A logic for making hard decisions

Roussi Roussev and **Marius Silaghi**
Florida Institute of Technology

Abstract

We tackle the problem of providing engineering decision makers with relevant information extracted from data obtained via a process model based on deliberation and voting. We list examples of potential applications from the area of bug-fix scheduling for software, as well as space-vehicles "go"-"no-go" decision making. In such application domains, important decisions have to be made hastily and therefore the decision factors have to be informed timely of the main issues discovered by the teams. A logic is proposed for reasoning with comments available in such deliberations. Search based algorithms are proposed which recommend the best justifications for a decision and retain the voting decisions for interested parties to tally. We have developed a Bayesian network for generating data by simulation based on probabilistic models that we can train from collected deliberation databases. The data generated in this way was used for evaluating the proposed search algorithm, showing how it can provide better than random recommendations of arguments to decision makers.

Introduction

Voting is an important research topic and as such it is heavily studied in the computing and social literature. It enables groups of people to reach a decision. The voting process starts with enumeration of eligible sets of decision makers through a process called census. It continues with discussion, presentation and agreement on the questions asked and the choices offered. After discussions of the merits of each choice, the actual votes are cast. The process finishes with a tally, verification and dissemination of the results.

It can be argued that at least as important as voting itself is the process of establishing the alternatives on the ballot, namely of how alternatives on the ballot are decided. This paper proposes solutions that help voters make educated choices. Voters are often provided with a choice from a set of pre-selected answers and they (hopefully) take the time to study them before important decisions are made. Voters pose a question for discussion, request and tally feedback.

To illustrate the importance of discussing alternatives, we provide interesting examples from common every day decisions familiar to us, computer scientists and engineers. One such important decision is whether to include a particular fix as part of a release. It is an important question that often confronts the management teams against the engineers, and the future success and reputation of the individuals and their organization (whose level of democracy may vary) is often at stake. When preparing to vote, a skilled decision maker organizes the set of prior justifications that she could find into supporting or opposing groups for each choice. If none of the existing justifications seem fit, she can create a new one. Presenting the justifications is where relevance comes into play.

There is a wide body of research in AI on "argumentation frameworks", starting with Dung's paper, which tries to reach conclusions based on logic with no special consideration to the internal structure of the arguments (Dung 1995). (Leite and Martins 2011) extends Dung's framework with votes on arguments. (Eğilmez, Martins, and Leite 2013) further extend it with votes on attacks. (Hunter 2013) takes into account the probabilistic nature of the arguments and extends the framework as appropriate. (Kaci and van der Torre 2008) present frameworks that handle preferences among arguments.

Argumentation methods has been applied to several engineering fields. (Baroni et al. 2015) present an argumentation framework as applied to three engineering decisions: a civil engineer's choice of foundation for a multi-story building, a water engineer's choice of wastewater treatment technologies, and a medical engineer's choices when designing a reusable precise-dosage

syringe.

We argue that it is human nature to be sometimes irrational, incorrect, (un)intentionally inconsistent or even deceptive. We introduce a very basic algorithm for scoring justifications based on bipartite graphs of user defined relationships between those justifications.

Example

In this section, we provide examples of non-trivial engineering decisions. It is indisputable that the productivity tools widely used by engineers (such as bug tracking databases like JIRA and Bugzilla) do a good job at recording decisions. For open source projects, they do a good job at making them publicly available. At the same time, they provide only limited decision supporting functionality, often only in the context of voting on whether an issue should be addressed or not. They do not provide functionality for tracking votes against. They do not provide voting on individual comments. Some allow limited threading to bring basic structure to the debate. Almost all are organized chronologically by creation timestamp and there is hardly any relevance computation. A decision-maker often has to sift through hundreds and potentially thousands of comments before she has the full context (Mozilla bug178993). One is often subscribed to changes made to issues of interest, and while some of the notifications received are indeed insightful, others are just checking for status, corrections to a previous comment or just automated responses. An average decision-maker has to juggle from as little as a few items a day (for a junior engineer) to tens or even hundreds issues per day (for management and cross-domain experts).

This section presents an example of reaching a "go"-"no-go" decision on whether to go ahead with the launch of a space vehicle. It is based on the Challenger decision to go ahead with the launch despite concerns from engineers (most notably from Bob Ebeling and Roger Boisjoly) that in the winter morning of the planned launch date, the temperature was forecasted to fall below the tested limits of some of the components (Bergin 2007). Engineers argued the launch should be postponed given the lack of testing data on the performance of the rubber O-rings during low temperature conditions, i.e. it was suspected that under freezing temperatures the rings may not sufficiently seal and cause catastrophic failure.

Both sides of the go-nogo decision had valid set of arguments. Arguments *for* continuing with the launch included multiple previous delays (AF1), the existence of secondary O-rings that would have provided backup (AF2), poor presentation of the technical arguments against (AF3), the next launch window would be as far as three months in the future (AF4), a safe temperature range was not provided by the engineers (AF5), lack of historical failures with the current design (AF6), redesign was already in progress that was hoped to be completed before any major failure was observed (AF7). Arguments *against* the launch included lack of test data for the O-ring performance at low temperatures (AA1), lack of failover testing ensuring that the secondary O-rings would prevent catastrophic failure in event of primary O-ring failure (AA2), O-rings were critical components (criticality 1) and a backup should not be relied upon (AA3), all launches should be postponed until an O-ring redesign already well under way was completed and the criticality of the O-rings downgraded (AA4).

The actual NASA justification is presumed to have contained a subset of the arguments for launch. The O-ring manufacturer's original justification is presumed to have contained a subset of the arguments against launch. It should be noted that, the O-ring manufacturer has subsequently changed the decision and voted for the launch with no well documented justification. It is not clear if NASA has released its own justification for proceeding with the launch and the level with which engineers agreed with it.

Another example of a non-trivial decision is whether to quickly address a security vulnerability if only complex, and thus risky, fixes are proposed. Engineers and managers are aware of the risks of not fixing the issue, e.g. watching their system being exploited in the wild. At the same time they need sufficient time to understand the impact, evaluate alternatives, "bake" the changes (get them properly reviewed, tested and deployed) and are afraid of causing regressions in quality due to the complexity involved. Everyone gets together (usually in a live triage meeting), presents their arguments for and against, and formally accepts risk through voting. In most engineering organizations, such triage happens on a daily or weekly basis.

Relations

For the context of this paper, a justification contains a set of arguments. It can be classified based on the type of signature that it accompanies (e.g. support, opposition, abstention). While we consider abstention important, in this paper we propose solutions with only two types of justifications: *supporting justifications* and *opposing justifications*. Two justifications are of the same type if both accompany the same type of signature (support or opposition).

Relations between justifications. Unlike classical argumentation where relations are extracted from

formal arguments, we assume that certain relations are explicitly offered by associating them with opaque arguments in the form of an opaque justification. This is commonly done in existing fora, where relations are provided via a threading model (e.g., each comment responds to another comment). While a formal logical argumentation could be used as support for much more complex mechanisms, the mechanism of opaque arguments we use can be seen as a basic case, where arguments in each provided justification form the premise of the associated vote (support or opposition):

$$arguments \rightarrow petition$$

$$arguments \rightarrow \neg petition$$

Another class of relations we support is **refutes**, where a justification is presented as an answer to a different justification that it corrects or enhances.

A third class of relations that we discuss is **subsumes**, claiming that a justification includes all arguments of another justification. It should be noted that refutes and subsumes are claims explicitly introduced by a voter and that may or may not be automatically verifiable or logically correct.

The last type of relation is **more_recent** which orders the justifications by submission date.

$justification$	**claimed_refutes**	$justification$
$justification$	**claimed_subsumes**	$justification$
$justification$	**more_recent**	$justification$

Given that those relations are provided by the voter, there is a varying degree of trust in them. For example, while voting systems try to guarantee the property of non-repudiation, it is still a best effort guarantee. The **more_recent** relation is stronger in a centralized system where there is a centralized time-keeper, and weaker in decentralized systems where clock skew is more prevalent. As one solution to the different levels of trust, we incorporate weights which is discussed further.

Closures

Under the assumption that each voter selects the most complete justification fitting a choice, a transitivity for the relation **claimed_refutes** can be defined as follows:

$$\begin{array}{l} p \textbf{ claimed\_refutes } n' \\ n' \textbf{ claimed\_refutes } p' \\ p' \textbf{ claimed\_refutes } n \\ \qquad\qquad\qquad \rightarrow \quad p \textbf{ claimed\_refutes } n \end{array}$$

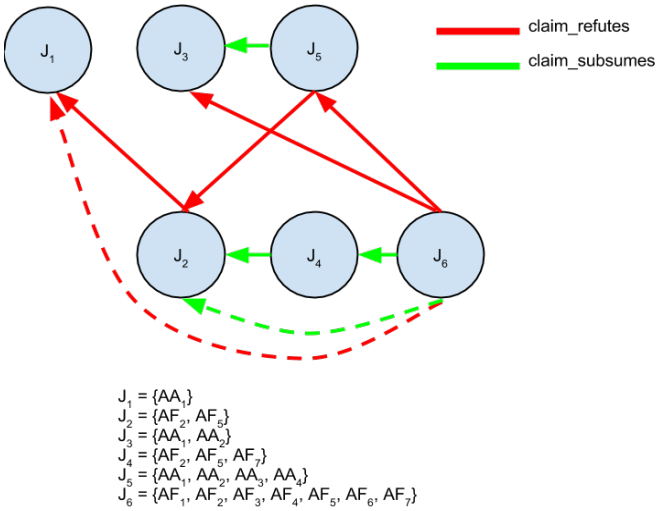
Another type of **transitive closure** is based on the **claimed_subsumes** relation to define a subsume closure with the goal of finding a small set of justifications subsuming most other relevant justifications. The definition of **claimed_subsumes** is simpler than the one of **claimed_refutes**:

$$\begin{array}{l} p \textbf{ claimed\_subsumes } p' \\ p' \textbf{ claimed\_subsumes } p'' \\ \qquad\qquad\qquad \rightarrow \quad p \textbf{ claimed\_subsumes } p'' \end{array}$$

Figure 1 shows an example refute-subsume graph based on the arguments from the Challenger example described earlier. Justification J2 containing arguments {AF2, AF5} is claimed to have refuted a justification J1 containing argument AA1. A more complete justification J4 containing {AF2, AF5, AF7} is claimed to have subsumed justification J2. A justification J5 containing {AA1, AA2, AA3, AA4} is claimed to have refuted a J2. It is also claimed to have subsumed a justification J3 containing {AA1, AA2}. A justification J6 containing {AF1, AF2, AF3, AF4, AF5, AF6, AF7} is claimed to have refuted a justification J3 and J5. It also claimedsubsumed J4. By the refute transitivity above, since J2 claimedrefutes J1 and J5 claimedrefutes J2 and J6 claimedrefutes J5, then J6 claimedrefutes J1. Similarly for subsumes, since J4 claimedsubsumes J2 and J6 claimedsubsumes J4, then J6 claimedsubsumes J2.

Using these special transitivities, one can search for the justifications that (within a limited depth) refute the largest number of justifications of the other type or subsume the largest number of the same type.

Figure 1: Justification graph for the Challenger example



Problem definition

Definition 1 (Answer to a voter) *A justification is said to* **answer** *to a* voter *if either it is associated with the signature of that voter, or if it was created with a specification that it* **claimed_refutes** *or* **claimed_subsumes** *the justification selected by that voter.*

Definition 2 (Representative Arguments Problem) *The Representative Arguments Problem (RAP) for a given petition M consists of a tuple $\langle N, P, V, R, S, K\rangle$. Here $N = \{n_1, ..., n_{m_n}\}$ is a set consisting of m_n opposing justifications of M, and $P = \{p_1, ..., p_{m_p}\}$ is a set of m_p supporting justifications for M.*

Each justification j is associated with a number of v_j signatures, as per the set $V = \{(j, v_j) \mid j \in N \cup P, v_j = signatures(j)\}$. The relation $R \subset (N \times \mathbb{N} \times P) \bigcup (P \times \mathbb{N} \times N)$ associates a weight to each pair between an opposing justification n_i and supporting justification p_j, and to each pair between a supporting justification p_i and an opposing justification n_j, by the **claimed_refutes** *relation. The function $S : P \cup S \rightarrow \mathcal{P}(P \cup S)$ where $S\mid_P : P \rightarrow \mathcal{P}(P)$ and $S\mid_N : N \rightarrow \mathcal{P}(N)$, associates each justification j to a set of justifications of the same type that it* **claimed_subsumes**.

$$
\begin{array}{lcl}
n_i & \xrightarrow{v_{n_i}} & \neg M \\
p_i & \xrightarrow{v_{p_i}} & M \\
p_i & \overset{w^p_{i,j}}{\textbf{claimed\_refutes}} & n_j \\
n_i & \overset{w^n_{i,j}}{\textbf{claimed\_refutes}} & p_j \\
j & \overset{w_{i,j}}{\textbf{claimed\_subsumes}} & k, \forall k \in S(j)
\end{array}
$$

The RAP problem is to find a set of at most K supporting justifications that **answer** *to a maximum number of signatories (both supporting and opposing M), and a set of at most K opposing justifications that* **answer** *to a maximum number of signatories given the defined weights and relations.*

Algorithms

From the perspective of graph theory, the problem of finding the best supporting and opposing justifications can be solved by searching a bipartite graph containing the relations. The algorithm looks similar to mini-max in that it traverses the search tree down to a certain depth. More exactly, in the basic case, one starts with the given justification and in subsequent steps one can apply **kind of** transitivity of the relation.

The algorithm pseudo-code for computing the most encompassing justifications via only the subsume relation is shown in Algorithm 1, which is then extended to take into account the refute relations in Algorithm 2 .

Algorithm 1: Algorithm to find subsuming arguments

```
function subsumes (j, level)
    for any i s.t. j claimed_subsumes i do
        add i to S_1
    for k=1;k≤ level; k++do
        S_k = {i | u ∈ S_{k-1}, u claimed_subsumes i}
    return ⋃_{k=0}^{level} S_k;
```

Algorithm 2: Algorithm to find counter-arguments

```
function refutes (j, level)
    for any i s.t. j claimed_refutes i do
        add i to R_1;
        add subsumes(i,level-1) to R_1;
    for k=1;k≤ level; k++do
        R_k = {subsumes(i, level − k) |
                u ∈ R_{k-1}, ∃t, v, u claimed_refutes t ∧
        v claimed_refutes i,
                v ∈ subsumes(t, level − k) ∪ t}
    return ⋃_{k=0}^{level} R_k;
```

This function can be applied to all justifications (for some level), and then one can compute the cardinality of the results to estimate the justifications containing the most arguments.

$$J_{level} := \max_j \mid refutes(j, level) \mid$$

One can integrate the votes on justifications and rela-

tions as weights to arguments, recognizing the difference in importance. They can further be discounted with a factor $\gamma <= 1$ to consider their depth in the tree:

$$\sum_{k=0}^{level} \gamma^k * votes(k)$$

where $votes(k)$ integrate the number of signatures for all justifications at level k as well as for the **claimed_refutes** and **claimed_subsumes** relations in the directions used for the transitivity: $votes(k) = votes_j(k) + \alpha * votes_{\rightarrow}(k)$.

To implement the weighted algorithm, one can substitute the union operation (line 6 of algorithm 1 and line 9 of algorithm 2) with the sum on the total scores of the justifications at each level and use a priority queue to keep the top justifications. It should be noted that the cardinality-based algorithm is a special case of the weighed one, when gamma is set to one and alpha is set to zero.

Experimental results

We have built a Bayesian network to generate arguments, justifications based on those arguments and relationships between justifications. In such a network corresponding to an instance of a problem, there are three nodes, A_j, R_j and S_j for each justification j. Each justification is introduced in the network in the order defined by the **more_recent** relation. The domain of A_j is the power set of A_M, $\mathcal{P}(A_M)$. The domain of R_j is the set of possible justifications (that are less recent than j), and specifies a justification of a different type that j **claimed_refutes**. Similarly, S_j specifies a justification of the same type that j **claimed_subsumes**.

We used that Bayesian network to generate 1000 justifications over 100 arguments. Each justification contains a uniform random subset of arguments from the power set of all arguments. Given empirical data that people are more likely to respond if they disagree than when they agree (Agrawal et al. 2003), we select our parameters to refute a justification 85 percent of the time and subsume 15 percent of the time. To limit the size of our network, we allow a justification to refute at most 10 other justifications and to subsume at most 3 ones. The votes are distributed proportionally with the total number of decision makers, the number of arguments a justification contains and how old a justification is under the assumption that the better a justification is, the more times it will be chosen, and the older it is the more opportunity it has had to get votes. As justifications are inserted into the network with increasing timestamp during data generation, we make sure that a justification refutes/subsumes only an earlier one and with distribution proportional to the number of arguments that it contains. We executed one hundred runs of the cardinality-based and weighted search algorithms on freshly generated data for the run. The algorithms find the best justifications 64 percent of the time which is improvement over chronological or random order. Of those, the cardinality based algorithm (which is a degenerate case of the weighted algorithm with gamma set to one and alpha set to zero, as described earlier) found the best justification 47-52 percent of the times when the best justifications were found. We also computed the best sets of values for gamma and alpha which are specific to each instance of the Bayesian network. We have found that the best justifications are usually in the top five levels, hence we limit the depth of the search to eight levels.

Conclusion

We believe that the methods that we describe are a small step towards enabling decision makers to reach better supported and agreed upon decisions. This is achieved by presenting the best possible justifications for each choice based on a set of user-annotated relations (**claimed_refutes** and **claimed_subsumes**) and metadata generated by the system (**more_recent**). While at present we take those relations for granted, using natural language processing to extract them is an area of future research for us.

References

[Agrawal et al. 2003] Agrawal, R.; Rajagopalan, S.; Srikant, R.; and Xu, Y. 2003. Mining newsgroups using networks arising from social behavior. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, 529–535. New York, NY, USA: ACM.

[Baroni et al. 2015] Baroni, P.; Romano, M.; Toni, F.; Aurisicchio, M.; and Bertanza, G. 2015. Automatic evaluation of design alternatives with quantitative argumentation. *Argument and Computation*.

[Bergin 2007] Bergin, C. 2007. Remembering the mistakes of challenger. *NASASpaceFlight.com*.

[Dung 1995] Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence* 77(2):321–357.

[Eğilmez, Martins, and Leite 2013] Eğilmez, S.; Martins, J.; and Leite, J. 2013. Extending social abstract argumentation with votes on attacks. In *Theory and Applications of Formal Argumentation*. Springer. 16–31.

[Hunter 2013] Hunter, A. 2013. A probabilistic approach to modelling uncertain logical arguments. *Inter-*

national Journal of Approximate Reasoning 54(1):47–81.

[Kaci and van der Torre 2008] Kaci, S., and van der Torre, L. 2008. Preference-based argumentation: Arguments supporting multiple values. *International Journal of Approximate Reasoning* 48(3):730–751.

[Leite and Martins 2011] Leite, J., and Martins, J. 2011. Social abstract argumentation. In *IJCAI*, volume 11, 2287–2292. Citeseer.

[Mozilla bug178993] Mozilla bug178993. `https://bugzilla.mozilla.org/show_bug.cgi?id=178993`.