $\{C_1 \dots C_i\}$. Each of these criteria is given a ranking $\{Q_1 \dots Q_k\}$, and a final decision about the item $Q_{total}$ is given based on these decisions [23].

With multiple experts judging the same thing, we run into several problems. The most apparent one is handling expert disagreements. Experts aren't going to agree 100 percent of the time, and when they do disagree, we must be ready to figure out who is right, or potentially compromise. This can mean not only resolving the conflict based on the current situation, but also evaluating the history of the experts. A veteran expert with a long history of good decisions should probably be trusted more than a rookie expert with a more spotted record. However, even then, people with good track records can be wrong, and we would be poorly served by always giving them the final say. So the question remains, "How do we come to a consensus?".

### 2.2.2 Voting Systems

We have successfully defined a system whereby multiple experts pass judgement on an item. However, in the end, we need to make some sort of final call on the item in question, using the experts' individual judgements to determine the result. This is where a voting system comes in.

A voting system [18] is any system that collects a series of individual preferences, and

outputs a total social preference. While there are a large number of possible voting systems, an "ideal" voting system should exemplify 6 qualities:

1. Universal Domain – All sets of valid individual preferences must yield a valid social preference

2. Completeness and Transitivity – The voting system must always return a definite output (completeness), and the output must always be consistent (transitivity).

3. Positive Association – If an individual places an item at a higher preference than before, that item should not become lower in the social preference order.

4. Independence of Irrelevant Alternatives – The social preference of two alternatives (A & B) should only depend on how voters rank those alternatives, and not a third alternative (C)

5. Non-Imposition – The social preference must not be independent of the individual preferences

6. Non-Dictatorship – The social preference cannot be determined solely by the preference of one individual

However, there is a problem with making an "ideal" voting system. Arrow's Impossibility Theorem states that there cannot exist a voting systems that exemplifies all qualities of an ideal voting system. [3]

While this is a good guide about what makes a good voting system, and why a "perfect" voting system doesn't exist, it doesn't answer what kind of voting systems are available, much less which one we should use. The first thing we consider is how many winners we can have. There are voting systems that support multiple winners, and others that support a single winner. Since we want a single final judgement on what the experts are evaluating, our focus is on single winner systems.

The most common form of a single winner voting system is first-past-the-post (FPTP), or plurality [18]. FPTP simply means that each voter chooses a single candidate, and the candidate with the most votes wins. This system can be expanded further with a runoff system. In a runoff system, there are multiple rounds of voting where candidates who receive less votes are eliminated until one candidate reaches a certain threshold of votes (usually 51%). In our model, multiple experts are choosing between multiple ranking to give the total quality of the item in question, and only one ranking can ultimately be given to the item. However, there are usually more than two rankings that can be given to the total quality. For this reason, we feel that a runoff voting system would be most suitable for our model.

Other single winner voting systems exist, such as ranked voting [25] and rated voting [25]. In ranked voting, voters rank candidates from most to least preferred. There are several methods of evaluating these ballots, such as instant-runoff voting. In rated voting, voters give each candidate a "grade", and the candidate with the best total grade is chosen.

### 2.2.3 Conflict Resolution and Other Challenges

Voting systems can allow us to come to a conclusion based on divergent opinions. However, we may not always want to do this. There are cases where we would rather try to resolve conflicts. For example, we may have a case where we have three experts evaluating a piece of software. Two of them think the software is "Good", but the third thinks it's only "Fair". Using a majority-rule voting system, we would naturally give the software a final ranking of "Good", but the third expert might have valid reasons for thinking it only deserves a "Fair" ranking.

One way conflicts could be resolved is to facilitate a mediated discussion where the experts come to a compromise about the item in question. The experts discuss amongst themselves the merits of the item and come to a compromise about its total quality. While this has the advantage of making everyone's voices heard and taken into account, it requires that we have access to the experts for these discussions to take place. As our research is

predicated on the assumption that we do not have access to all the experts, this solution will not work.

Another solution is to create "clusters" of experts that, while they may disagree, agree closely enough to be able to form a "coalition". With these coalitions, we can then use a majority rule voting system to make our final decision. This system would closely resemble a parliamentary system, where certain blocs would form coalitions on a case-by-case basis.

For example, say that there are 11 experts rating software on a scale from 1-5. 5 experts give the software a "1', 3 give it a "4", and 3 give it a "5". In a standard majority rule, we would conclude the software should be giving a ranking of "1". But if we group the two groups of 3 experts into a "4.5" voting bloc, then we can give the software a "4.5" rating.

## 2.3 Argumentation Frameworks

A major issue with previous approaches to MEMCDM was that they do not handle expert disagreements and conflicts well. They take all data into account without discerning or accounting for possible inconsistencies, and optimization handles them later (but rather in an average way than in a classification way) [5]. This approach is inefficient, and this inefficiency makes it difficult and time consuming to model cases with many criteria. For this reason, we decided to formalize the expert's arguments and conflicts as an argumentation framework (AF) [11].

An abstract argumentation framework is a pair $\langle A, R \rangle$ of a set of arguments $A$, and a binary attack relation $R \subseteq A \times A$, such that for $\forall a, b \in A$, $aRb$ means that $a$ attacks $b$. Argumentation frameworks can be represented as directed graphs, where the arguments are the nodes, and the attack relations are the edges. This is a useful formalization of argumentation, but we need a way to determine which arguments we want to accept as valid.

- Extensions:

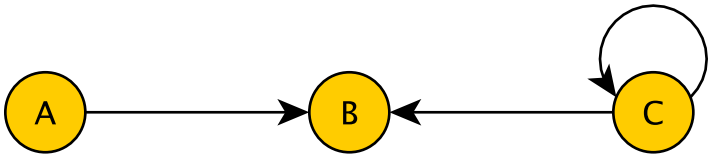


Figure 2.1: An example Argumenation Framework

The usefulness of AFs comes from knowing which arguments we want to accept, and which we want to reject. To do this, we use a concept called "extensions". An extension is a set of arguments that collectively fulfill some criteria of acceptability, which can vary depending on the extension. In order to define extensions, we must first define 3 separate semantics. First is the idea of a conflict-free set.

**Definition 4** *A set $B \subseteq A$ is* conflict free *iff for any two arguments $a, b \in B$, $aRb$ does not exist.*

Next, we must define the concept of defense.

**Definition 5** *An argument $b \in A$ is* defended *by a set $B \subseteq A$ iff $\forall c$ where $cRb, \exists d \in B$ such that $dRc$*

Finally, we must define admissibility.

**Definition 6** *A set $B \subseteq A$ is* admissible *iff $B$ is conflict free and $\forall b \in B$, $B$ defends $b$*

Now that we have properly defined admissibility, along with defense and conflict-free sets, we can now define extensions. Since our research currently focuses on stable extensions, we will define those.

**Definition 7** *A set $B \subseteq A$ is a* stable extension *of $A$ iff it is conflict-free, and $\forall c \notin B$, $\exists b \in B$ s.t. $bRc$ exists.*

Now that we have defined the stable extension, the question is how to find it. Fortunately, we are able to map this problem into a constraint satisfaction problem (CSP) and use constraint programming to find the solutions.

- Constraint-based solutions [2]:

  A CSP is a tuple $\langle V, D, C \rangle$, where $V$ is a set of variables, $D$ is a domain set such that each variable in $V$ has a value in $D$, and $C$ is a set of constraints. The objective is to find all possible value assignments for $V$ (within the domain $D$) as to satisfy the constraints $C$. We can map argumentation frameworks to CSPs in order to find conflict-free sets, admissible sets, and extensions. This is done by setting the variables to the arguments in the AF, and set the domain to {0,1} (either not taken, or taken, respectively). The constraints, however, will need to be adjusted based on what we want to find. In this example, let '$a$' be an argument, '$b$' be any argument that attacks '$a$', and '$b_1 \dots b_n$' be the set of all arguments that attack '$a$'. For the conflict-free constraints, we have:

$$\neg(a = 1 \wedge b = 1) \tag{2.1}$$

  For the stable constraints, we have:

$$\neg(a = 0 \wedge b_1 = 0 \wedge b_2 = 0 \wedge ... \wedge b_n = 0) \tag{2.2}$$

  To find the stable extensions, we use the constraints $C = \{conflict\text{-}free \cup stable\}$. With this, we can use a constraint solver in order to find the stable extension of any formally defined argumentation framework.

## 2.4 Optimization

In past works on MEMCDM, optimization was vital in fuzzy measure extraction and finding the best solution. Hence, it was critical to use optimization techniques that were hoped to guarantee a high degree of accuracy, while also offering large time savings. In the work of Xiaojing Wang [22], optimization was necessary in the extraction of fuzzy measures. In her work, she tested various optimization techniques, including genetic algorithms, gradient descent approach, and simulated annealing [14, 16]. Ultimately, the Bees algorithm was used [19].

In the Bees algorithm, several random points in the search space are selected. These spaces are then ranked by how well they maximize (or minimize) the objective function. The best points are selected, and from these points, local searches are conducted in order to find the optimal solution in the vicinity of those areas. This process then repeats with new random spaces to search. After a number of iterations, the best solution is chosen as the final answer of the Bees algorithm and an interval searching process is used to guarantee that this solution was correct as an optimum or to improve it otherwise.

The use of interval techniques to solve the optimization algorithm associated with identifying fuzzy measures made the whole process more reliable and unlikely to fall into local optima like previous approaches would [19].

However robust this approach was, it has two main flaws: (1) It is not very scalable: past 5 criteria ($2^5$ variables), the optimization algorithm has troubles terminating. (2) It ignores inconsistencies and does not allow to enforce properties of the decisions (such as fairness).

# Chapter 3

# The ConArg Solver for Argumentation Frameworks

When dealing with argumentation frameworks, we need a way to quickly determine what the extensions are. For this, we have the ConArg tool [7]. The ConArg tool has the ability to build an argumentation framework using a standardized syntax, as well as determine the extensions by using a constraint solver. There are two versions of the ConArg tool, one with a GUI, and one without[1]. In this section, the usage of the ConArg tool will be demonstrated using the GUI version.

## 3.1 The ASPARTIX Syntax

The first thing the ConArg tool does is that it takes an ASPARTIX [12] (Answer Set Programming Argumentation Reasoning Tool) file, and uses the information to create an argumentation framework. The ASPARTIX file has two basic statements: arguments and attacks. An argument $A$ is written as $arg(A)$., while the attack $ARB$ is written as $att(A, B)$.

This file is then put into the tool, at which point, the argumentation framework described by the ASPARTIX file is graphically represented.

[1] http://www.dmi.unipg.it/conarg/

```
arg(A).
arg(B).
arg(C).
att(A,B).
att(C,B).
att(C,C).
```

Figure 3.1: A sample ASPARTIX File

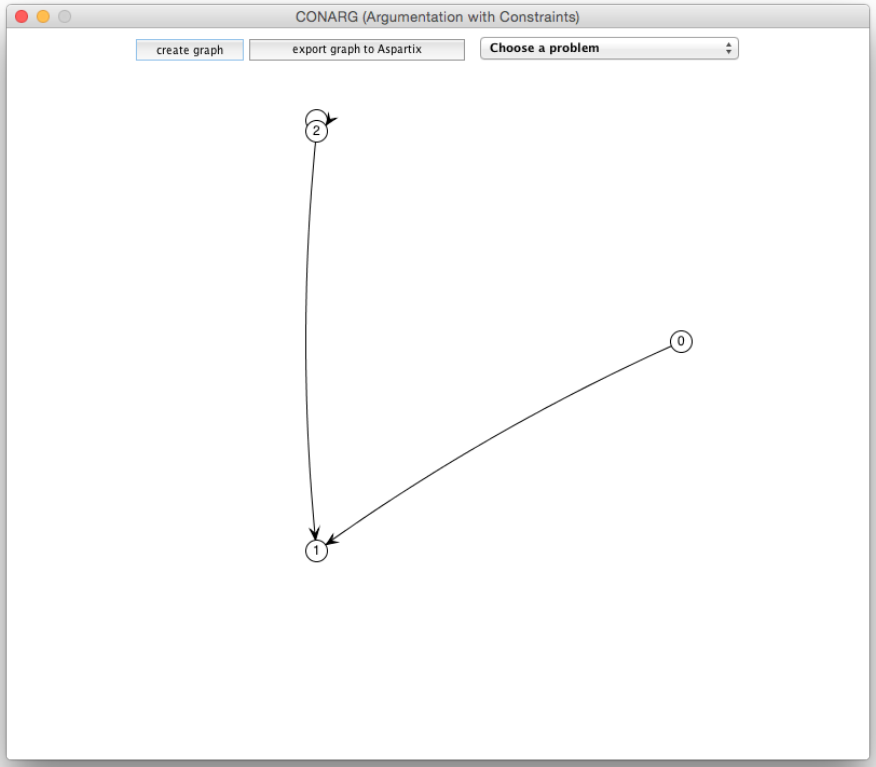


Figure 3.2: A sample AF in ConArg

## 3.2 Finding Extensions Using Constraint Solving

The other feature of the ConArg tool is its ability to determine the extensions of a given argumentation framework. The tool accomplishes this by translating the AF into a constraint

satisfaction problem (CSP), as described in chapter 2, and uses constraint programming to solve the problem.

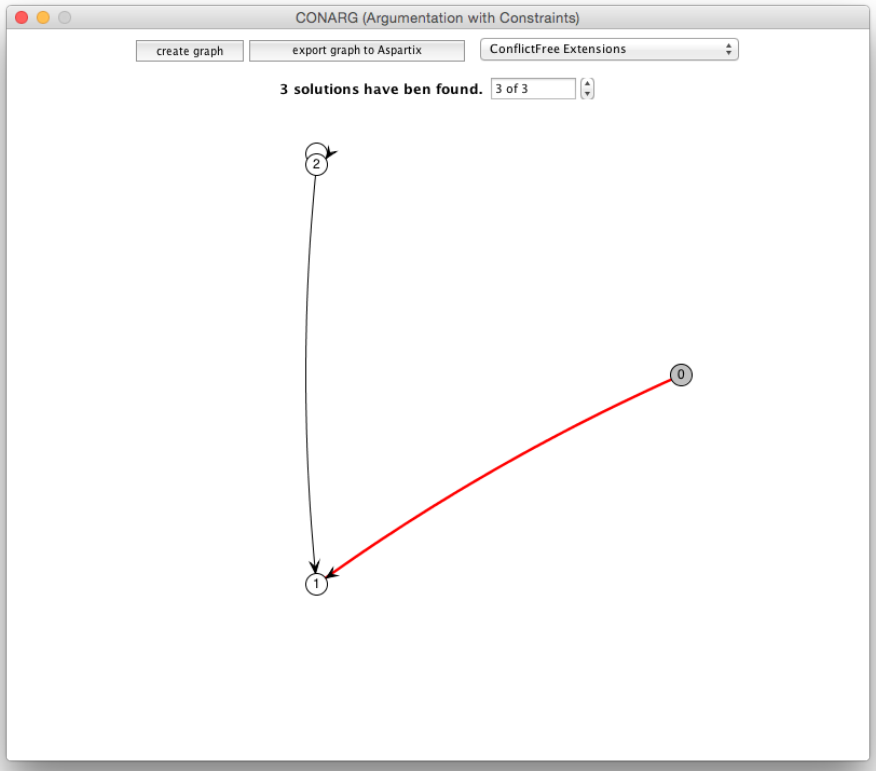


Figure 3.3: A sample conflict-free extension in ConArg

## 3.3 Soft Constraints

So far, we have only discussed argumentation with the assumption that all attacks are equally strong. But this is not always the case. Oftentimes, attacks will have varied strengths, and we may want to account for this strength difference. We can do this by attaching weights to the attack to create a weighted argumentation framework.

But our method on finding the extensions ignored the possibility of weights. We need

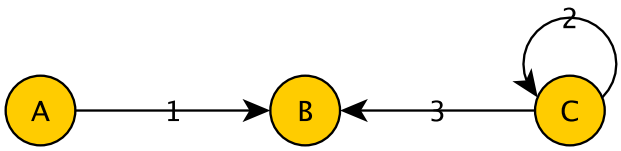


Figure 3.4: An example of a weighted AF

```
arg(A).
arg(B).
arg(C).
att(A,B):-1.
att(C,B):-3.
att(C,C):-2.
```

Figure 3.5: A sample ASPARTIX file with weights

a way to find these extensions while taking this weights into account. This is where $\alpha$-extensions and soft constraints come in.

Soft constraints, much like standard constraints, act as restrictions on what values certain variables can hold. However, unlike standard constraints, some degree of violation of the constraints is allowed. This level of violation is set by the user to their personal preference. In addition, there exist soft constraint satisfaction problems (SCSP), which, much like constraint satisfaction problems, have a domain of variables, a range of values that the variables can have, and a series of constraints.

A Soft Constraint Satisfaction Problem (SCSP) [4] is a tuple of $\langle V, D, C, k \rangle$, where $V$ is a set of variables, $D$ is the domain of values that each variable in $V$ can occupy, and $C$ is the set of soft constraints. For a given mapping of $D$ onto $V$, each constraint $C$ is mapped to a value $w$, which is determined by how much that constraint is violated by the variable mapping. If the constraint is fully satisfied, then $w$ is 0. Then, for constraints $c_1, \ldots, c_n$ and corresponding weights $w_1, \ldots, w_n$, the SCSP is satisfied as long as $w_1 \oplus \ldots \oplus w_n < k$,

where $\oplus$ is an aggregation function.

We can use soft constraint satisfaction problems in order to determine the $\alpha$-extensions, of weighted argumentation frameworks. $\alpha$-extensions, work in much the same way as standard extensions, except that these extensions can ignore attacks with a cumulative weight up to a pre-set $\alpha$. Much like how standard AF extensions can be reduced to constraint satisfaction problems, these $\alpha$-extensions, can be similarly reduced to SCSPs.

These weighted argumentation frameworks and soft constraints allow for minor disagreements, represented by weak attacks, to be ignored when searching for the set of arguments to consider in the final decision. This in turn will result in more feasible solutions. However, our current model does not use weighted attacks, and the usage of such is left to future work.

# Chapter 4

# How to blend AFs and MEMCDM

## 4.1 Problem Description

Now that we have defined Argumentation Frameworks, we need to define a model that will help us solve our problem. To accomplish this, we need a model that has the following properities:

1. All explicit data exists as arguments in the framework.
2. The framework is not disjoint, otherwise, we're solving separate smaller problems.
3. The model should represent disagreements and inconsistencies as attacks.
4. The extensions of the model should also give us a final outcome of the quality of the item in question.

In this next section, we define a model that meets all of these properties.

## 4.2 Modeling MEMCDMs as AFs

Here, we describe our model: given an MEMCDM problem with $n$ criteria and $p$ experts, how do we "translate"/model it as an AAF? In other words, which arguments and attacks should compose it? [5]

### 4.2.1 Arguments

**• What does the data we use (e.g., experts' evaluation of software in this case) tell us about the arguments to add to the network?**

We differentiate arguments that come from the data (i.e., Expert $i$ said that Software $j$ is good) from arguments that are implicit (i.e., Software $k$ is Poor).

1. **Expert $i$ gives Item $j$ a total quality $D_{ij}$** (which, in the case of Software Quality Assessment – SQA, can be Bad, Poor, Fair, Good, or Excellent):

   $$\textbf{Argument} \quad (E_i, S_j, D_{ij})$$

   Let us call such arguments, **arguments of type ESD**.

2. **Expert $i$ judges that Item $j$ satisfies criterion $m$ up to quality $D_{ijm}$**

   $$\textbf{Argument} \quad (E_i, S_j, c_m, D_{ijm})$$

   Let us call such arguments, **arguments of type EScD**.

**• Which implicit arguments should be part of the argumentation network for this specific type of problem?**

For each item, independently from what experts say, there will be a decision made. This decision will be in the form of a final ranking, ranging over all possibly ranking values (in the case of SQA: Bad, Poor, Fair, Good, Excellent). So regardless of ESD arguments, we add to the argumentation network the following arguments:

$$\forall \text{ item } S_i, \forall \text{ ranking } D_j : \ \textbf{Argument} \quad (S_i, D_j)$$

Let us call such arguments, **arguments of type SD**.

• **Coalitions of Arguments**

Here we aim to model the fact the $n$ decisions of any expert on the $n$ criteria of the problem at hand belong together: they together form the support for the expert's final decision on the given item. As a result, for any expert $E_i$ and any item $S_j$, we define a coalition of "supporting" decisions as:

$$\forall E_i, \forall S_j, \ \textbf{Coalition:} \ \{(E_i, S_j, c_k, D_{i,j,k}), \ k \in \{1, \ldots, n\}\}$$

Let us call such coalitions of EScDs, extended arguments of type **CoEScD**. The result of modeling such coalitions is that all arguments in the coalition will be forced to be either all in or all out of extensions. *Per se, we are enforcing an equality constraint on the belonging of these arguments to any extension.*

### 4.2.2 Attacks

In this subsection, we answer the following question: What are the **attacks (***edges of the network***)** between these arguments (*nodes*)? *Note:* All attacks we define are reciprocal, hence the edges are always set bidirectionally.

For attacks too, we differentiate between attacks that come from inconsistencies in the decision data (disagreement between experts, inconsistency in decisions of a single expert, lack of fairness, irrationality). An assumption that we make in designing the network model is that experts should be rational: in this, we mean that even if they are not (which we know), they should be and we aim to elicit decisions that are as rational as can be.

• **Attacks derived from lack of fairness**

Here, we assume that if an expert is fair, then s/he should derive the same final ranking from the same criteria rankings. For instance, if there are 3 criteria ($c_1$, $c_2$, and $c_3$) to assess

items and an expert E has the following decision history:

$$\begin{cases} E, S_i, c_1, D_1 \\ E, S_i, c_2, D_1 \\ E, S_i, c_3, D_1 \end{cases} \longrightarrow \quad E, S_i, D$$

and (with $S_i \neq S_j$):

$$\begin{cases} E, S_j, c_1, D_1 \\ E, S_j, c_2, D_1 \\ E, S_j, c_3, D_1 \end{cases} \longrightarrow \quad E, S_j, D'$$

where $D \neq D'$, then we should see arguments $(E, S_i, D)$ and $(E, S_j, D')$ are a lack of fairness in judgment and therefore add the following attack in the argumentation network: $(E, S_i, D) \longleftrightarrow (E, S_j, D')$.

More generally, assuming that the criteria that are considered by the experts are $c_k$, with $k \in K$, and that the possible rankings are denoted by $D_r$, with $r \in R$, then we add the following rule to our model:

$$\forall S_i,\ S_j,\ E \text{ s.t. } i \neq j,\ \forall k \in K, \exists r \in R,\ (S_i, E, c_k, D_r) \text{ and } (S_j, E, c_k, D_r):$$

$$\text{if } (S_i, E, D_i) \text{ and } (S_j, E, D_j) \text{ and } D_i \neq D_j$$
$$\text{then } \textbf{Attack } (S_i, E, D_i) \longleftrightarrow (S_j, E, D_j)$$

• **Attacks derived from lack of rationality**

Let us recall that we assume that the rankings $D_r$, with $r \in R$, are totally ordered. However, with $n$ criteria, the set of $n$-tuples of rankings is only partially ordered:

$$(D_1, D_2, \ldots, D_n) \prec (D'_1, D'_2, \ldots, D'_n)$$
$$\text{iff :}$$
$$\forall i \in \{1, \ldots, n\} :\ (D_i \neq D'_i) \longrightarrow D_i < D'_i$$

Now: $\forall E_i$ and $\forall S_j$, we denote by $(D_{1,i,j}, \ldots, D_{n,i,j})$ the set of $n$ decisions made by Expert $E_i$ on each of the criteria $c_1$, ..., $c_n$ for Item $S_j$, and by $D_{i,j}$ the final decision of Expert $E_i$ on Item $S_j$.

Being rational for any given expert $E_i$ means that if for Item $S_j$, s/he ranks criteria lower (w.r.t. above partial order) than s/he ranks the criteria of Item $S_k$, then his/her final ranking of $S_j$ should not be higher than his/her ranking of $S_k$. Formally, it is expressed as follows:

$$\begin{aligned}&\forall E_i,\ \forall S_j,\ \forall S_k (j \neq k):\\ &\text{if: } (D_{1,i,j}, \ldots, D_{n,i,j}) \prec (D_{1,i,k}, \ldots, D_{n,i,k}) \text{ and: } D_{i,j} > D_{i,k}\\ &\text{then: } \textbf{Attack } (S_j, E_i, D_{i,j}) \longleftrightarrow (S_k, E_i, D_{i,k})\end{aligned}$$

• **Attack related to implicit arguments: SD**

In this subsection, we describe the following attacks:

- attacks between implicit arguments SD; and
- attacks across SD and ESD.

1. Attacks among SDs: SD Arguments associate an item with a ranking. For each item $S_i$, there is $p$ SD arguments if there are $p$ possible ranking levels. Each of these $p$ arguments attack each other (they form a complete subgraph). In other words:

$$\forall S_i, \forall r_1, r_2 \in R, \text{ with } r_1 \neq r_2, \textbf{Attack: } (S_i, D_{r_1}) \longleftrightarrow (S_i, D_{r_2})$$

2. Attacks between SDs and ESDs: For any given item $S_i$, an argument saying that $S_i$ is evaluated $D_j$ is in contradiction (and therefore attacks – and vice-versa) any argument $(E, S_i, D_k)$ as soon as $D_j \neq D_k$. As a result:

$$\forall E,\ \forall S_i,\ (D_j \neq D_k) \ \rightarrow\ \textbf{Attack: } (S_i, D_j) \longleftrightarrow (E, S_i, D_k)$$

• **Attacks between Coalitions and ESDs**

Here we aim to model the fact that coalitions of decisions on criteria support experts' decisions. In order word:

$$\forall E_i, \forall S_j,\ \{(E_i, S_j, c_k, D_{i,j,k}),\ k \in \{1, \ldots, n\}\} \textbf{ supports } (E_i, S_j, D_{i,j})$$

In terms of attacks, this is expressed as follows:

$$\forall E_i, E_j \forall S_k :\ D_{i,k} \neq D_{j,k}\ \rightarrow$$
$$\textbf{Attack: } \{(E_i, S_k, c_l, D_{i,k,l}),\ k \in \{1, \ldots, n\}\} \longleftrightarrow (E_j, S_k, D_{j,k})$$

## 4.3 Reflection about AFs for Decision Prediction and Conflict

We believe that the model will allow us to resolve expert disagreements and come to a final decision on software based on the attacks between differing expert decisions, as well as expert decisions that contradict each other. However, another aspect of expert decision making that we wish to explore is predicting expert decisions. As stated earlier, expert prediction can allow us to automate expert decision-making and save a large amount of time and money. Fuzzy measure extraction can perform this functionality, provided the criteria assessment is known ahead of time, but there are several issues with it that we would like to address with our model.

We attempted to predict expert decisions by creating implicit criteria arguments, much like the implicit software quality arguments described earlier. These arguments took the form of $C_nR$ where $n$ corresponds to the $n$th criteria, and $R$ corresponds to the ranking. Much like the software decisions, these decisions would have implicit attacks in the form of $C_iR$ attacks $C_jR'$ (and vice versa), where $i = j$ and $R \neq R'$. Additionally, there are attacks between $C_iR$ and $S_jE_kC_iR'$ for all $i$, $j$, and $k$ where $R \neq R'$. We expected this set of arguments and attacks to aid in prediction of expert quality assessments when we have an indication of their quality per each criterion. However, we later removed these arguments

from our model, as we opted for a simpler, less complex model for our first model and to ensure that the ConArg tool ran more quickly.

In the end, our current model does not have a mechanism for predicting expert decisions. However, we wish to address this topic and modify our model to accommodate this feature in future work.

# Chapter 5

# Experimental Results

## 5.1 Motivation

It has been shown that abstract argumentation frameworks can be used to model expert decisions and disagreements [10]. However, we already have a method of evaluating multi-criteria decision making in the form of fuzzy measure extraction and the Choquet integral. So the question remains, what makes argumentation frameworks a better model for expert decision making than fuzzy measures. We believe that it has numerous advantages, including being able to distinguish between experts and software, allowing them to be evaluated individually. It also allows to weigh expert decisions differently, rather than listening to all decisions equally. However, these are only proposals. In this section, we wish to show whether or not this is the case in practice.

## 5.2 Comparison Method

To perform these tests, numerous toy-examples were created, with a varying number of software and criteria evaluated by a varying number of experts. Each toy case has either 3 or 5 rankings that can be given to a criteria or total quality. In the case of 3 evaluations, these rankings are poor, fair, and good. In the case of 5, these rankings are bad, poor, fair, good, and excellent.

For each toy example, an argumentation framework was created based on the data. These argumentation frameworks were put in the ConArg tool to search for the stable extensions. These extensions were then parsed for relevant results. For results to be

relevant, they must:

1. For every software to have at least one expert total quality assessment ($S_iE_jC$). In other words, the final quality assessment must be based on the input of at least one expert. This is based on the non-imposition quality of voting systems described in section 2.2.2.

2. For every software to have a final quality assessment ($S_iC$). In other words, the votes of the expert must yield a result. This is based off the universal domain quality of voting systems described in section 2.2.2.

However, even with this parsing, multiple possible solutions remain. To address the multiple solutions, the number of times each expert's total quality assessment and each final software assessment appear are tallied from the results. From these tallies, we can make a comparison between the number of times a final software assessment appears and votes on the final assessment. We can also make a comparison between the number of times an expert's total quality assessment appears and how influential the expert is. This will be explained in more detail in section 5.3.

For comparison, this data is also given to the hybrid algorithm for fuzzy measure extraction. However, this algorithm requires that data be in values between 0 and 1. To accommodate this, the discrete rankings are translated into discrete values. For the case of 3 rankings:

1. Poor is given a value of 0

2. Fair is given a value of 0.5

3. Good is given a value of 1

In the case of 5 rankings:

1. Bad is given a value of 0

2. Poor is given a value of 0.25

3. Fair is given a value of 0.5

4. Good is given a value of 0.75

5. Excellent is given a value of 1

From this data, we then extract the fuzzy measures using the hybrid algorithm. These fuzzy measures are then put into the Choquet integrals they were extracted from, in order to compare the result with the total quality assessment. However, there is an issue when it comes to determining the final quality assessment of a software in the case of Choquet. The algorithm to determine the final quality of a software in Choquet requires a large number of evaluations to be made. However, the ConArg tool does not currently run well with such data. Therefore, a true in-depth comparison cannot be done. To resole this, we will use a naive approach for Choquet, whereby the expert evaluations for the software according to Choquet are averaged.

## 5.3 The Tests

For this experiment, we ran a total of 21 tests, with various goals in mind.

The first test was ran with 2 criteria to test if the ConArg tool and the hybrid algorithm worked properly.

Tests 2-5 were run on a single set of raw data. However, for each test, the data was divided between experts and software in different ways. The idea was to run tests where the fuzzy measure data was the same, but the argumentation framework data was different.

Test 6 was run on a set of data designed around experts with various personality trends.

Test 7 was simply a set of randomized data.

Test 8 was run on a subset of real raw data, where a set of experts were analyzing various parts of the same software (different criteria assessment, but all same quality assessment)

Test 9 was run on a subset of the same raw data, but this time, it was across different software.

Tests 11-22 were randomly generated tests created the following way:

Tests 11-14 had 3 criteria, Tests 15-18 had 4 criteria, and Tests 19-22 had 5 criteria. Tests 11, 12, 15, 16, 19, & 20 had 3 rankings, while the rest of the tests had 5 rankings. The tests with 3 rankings had 5 experts evaluating 4 software, and the tests with 5 rankings had 4 experts. Tests 11, 15, 19, and 22 were untouched after generation. Tests 14 and 18 had some inconsistencies removed, while keeping some in and adding new ones manually. The rest of the tests had all inconsistencies removed.

**Note: All raw data for these tests can be found in the appendix.**

## 5.4 Results

We started with a small, but non-trivial test case in the form of Test 2. This test had 4 experts evaluating 3 software over 3 criteria, using a 3 ranking system. To test the various features of our model, we wanted a test that would contain various inconsistencies. For this, the following toy example was constructed.

In our model in the abstract argumentation framework, the data would be represented as such:

It should be noted that there are attacks between $S_1E_1F$ and $S_2E_1P$. This attack is a result of a lack of fairness on the part of $E_1$. She gives the same sequence of criteria assessments to $S_1$ and $S_2$, but gives them different total quality assessments. Likewise, there are attacks between $S_1E_2G$ and $S_3E_2F$, as well as between $S_2E_2G$ and $S_3E_2F$. This is a result of a lack of rationality on the part of $E_2$. Despite the fact that his criteria assessment of $S_3$ is strictly greater than that of $S_1$ and $S_2$, he give $S_3$ a lower total quality assessment than the other two.

Using the process described in the previous section, these rankings were converted into discrete values, as shown in the following table. Notice how the software and experts are

| Software | Expert | Total Quality | Criteria 1 | Criteria 2 | Criteria 3 |
|---|---|---|---|---|---|
| S1 | E1 | Fair | Poor | Good | Good |
| S1 | E2 | Good | Good | Poor | Poor |
| S1 | E3 | Fair | Fair | Fair | Poor |
| S1 | E4 | Poor | Fair | Poor | Poor |
| S2 | E1 | Poor | Poor | Good | Good |
| S2 | E2 | Good | Poor | Good | Fair |
| S2 | E3 | Fair | Poor | Good | Poor |
| S2 | E4 | Good | Good | Poor | Fair |
| S3 | E1 | Good | Good | Good | Fair |
| S3 | E2 | Fair | Good | Good | Good |
| S3 | E3 | Fair | Fair | Fair | Poor |
| S3 | E4 | Good | Good | Fair | Poor |

Table 5.1: Raw data for Test 2

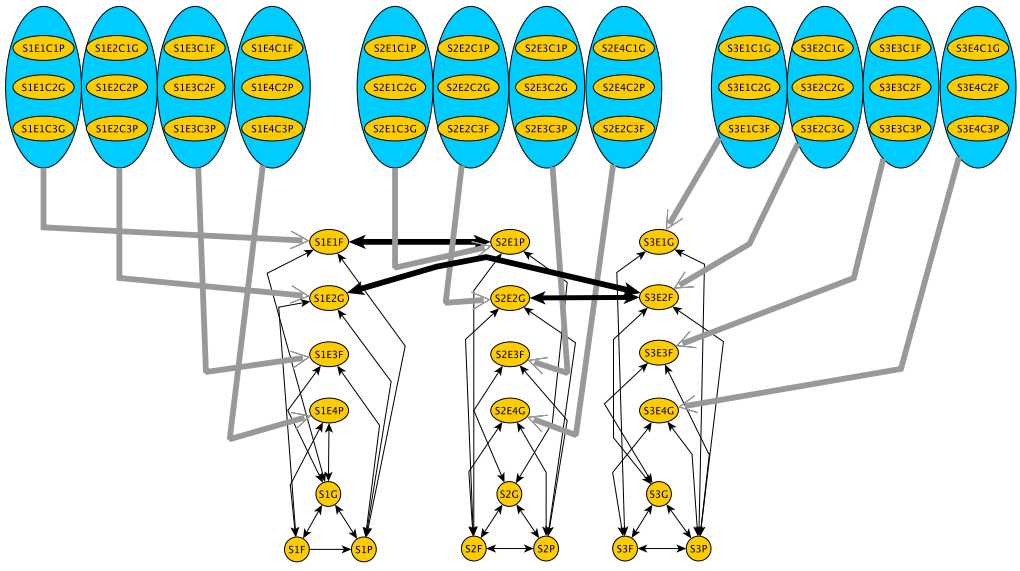


Figure 5.1: The AF Representing the data in Table 5.1 Data

not defined here.

Translating this data into the ASPARTIX format for our model, inputting it into the ConArg too, and parsing the data as described earlier, the following counts were tallied.

Here, we have tallied the number of times $S_iD$ appears, as well as how many times $S_iE_jD$ appears. The number of times the former appears is the voting results of the final decision of Software $i$. The latter represents the amount of influence Expert $j$ had on the voting of Software $i$. This value will further be referred to as that expert's "clout". What's important to note is that in $S_1$ and $S_2$, $E_1$ and $E_2$ have lower clout than $E_3$ and $E_4$. This is a result of $E_1$ and $E_2$'s inconsistencies noted earlier. Because they were inconsistent, their clout, and thus their influence in the voting, dropped. This is also reflected in the voting results. Despite the fact that just as many experts said that $S_1$ was Poor as did Good, Poor has more votes. This is because $E_4$, who voted Poor, was consistent, while $E_2$ was not. What is also important is that the vote tallies are not simply an addition of the clouts of each expert that made that decision.

For the fuzzy measures, we determined the values of the Choquet integrals corresponding to each expert decision using the process described in the previous section. To get the final software quality, we determined the mean of the Choquet integral values for each expert in a given software. However, this is a naive way to perform this task, and that doing so in a more accurate manner requires larger data sizes that we were able to process. For Software 1, we got an average value of 0.68, which indicates that the software should be judged as "Fair", but slightly leaning towards "Good". However, $E_1$, who voted "Fair", and $E_2$, who voted "Good", contradicted themselves, so their input should be taken less into consideration, and the result should be slightly **lower** than 0.5, rather than higher. Indeed, the AF analysis gives us just that. In software 2, we acquired an average of 0.5825.

In Software 3, we obtained an average value of 0.875. In $S_3$, we have 2 Good votes, and 2 Fair votes. However, $E_2$'s assessment contradict their assessments of both $S_1$ and $S_2$, so the votes should lean closer to "Good". And they do in both the Choquet and the AF analysis.

```
Test 2 Analysis

S1:     Poor – 84
        Fair – 114
        Good – 70

        E1 – 60
        E2 – 70
        E3 – 84
        E4 – 84

S2:     Poor – 68
        Fair – 80
        Good – 120

        E1 – 68
        E2 – 70
        E3 – 80
        E4 – 85

S3:     Poor – 0
        Fair – 127
        Good – 141

        E1 – 94
        E2 – 56
        E3 – 99
        E4 – 94
```

Figure 5.2: Tallied results for Test 2

In Tests 3, 4, and 5, we used the same raw data as in Test 2, except that the number of softwares and experts were altered. Test 3 has 2 softwares and 6 experts. Test 4 has 4 softwares and 3 experts. Test 5 has 6 softwares and 2 experts. Other than the labeling of softwares and experts, all the data is unchanged. Doing this would not alter the fuzzy measures, as that does not take experts and software into account, but would alter the ConArg results. The ConArg results are as follows:

In Tests 3 and 4, there are no inconsistencies in the experts' decisions. Therefore, all experts have equal clout, and all votes are weighted equally. This simplifies the problem into a simple counting of real-world votes.

For the Choquet in Test 3, we get a value of 0.675 in $S_1$, and 0.75 in $S_2$. The latter of these makes sense, as we have 3 votes for Fair and 3 for Good with no contradictions. However, the former is strange, as we have 2 votes for each of the 3 rankings, which should give us a value closer to 0.5. However, we do end up with this equivalence in the AF.

The Choquet results for Test 4 were quite strange. For example, despite $S_1$ and $S_4$ having 2 votes for Fair, and one vote for Good without contradictions, the Choquet value of $S_1$ was 0.74 and the value for $S_4$ was 0.833. Not only should these values probably have been the same, they should have been lower in light of the fact that Fair outnumbers Good 2-to-1. Other odd discrepancies include the fact that $S_3$ has a value of 0.667, lower than $S_1$ and $S_4$, despite it actually having two Good votes and one Fair vote. In $S_2$, there are two Poor votes, and one Good vote. Despite this, it has a value of 0.61, only slightly lower than $S_3$.

In Test 5, however, we have many more contradictions in the data. This leads to voting results being decided mainly on these contradictions, as there are only two experts in this case. In $S_5$, $E_2$'s inconsistencies are so bad, than her input isn't even considered. Expert 2's decision on Software 5 contradicts her decisions on Softwares 1, 3, 4, and 6. When one makes a decision that contradicts so many other decisions, it should probably be called into question.

In regards to the Choquet integrals, Test 5 has some results that make sense, and

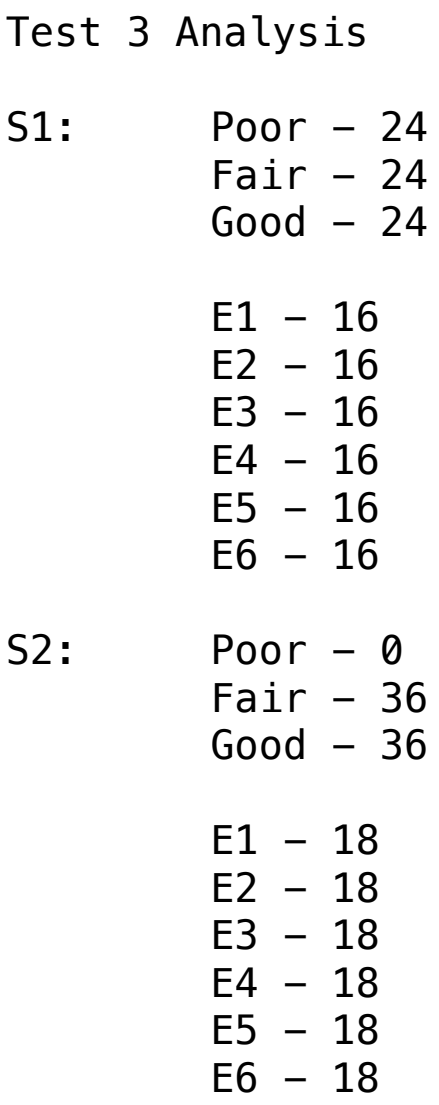


Figure 5.3: Tallied results for Test 3

```
Test 4 Analysis

S1:     Poor – 0
        Fair – 375
        Good – 250

        E1 – 250
        E2 – 250
        E3 – 250

S2:     Poor – 375
        Fair – 0
        Good – 250

        E1 – 250
        E2 – 250
        E3 – 250

S3:     Poor – 0
        Fair – 250
        Good – 375

        E1 – 250
        E2 – 250
        E3 – 250

S4:     Poor – 0
        Fair – 375
        Good – 250

        E1 – 250
        E2 – 250
        E3 – 250
```

Figure 5.4: Tallied results for Test 4

```
Test 5 Analysis

S1:     Poor – 0
        Fair – 512
        Good – 768

        E1 – 512
        E2 – 768

S2:     Poor – 640
        Fair – 640
        Good – 0

        E1 – 640
        E2 – 640

S3:     Poor – 256
        Fair – 0
        Good – 1024

        E1 – 256
        E2 – 1024

S4:     Poor – 0
        Fair – 512
        Good – 768

        E1 – 512
        E2 – 768

S5:     Poor – 0
        Fair – 0
        Good – 1280

        E1 – 1280
        E2 – 0

S6:     Poor –
        Fair – 640
        Good – 640

        E1 – 640
        E2 – 640
```

Figure 5.5: Tallied results for Test 5

some that don't. $S_1$, $S_3$, and $S_5$ involve conflicts that affect the weighting of one or both of the expert's votes, and the Choquet actually reflects this. In all 3 cases, the integral average actually leans towards the expert with the higher clout. Likewise, $S_6$ involves contradiction-free votes between Fair and Good, and has an average value of 0.75. However, $S_2$ has an average value of 0.5, despite it containing one Fair vote and one Poor vote, with no contradictions across software. In addition, $S_4$ also has an average of 0.5, despite it containing one Fair vote and one Good vote. But in this case, $E_1$, who voted Fair, has a lower clout than $E_2$, who voted Good. So the average is even lower than it probably should be.

In Test 6, we created a set of experts who each have different personality quirks that influence their voting. Their decisions, and the analysis are as follows:

From this, we can see that all experts, save for Expert 2, have equal clout in all decisions. However, Expert 2's decision making process is simply the opposite of Expert 1's. From this, we can reasonably assume that, at least in this toy example, the argumentation framework model doesn't determine the soundness of the decision-making process, so long as said process yields internally consistent results.

The Choquet values of Test 6 made sense for the most part. $S_1$ had a value of 0.289, which makes sense given that Good and Fair had one vote each while Poor had 3. Likewise, $S_4$ had a value of 0.368, which is reasonable given that there were two Poor votes, two Fair votes, and one contradicted Good vote. $S_2$ had one Good vote, one Poor vote, and 3 Fair votes, while $S_3$ had two Good, two Poor, and one Fair. Both of these had Choquet integral values of 0.4866 and 0.4317 respectively, only slightly lower than the expected Fair result of 0.5.

Test 7 was just a simple randomly-generated sample of data for 4 software and 4 experts.

Test 7 has some interesting conflicts that arise from the randomly-generated data. Expert 1 has conflicts between Softwares 1 and 2, as well as Softwares 2 and 4. Expert 2 has numerous conflicts, including between Softwares 1 & 2, 1 & 3, 2 & 4, and 3 & 4. Expert 3 just has a conflict between Softwares 1 & 2. Expert 4 has no conflicts. This, in turn leads

```
Test 6 Analysis:

S1:     Good - 992
        Fair - 992
        Poor - 1984

        E1 - 992
        E2 - 992
        E3 - 992
        E4 - 992
        E5 - 992

        Judgement: Poor

S2:     Good - 1024
        Fair - 1920
        Poor - 1024

        E1 - 1024
        E2 - 768
        E3 - 1024
        E4 - 1024
        E5 - 1024

        Judgement: Fair

S3:     Good - 1488
        Fair - 992
        Poor - 1488

        E1 - 992
        E2 - 992
        E3 - 992
        E4 - 992
        E5 - 992

        Judgement: Fair

S4:     Good - 896
        Fair - 1536
        Poor - 1536

        E1 - 1024
        E2 - 896
        E3 - 1024
        E4 - 1024
        E5 - 1024

        Judgement: Fair
```

Figure 5.6: Tallied results for Test 6

```
Test Analysis:

S1-S5:   Bad: 6561
         Poor: 19683
         Fair: 6561
         Good: 19683
         Excellent: 6561

         E1: 13122
         E2: 13122
         E3: 6561
         E4: 6561
         E5: 13122
         E6: 6561
         E7: 13122
```

Figure 5.7: Tallied results for Test 8

to the wildly varying clout scores seen in the results. Unlike the other tests, Test 7 did not return any results for the fuzzy measure extraction. This is also seen in other instances where test data is randomly generated and unaltered.

With Tests 8 and 9, we ran our algorithm on a subset of real raw data. In this raw data, sets of experts evaluate sections of various software separately over a set of criteria, then come to a final decision of that software as a whole. Test 8 covers various sections of a single software, which all have the same total quality assessment. Test 9, on the other hand, covers the evaluations of the first section of separate software, so the total quality assessments are different for each expert across software. It should also be noted that this is the first tests run where there are 5 rankings, rather than three.

As we can see, every "software" in Test 8 gives us the same results. This makes sense, since in the data, each total quality is the same, and the experts are just analyzing different sections of the software. We would hope that the results would be the same in this case. Likewise, the Choquet integral average also gave expected results, yielding a value of 0.480. Given the votes that are split evenly down the middle, it would make sense to have a value close to 0.5.

In Test 9, we get to see how the experts analyze multiple actual softwares. We can see that, for the most part, the experts are fairly consistent. However, Expert 6 has a

conflict between Softwares 2 and 3. While this conflict is not enough to swing the results, as the conflicts arise in cases where an overwhelming consensus already exists, it shows that conflicts can arise in real-world scenarios and that our model can account for these scenarios.

The Test 9 Choquet averages yield results that are reasonable. In all 3 software, the votes are split down the middle, with every Bad vote having an Excellent vote, and every Poor vote having a Good vote. Likewise, all 3 averages are close to 0.5 (0.4747, 0.469, 0.5 respectively). In the case of $S_2$, $E_6$ voted Good, but in such a way that it contradicts their vote in $S_3$. The average of Software 2 is correspondingly lower than either Softwares 1 or 3. However, since this was the only contradiction among the 7 experts, it didn't impact the results all that much. However, the results weren't lower in the stable extension tally, either.

For the set of randomized tests, all of the tests that were unaltered after creation (11, 15, 19, 22) as well as Test 12 did not return a solution for the Choquet integral, but they did return stable extensions, much like Test 7. We believe that this shows that the AF model is better at handling wildly inconsistent data than the fuzzy measure model.

Tests 13 and 14 were mostly consistent in both the AF tallies and the Choquet average. However, in Test 13, the Choquet average for $S_3$ and $S_4$ were both 2 votes for Fair and 2 for Poor. However, in Software 3, Expert 2, who voted Poor, was inconsistent. In spite of this, Software 3's Choquet average was lower than Software 4's (0.332 and 0.427).

Test 16, 17, and 18 also had fairly consistent results, as well. However, there were a few interesting quirks. In Test 16, for Software 4, we acquired a Choquet average of 0.611. This is strange, as this test had two votes for poor, two for fair, and one for good, with no contradictions. Test 17's results were consistent with what was found in the stable extension tally. Test 18 had a few interesting anomalies. Software 1's average was 0.589, despite everyone voting fair. Additionally, Software 4 had an average of 0.667, despite no one voting worse than Good, which would result in an expected minimum of 0.75. However, some of the experts of this software contradict themselves in Software 3. However, according

```
Test Analysis:

S1:     Bad: 79
        Poor: 237
        Fair: 79
        Good: 237
        Excellent: 79

        E1: 158
        E2: 158
        E3: 79
        E4: 79
        E5: 158
        E6: 79
        E7: 158

S2:     Bad: 0
        Poor: 90
        Fair: 540
        Good: 81
        Excellent: 0

        E1: 180
        E2: 180
        E3: 180
        E4: 90
        E5: 180
        E6: 81
        E7: 180

S3:     Bad: 0
        Poor: 216
        Fair: 279
        Good: 216
        Excellent: 0

        E1: 144
        E2: 144
        E3: 144
        E4: 144
        E5: 144
        E6: 126
        E7: 144
```

Figure 5.8: Tallied results for Test 9

to the stable extension tally, this should result in a slight favor towards Excellent, which is the opposite of Choquet's result.

Test 20's results are reasonable at first glance, but become somewhat peculiar on second glance. First, it should be noted that there are no cross-software contradictions in this test. Software 1 and SoftWare 2 both have two votes for Fair, two for Poor, and one for Good. However, Software 1 has a choquet average of 0.458 while Software 2 has an average of 0.335. Furthermore, Software 3 has an average of 0.437, lower than Software 1's, despite Software 3 having 3 Fair votes and only one Poor vote. Software 4 has the lowest value of 0.259, which makes sense, as it is the only software where no expert voted Good.

Like most of the tests before it, Test 21 is also fairly consistent. Some of the values are higher or lower than expected, but still within an acceptable range. For example, Software 1 had a Choquet average of 0.453, despite 3 votes for Poor and one for Good. One would expect the value to be closer to 0.25. Another is that Software 2's Choquet average value is 0.604, which is between the two votes for fair and the two for excellent, but would be expected to be closer to 0.75.

## 5.5 Summary of Findings

From this data, we can come to some interesting conclusions. The first one is consistency in delivering results. Every test that was run returned a set of stable extensions, but not all tests returned a set of fuzzy measures. However, the tests where the fuzzy measures were not returned were all randomly generated, and were likely caused due to highly contradictory data. It is highly unlikely that one would see this type of data in a practical setting. Another is the delivery of logical conclusions. With the stable extensions, we get tallies in proportion to how many experts voted on a particular software quality ranking. However, the Choquet average often gave strange results and sometimes gave very different results in similar settings. However, the Choquet average is a naive technique used due the lack of a more precise aggregation technique for smaller data sets. Finally, there's scalability.

Fuzzy measure extraction is known for having trouble handling even a moderate number of criteria, due to exponential growth of the number of values to extract. The ConArg tool, however, does not have this issue and can handle these criteria quite easily. However, the tool has a great deal of trouble with a moderate number of software and experts. Fuzzy measure extraction, on the other hand, can handle large data sizes for this extraction.

From the data we have tested, we have shown that argumentation frameworks have several advantages over fuzzy measures. However, these advantages have only been shown to appear in small data sizes, rather than larger, real-world sample data. Nevertheless, we believe that these conclusions show that argumentation frameworks are an avenue worth pursuing in regards to multi-expert multi-criteria decision-making.

## 5.6 Future Analysis

All of the tests described in this section are relatively small compared to previous works in this area. We attempted to run our model on larger, real world examples, such as the entirety of the data that tests 8 and 9 were based on. However, these tests proved to take an extraordinary amount of time to run. Even smaller tests proved daunting. Tests 11-22 were supposed to have 5 experts and 5 software. However, these took too long to run and had to be shrunk in order to acquire results. We attempted to solve this issue by creating a model where the criteria coalitions described in Chapter 4 were projected onto the total quality assessment arguments. We did this by removing the coalitions, and letting the attacks on the implicit software quality arguments act as the way to prevent a stable extension from taking two expert total quality assessment that disagree, thus shrinking our model. This made the tests run faster, but we acquired different results. In the future, we would like to find a way to address this issue and run larger tests and compare them to fuzzy measure extraction.

# Chapter 6

# Conclusion and Future Work

We have managed to show that modeling expert decisions using argumentation frameworks, rather than fuzzy measures, provides numerous advantages. Primary among them is the ability to detect experts that make inconsistent decisions, and lower their influence in the final decision making process as a result. However, there are still some features that are important to implement in future iterations of the model.

The first is prediction. The fuzzy measure model can help predict future decisions an expert will make by using extracted fuzzy measures in new Choquet integrals to determine outcomes. Our current model lacks this predictive ability. In the future, it would be critical to extend our model to determine future decisions made by experts. This would lead further into the automation of expert decisions, which would offer large time and cost savings. Another issue is speed. While the fuzzy measure extraction has issues with scalability when a large number of criteria are taken into account, finding extensions in an argumentation framework takes a considerable amount of time, even if there are only a handful of experts and software. We attempted to solve this issue by projecting individual criteria arguments and attacks onto the total quality arguments. Unfortunately, this attempt gave us different results. One idea is to try to tweak the ConArg tool to be better optimized for the kind of argumentation frameworks we are dealing with in our problem.

Our model has a distinct advantage over the fuzzy measure model in that it takes into account inconsistent expert decisions across items, and ignores expert decisions that don't match their criteria judgement; whereas the fuzzy measure model ignores inconsistencies across items, and breaks upon encountering mismatches between a final decision and criteria assessment. However, we would like our model to take inconsistencies within the same item

into account (aside from where they contradict other decisions). However, this is not as simple as punishing the expert for this inconsistency, as the issue may not be with the expert, but the evaluation criteria. We want to be able to detect if this contradiction happens at a significant rate, and if so, suggest expanding the criteria in which the items are evaluated under.

In conclusion, while this model is still in its early stages, its advantages are already apparent, and shows that argumentation frameworks are a promising direction in the modeling and evaluation of multi-expert multi-criteria decision making. However, much work remains to be done before it could potentially replace fuzzy-measure extraction.

# References


[1] Threat Assessment: Predicting and Preventing School Violence. http://www.nasponline.org/resources/factsheets/threatassess_fs.aspx.

[2] L. Amgoud, C. Devred. 'Argumentation Frameworks as Constraint Satisfaction Problems'. *Proceedings of the 5th International Conference on Scalable Uncertainty Management (SUM 2011)*, Vol. 6929, 110-122, 2011.

[3] K. Arrow. 'A Difficulty in the Concept of Social Welfare'. *Journal of Political Economy*, 58(4), 328-346, 1950.

[4] S. Bistarelli. 'Semirings for Soft Contraint Solving and Programming', *Lecture Notes in Computer Science*, Vol. 2962, Springer, 2004.

[5] S. Bistarelli, M. Ceberio, J. Henderson, and F. Santini. 'Abstract Argumentation Frameworks to Promote Fairness and Rationality in Multi-Experts Multi-Criteria Decision Making'. *Italian Conference on Theoretical Computer Science (ICTCS)*, 247-257, 2015.

[6] S. Bistarelli, F. Santini. 'A Common Computational Framework for Semiring-based Argumentation Systems'. *Euro- pean Conference on Artificial Intelligence (ECAI10)*, 2010.

[7] S. Bistarelli, F. Santini. 'ConArg: a Tool to Solve (Weighted) Abstract Argumentation Frameworks with (Soft) Constraints'. arXiv:1212.2857v2. 2013.

[8] M. Ceberio, F. Modave. 'An Interval-valued, 2-additive Choquet Integral for Multi-criteria Decision Making'. *Proceedings of the 10th Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'04)*, Perugia, Italy. July 2004.

[9] E.F. Combarro, P. Miranda. 'Identification of Fuzzy Measures from Sample Data with Genetic Algorithms', *Computers & Operations Research*, 33(10), 3046-3066, 2006.

[10] S. Coste-Marquis, C. Devred, S. Konieczny, M. Lagasquie-Schiex, P. Marquis. 'On the Merging of Dung's Argumentation Systems'. *Artificial Intelligence*, 171(10-15), 730-753, 2007.

[11] P.M. Dung. 'On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-person Games'. *Artificial Intelligence*, 77(2), 221-357, 1995.

[12] U. Egly, S.A. Gaggl, S. Woltran. 'ASPARTIX: Implementing Argumentation Frameworks Using Answer-Set Programming'. *International Conference on Logic Programming (ICLP)*, 734-738, 2008.

[13] P.C. Fishburn. 'Additive Utilities with Incomplete Product Set: Applications to Priorities and Assignments'. *Operations Research Society of America (ORSA)*, 15(3), 537-542, 1967.

[14] M. Grabisch. 'A New Algorithm for Identifying Fuzzy Measures and its Applications to Pattern Recognition'. *Proceedings of 4th IEEE International Conference on Fuzzy Systems*, Vol. 1, 145-150, Yokohama, Japan. March 1995.

[15] M. Grabisch. 'The Application of Fuzzy Integrals in Multicriteria Decision Making'. *European Journal of Operational Research*. 89(3), 445-456, 1996.

[16] S. Kirkpartick, C.D. Gelatt, M.P. Vecchi. 'Optimization by Simulated Annealing'. *Science*, 220(4598), 671-680, 1983.

[17] F. Modave, P.W. Eklund. 'A Measurement Theory Perspective for MCDM'. *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, 1068-1071, 2001.

[18] P. Norris. 'Choosing Electoral Systems: Proportional, Majoritarian, and Mixed Systems'. *International Political Science Review*, 18(3), 297-312, (1997).

[19] D. Pham, A. Ghanbarzadeha, E. Koc, S. Otri, S. Rahim, and M. Zaidi. 'The Bees Algorithm-A Novel Tool for Complex Optimization Problems'. *Proceedings of 2nd International Virtual Conference on Intelligent Production Machines and Systems*, 454-459, 2006.

[20] B. Roy. 'Problems and Methods with Multiple Objective Functions'. *Mathematical Programming*, 1, 1971.

[21] T.L. Saaty. *The Analytical Hierarchical Process: Planning, Priority Setting, Resource Allocation*. McGraw-Hill, New York. 1980.

[22] X. Wang. *Extracting Fuzzy Measures from Sample Data: Optimization Algorithms and Applications*. PhD Dissertation. University of Texas at El Paso, El Paso, TX. 2012.

[23] R. Yager. 'Non-numeric Multi-criteria Multi-person Decision-making'. *Group Decision and Negotiation*, Vol. 2, 81-93, 1993.

[24] K. Yoon, C. Hwang. *Multiple Attribute Decision Making: an Introduction*. Sage Publications, Thousand Oaks, Ca. 1995.

[25] R. Zeckhauser. 'Voting Systems, Honest Preferences and Pareto Optimality'. *American Political Science Review*, 67(3), 934-946, 1973.

# Appendix A

# Raw Data

| Software | Expert | Total Quality | Criteria 1 | Criteria 2 | Criteria 3 |
|---|---|---|---|---|---|
| S1 | E1 | F | P | G | G |
| S1 | E2 | G | G | P | P |
| S1 | E3 | F | F | F | P |
| S1 | E4 | P | F | P | P |
| S2 | E1 | P | P | G | G |
| S2 | E2 | G | P | G | F |
| S2 | E3 | F | P | G | P |
| S2 | E4 | G | G | P | F |
| S3 | E1 | G | G | G | F |
| S3 | E2 | F | G | G | G |
| S3 | E3 | F | F | F | P |
| S3 | E4 | G | G | F | P |

Table A.1: Raw data for Test 2

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|---|---|---|
| S1 | E1 | P | P | F | F | P | P | P |
| S1 | E2 | G | G | F | G | G | F | G |
| S1 | E3 | P | G | F | P | P | G | P |
| S1 | E4 | F | F | G | G | P | F | F |
| S1 | E5 | P | P | F | F | G | P | G |
| S2 | E1 | P | F | F | P | F | F | P |
| S2 | E2 | F | F | G | G | F | G | F |
| S2 | E3 | F | P | P | F | F | P | P |
| S2 | E4 | G | G | P | F | F | G | G |
| S2 | E5 | F | F | G | P | F | F | F |
| S3 | E1 | F | P | P | F | P | P | F |
| S3 | E2 | G | G | F | G | G | G | G |
| S3 | E3 | P | F | P | P | G | G | P |
| S3 | E4 | G | G | P | G | F | G | F |
| S3 | E5 | P | P | F | F | G | P | G |
| S4 | E1 | P | F | F | P | P | F | P |
| S4 | E2 | G | F | F | G | F | F | F |
| S4 | E3 | F | P | G | F | P | G | F |
| S4 | E4 | P | P | F | P | G | P | G |
| S4 | E5 | F | F | F | P | F | F | F |

Table A.2: Raw data for Test 6

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 |
|---|---|---|---|---|---|---|
| S1 | E1 | P | P | F | G | F |
| S1 | E2 | G | F | P | F | F |
| S1 | E3 | F | P | G | P | P |
| S1 | E4 | G | F | P | G | G |
| S2 | E1 | F | P | P | F | F |
| S2 | E2 | F | G | G | G | G |
| S2 | E3 | G | P | G | P | P |
| S2 | E4 | G | F | P | P | P |
| S3 | E1 | F | F | G | P | F |
| S3 | E2 | P | F | G | F | F |
| S3 | E3 | P | F | P | G | G |
| S3 | E4 | G | G | P | F | G |
| S4 | E1 | P | G | P | G | G |
| S4 | E2 | G | P | P | P | P |
| S4 | E3 | F | G | F | P | F |
| S4 | E4 | G | F | P | G | G |

Table A.3: Raw data for Test 7

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|---|---|---|
| S1 | E1 | P | F | F | B | P | P | F |
| S1 | E2 | G | G | G | G | P | F | F |
| S1 | E3 | E | E | G | G | G | E | G |
| S1 | E4 | F | F | F | P | F | F | F |
| S1 | E5 | G | G | G | F | G | G | F |
| S1 | E6 | B | B | B | B | P | P | P |
| S1 | E7 | P | F | P | B | P | F | F |
| S2 | E1 | P | F | F | B | P | P | F |
| S2 | E2 | G | G | G | G | P | F | F |
| S2 | E3 | E | E | G | G | G | E | G |
| S2 | E4 | F | F | F | P | F | F | F |
| S2 | E5 | G | G | G | F | G | G | F |
| S2 | E6 | B | B | B | B | P | P | P |
| S2 | E7 | P | F | P | B | P | F | F |
| S3 | E1 | P | F | F | B | P | P | F |
| S3 | E2 | G | G | G | G | P | F | F |
| S3 | E3 | E | E | G | G | G | E | G |
| S3 | E4 | F | F | F | P | F | F | F |
| S3 | E5 | G | G | G | F | G | G | F |
| S3 | E6 | B | B | B | B | P | P | P |
| S3 | E7 | P | F | P | B | P | F | F |
| S4 | E1 | P | F | F | B | P | P | F |
| S4 | E2 | G | G | G | G | P | F | F |
| S4 | E3 | E | E | G | G | G | E | G |
| S4 | E4 | F | F | F | P | F | F | F |
| S4 | E5 | G | G | G | F | G | G | F |
| S4 | E6 | B | B | B | B | P | P | P |
| S4 | E7 | P | F | P | B | P | F | F |
| S5 | E1 | P | F | F | B | P | P | F |
| S5 | E2 | G | G | G | G | P | F | F |
| S5 | E3 | E | E | G | G | G | E | G |
| S5 | E4 | F | F | F | P | F | F | F |
| S5 | E5 | G | G | G | F | G | G | F |
| S5 | E6 | B | B | B | B | P | P | P |
| S5 | E7 | P | F | P | B | P | F | F |

Table A.4: Raw data for Test 8

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|---|---|---|
| S1 | E1 | P | F | F | B | P | P | F |
| S1 | E2 | G | G | G | G | P | F | F |
| S1 | E3 | E | E | G | G | G | E | G |
| S1 | E4 | F | F | F | P | F | F | F |
| S1 | E5 | G | G | G | F | G | G | F |
| S1 | E6 | B | B | B | B | P | P | P |
| S1 | E7 | P | F | P | B | P | F | F |
| S2 | E1 | F | F | P | P | F | P | G |
| S2 | E2 | F | G | F | P | G | F | G |
| S2 | E3 | F | G | G | B | P | P | G |
| S2 | E4 | P | P | P | P | P | P | F |
| S2 | E5 | F | E | G | F | G | B | E |
| S2 | E6 | G | F | P | G | P | P | F |
| S2 | E7 | F | G | G | F | P | P | F |
| S3 | E1 | G | G | G | F | G | E | G |
| S3 | E2 | P | G | F | B | P | P | G |
| S3 | E3 | F | G | G | B | F | P | G |
| S3 | E4 | P | F | P | B | P | P | F |
| S3 | E5 | F | G | F | G | G | B | E |
| S3 | E6 | F | F | F | G | P | F | G |
| S3 | E7 | G | G | F | F | F | F | F |

Table A.5: Raw data for Test 9

| Software | Expert | Total Quality | C1 | C2 | C3 |
|---|---|---|---|---|---|
| S1 | E1 | F | G | P | G |
| S1 | E2 | P | G | F | F |
| S1 | E3 | F | F | G | F |
| S1 | E4 | F | F | F | P |
| S1 | E5 | G | F | G | G |
| S2 | E1 | G | G | G | P |
| S2 | E2 | P | P | G | G |
| S2 | E3 | P | P | G | G |
| S2 | E4 | G | F | F | F |
| S2 | E5 | P | F | P | F |
| S3 | E1 | P | G | G | G |
| S3 | E2 | F | F | G | G |
| S3 | E3 | P | G | F | G |
| S3 | E4 | P | F | G | G |
| S3 | E5 | F | F | P | G |
| S4 | E1 | G | F | F | G |
| S4 | E2 | P | G | P | F |
| S4 | E3 | F | F | G | P |
| S4 | E4 | G | F | P | F |
| S4 | E5 | P | P | F | F |

Table A.6: Raw data for Test 11

| Software | Expert | Total Quality | C1 | C2 | C3 |
|---|---|---|---|---|---|
| S1 | E1 | G | F | G | G |
| S1 | E2 | G | G | P | G |
| S1 | E3 | P | F | P | P |
| S1 | E4 | P | P | F | P |
| S1 | E5 | F | F | P | P |
| S2 | E1 | P | G | G | P |
| S2 | E2 | F | G | F | P |
| S2 | E3 | G | P | G | P |
| S2 | E4 | P | F | P | P |
| S2 | E5 | P | P | P | F |
| S3 | E1 | F | F | G | G |
| S3 | E2 | P | G | G | P |
| S3 | E3 | G | G | G | G |
| S3 | E4 | P | P | P | P |
| S3 | E5 | P | P | F | F |
| S4 | E1 | P | G | P | P |
| S4 | E2 | G | F | G | F |
| S4 | E3 | F | G | P | P |
| S4 | E4 | F | P | G | P |
| S4 | E5 | F | P | F | F |

Table A.7: Raw data for Test 12

| Software | Expert | Total Quality | C1 | C2 | C3 |
|---|---|---|---|---|---|
| S1 | E1 | P | G | B | F |
| S1 | E2 | P | E | E | B |
| S1 | E3 | F | B | P | G |
| S1 | E4 | E | E | E | P |
| S2 | E1 | G | G | F | E |
| S2 | E2 | B | B | B | G |
| S2 | E3 | P | P | P | P |
| S2 | E4 | E | E | E | E |
| S3 | E1 | P | B | B | P |
| S3 | E2 | P | B | B | F |
| S3 | E3 | F | B | G | E |
| S3 | E4 | F | G | E | F |
| S4 | E1 | P | P | P | P |
| S4 | E2 | F | F | F | F |
| S4 | E3 | F | P | E | E |
| S4 | E4 | P | P | F | F |

Table A.8: Raw data for Test 13

| Software | Expert | Total Quality | C1 | C2 | C3 |
|---|---|---|---|---|---|
| S1 | E1 | G | P | G | G |
| S1 | E2 | P | G | P | G |
| S1 | E3 | G | G | E | E |
| S1 | E4 | G | E | P | P |
| S2 | E1 | P | G | P | E |
| S2 | E2 | G | B | G | G |
| S2 | E3 | P | B | F | B |
| S2 | E4 | F | P | P | G |
| S3 | E1 | F | G | B | G |
| S3 | E2 | F | F | F | F |
| S3 | E3 | F | E | P | B |
| S3 | E4 | G | E | G | E |
| S4 | E1 | G | G | G | P |
| S4 | E2 | P | G | B | B |
| S4 | E3 | E | B | E | B |
| S4 | E4 | E | E | G | E |

Table A.9: Raw data for Test 14

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 |
|---|---|---|---|---|---|---|
| S1 | E1 | P | F | F | P | G |
| S1 | E2 | F | P | P | G | F |
| S1 | E3 | F | P | P | G | P |
| S1 | E4 | P | G | F | G | P |
| S1 | E5 | P | P | F | F | G |
| S2 | E1 | P | P | P | G | F |
| S2 | E2 | F | P | G | G | G |
| S2 | E3 | G | F | F | G | G |
| S2 | E4 | P | G | F | P | F |
| S2 | E5 | G | G | P | G | F |
| S3 | E1 | G | P | F | F | P |
| S3 | E2 | F | P | F | G | P |
| S3 | E3 | F | G | P | G | P |
| S3 | E4 | G | P | G | G | F |
| S3 | E5 | P | G | F | G | F |
| S4 | E1 | F | F | G | F | P |
| S4 | E2 | P | F | P | P | P |
| S4 | E3 | P | P | P | F | F |
| S4 | E4 | G | P | G | F | G |
| S4 | E5 | P | G | F | P | G |

Table A.10: Raw data for Test 15

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 |
|---|---|---|---|---|---|---|
| S1 | E1 | F | P | F | G | G |
| S1 | E2 | G | P | P | G | G |
| S1 | E3 | P | P | F | P | P |
| S1 | E4 | P | P | G | P | G |
| S1 | E5 | F | G | F | F | P |
| S2 | E1 | F | G | F | P | F |
| S2 | E2 | F | F | G | F | G |
| S2 | E3 | F | P | P | F | F |
| S2 | E4 | F | G | F | F | F |
| S2 | E5 | G | G | P | P | F |
| S3 | E1 | P | G | F | P | P |
| S3 | E2 | F | G | P | P | F |
| S3 | E3 | P | P | F | P | F |
| S3 | E4 | P | P | P | P | F |
| S3 | E5 | F | G | G | G | P |
| S4 | E1 | P | P | F | P | F |
| S4 | E2 | P | P | F | F | F |
| S4 | E3 | G | G | F | G | G |
| S4 | E4 | F | F | F | F | F |
| S4 | E5 | F | F | F | G | G |

Table A.11: Raw data for Test 16

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 |
|---|---|---|---|---|---|---|
| S1 | E1 | G | P | F | E | E |
| S1 | E2 | E | G | E | E | F |
| S1 | E3 | G | E | E | G | G |
| S1 | E4 | F | P | B | B | E |
| S2 | E1 | F | B | F | G | B |
| S2 | E2 | G | G | G | E | E |
| S2 | E3 | P | F | G | B | P |
| S2 | E4 | F | G | B | E | G |
| S3 | E1 | P | P | B | B | P |
| S3 | E2 | P | E | B | P | B |
| S3 | E3 | F | F | P | B | F |
| S3 | E4 | P | B | G | G | B |
| S4 | E1 | G | G | P | E | B |
| S4 | E2 | P | F | B | G | E |
| S4 | E3 | F | B | P | B | G |
| S4 | E4 | E | E | B | G | B |

Table A.12: Raw data for Test 17

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 |
|---|---|---|---|---|---|---|
| S1 | E1 | F | E | F | P | P |
| S1 | E2 | F | B | E | B | F |
| S1 | E3 | F | F | F | G | E |
| S1 | E4 | F | B | G | F | B |
| S2 | E1 | G | E | G | F | P |
| S2 | E2 | G | F | E | F | P |
| S2 | E3 | B | P | P | B | G |
| S2 | E4 | F | B | B | E | P |
| S3 | E1 | P | G | B | G | G |
| S3 | E2 | F | E | B | E | F |
| S3 | E3 | G | E | F | G | P |
| S3 | E4 | G | E | E | F | E |
| S4 | E1 | G | G | B | G | G |
| S4 | E2 | G | G | B | E | B |
| S4 | E3 | E | F | E | P | B |
| S4 | E4 | E | B | E | B | E |

Table A.13: Raw data for Test 18

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 | C5 |
|---|---|---|---|---|---|---|---|
| S1 | E1 | G | G | G | G | F | P |
| S1 | E2 | G | F | F | F | P | P |
| S1 | E3 | F | G | G | F | P | F |
| S1 | E4 | P | G | F | F | G | F |
| S1 | E5 | G | G | P | F | G | G |
| S2 | E1 | P | P | G | F | G | P |
| S2 | E2 | G | G | F | F | G | P |
| S2 | E3 | G | P | F | P | F | P |
| S2 | E4 | F | G | G | F | F | F |
| S2 | E5 | P | F | G | P | F | G |
| S3 | E1 | F | F | F | F | P | F |
| S3 | E2 | P | P | G | P | P | F |
| S3 | E3 | P | P | F | G | P | G |
| S3 | E4 | G | F | P | P | F | P |
| S3 | E5 | P | G | P | G | G | P |
| S4 | E1 | P | G | P | P | P | F |
| S4 | E2 | F | G | P | P | F | P |
| S4 | E3 | F | F | F | P | G | G |
| S4 | E4 | F | F | P | F | G | G |
| S4 | E5 | G | G | F | G | F | P |

Table A.14: Raw data for Test 19

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 | C5 |
|---|---|---|---|---|---|---|---|
| S1 | E1 | P | G | F | P | F | G |
| S1 | E2 | P | G | P | F | F | P |
| S1 | E3 | G | F | F | P | G | F |
| S1 | E4 | F | P | P | G | F | P |
| S1 | E5 | F | P | F | F | F | P |
| S2 | E1 | P | P | P | P | F | F |
| S2 | E2 | F | P | G | G | P | F |
| S2 | E3 | G | G | F | F | F | P |
| S2 | E4 | P | F | G | F | P | F |
| S2 | E5 | F | P | G | F | F | G |
| S3 | E1 | G | G | F | G | G | G |
| S3 | E2 | F | F | P | P | P | F |
| S3 | E3 | F | G | F | P | G | P |
| S3 | E4 | P | F | F | P | P | P |
| S3 | E5 | F | F | G | F | F | P |
| S4 | E1 | P | P | F | P | P | F |
| S4 | E2 | F | P | G | P | F | G |
| S4 | E3 | P | P | G | G | P | F |
| S4 | E4 | F | G | F | P | P | G |
| S4 | E5 | F | P | G | P | P | P |

Table A.15: Raw data for Test 20

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 | C5 |
|---|---|---|---|---|---|---|---|
| S1 | E1 | P | F | G | P | B | E |
| S1 | E2 | P | F | B | G | B | P |
| S1 | E3 | G | B | F | P | G | G |
| S1 | E4 | P | B | E | F | B | F |
| S2 | E1 | F | B | P | P | E | F |
| S2 | E2 | F | G | B | G | E | P |
| S2 | E3 | E | P | G | E | B | F |
| S2 | E4 | E | B | E | F | B | F |
| S3 | E1 | G | P | P | E | F | G |
| S3 | E2 | F | B | G | B | P | P |
| S3 | E3 | F | G | B | B | B | E |
| S3 | E4 | B | B | P | G | B | P |
| S4 | E1 | P | E | E | E | P | P |
| S4 | E2 | E | B | B | F | E | G |
| S4 | E3 | G | E | G | P | E | E |
| S4 | E4 | P | E | F | B | F | P |

Table A.16: Raw data for Test 21

| Software | Expert | Total Quality | C1 | C2 | C3 | C4 | C5 |
|---|---|---|---|---|---|---|---|
| S1 | E1 | G | P | P | F | F | F |
| S1 | E2 | E | E | E | B | E | P |
| S1 | E3 | E | P | E | G | P | E |
| S1 | E4 | G | B | P | F | F | F |
| S2 | E1 | G | G | E | F | P | G |
| S2 | E2 | P | G | E | P | P | B |
| S2 | E3 | E | F | G | G | B | G |
| S2 | E4 | G | P | E | P | P | F |
| S3 | E1 | P | B | G | B | P | E |
| S3 | E2 | B | E | F | P | P | G |
| S3 | E3 | E | P | G | P | G | F |
| S3 | E4 | P | F | P | F | B | G |
| S4 | E1 | B | P | E | G | B | E |
| S4 | E2 | B | F | G | G | E | F |
| S4 | E3 | F | E | G | B | G | G |
| S4 | E4 | B | G | G | P | P | B |

Table A.17: Raw data for Test 22

# Curriculum Vitae

Joel Henderson was born in Houston, Texas to Thomas and Dorlee Henderson. He entered Grinnell College in 2005, and graduated with a bachelor's in mathematics in 2009. He enrolled in the master's of computer science program at the University of Texas at El Paso in 2010. While enrolled at UTEP, he has worked with Prof. Stefano Bistarelli in Perugia, Italy. He has presented his work to the Numerical Computations, Theory, and Algorithms (NUMTA) conference in Falerna, Italy in June 2013 and at The Society for Advancement of Chicanos and Native Americans in Science (SACNAS) in San Antonio in October 2013. Currently, he works at the Army Research Lab Survivability/Lethality Analysis Directorate (ARL/SLAD) in White Sands, New Mexico.

Address: 210 W. California
Apt. #3
El Paso, TX 79902