A System for Updating Trust and Performing Belief Revision

Aaron Hunter and Sam Tadey
Department of Computing, British Columbia Institute of Technology, Burnaby, Canada

Keywords: Belief Revision, Trust, Systems.

Abstract: The process of belief revision is impacted by trust. In particular, when new information is received, it is only believed if the source is trusted as an authority on the given information. Moreover, trust is actually developed over time based on the accuracy of past reports. Any practical tool for representing and reasoning about the beliefs of communicating agents will therefore require some mechanism for modeling trust, both in terms of how it changes over time and in terms of how it impacts belief revision. In this paper, we present such a tool. We use so-called trust graphs to give a compact representation of how strongly one agent trusts another to distinguish between possible states of the world. Our software allows a trust graph to be updated incrementally by looking at the accuracy of past reports. After constructing a trust graph, the software can then compute the result of AGM-style belief revision using two different approaches to incorporating trust. In the first approach, trust is treated as a binary notion where an agent is either trusted to distinguish certain states or they are not. In the second approach, the relative strength of trust is compared directly with the strength of the initial beliefs. The end result is a tool that can flexibly model and reason about the dynamics of trust and belief.

1 INTRODUCTION

Belief revision refers to the process in which an agent's beliefs change in response to new information. In practical settings, new information is often obtained from other agents. As such, there is a connection between belief revision and trust; we only want to incorporate new information if we trust the reporting agent to be an authority on the topic. The trust that we hold in other agents is built by looking at past reports, to see how accurate they have been. As such, a complete treatment of belief change in a multi-agent setting requires a framework that can perform the following steps:

1. Create a model of trust for a *reporting agent*.
2. Update the model based on the accuracy of past reports.
3. Calculate the result of the revision from a new report, taking trust into account.

In this paper, we describe a software tool that can perform all of these steps. The tool is based on the notion of *trust graphs*, as defined in (Hunter, 2021).

We make several contributions to the literature on belief revision and trust. First, the software presented here is a useful addition to the relatively small collection of existing belief revision solvers, because it extends the class of practical problems that we can model and solve. To the best of our knowledge, the software presented in this paper is the first implemented system that incrementally builds a model of trust that is specifically intended to inform the process of belief revision. The work here also makes a contribution to our understanding of the connection between revision and trust. Our software allows two different approaches to revision that incorporate trust in different ways. By implementing both approaches, we make it possible to experiment with alternative views on the relationship between strength of belief and strength of trust.

2 PRELIMINARIES

2.1 Belief Revision and Trust

We are interested in belief revision in the setting of propositional logic. We assume a *finite* set V of propositional variables that can be combined with the usual propositional connectives $\neg, \wedge$ and $\vee$. A *state* is a propositional interpretation of V, and we let S denote the set of all states.

We briefly introduce two well-known approaches to belief revision. The most influential approach is the

Hunter, A. and Tadey, S.
A System for Updating Trust and Performing Belief Revision.
DOI: 10.5220/0011614800003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 55-62
ISBN: 978-989-758-623-1; ISSN: 2184-433X

so-called AGM approach (Alchourrón et al., 1985). In the AGM approach, the beliefs of an agent are represented by a set of formulas Ψ. An AGM revision function $*$ takes the initial belief state Ψ and a formula ϕ as input. This pair is mapped to a new belief state $\Psi * \phi$, subject to a set of rationality postulates. It is well known that every AGM revision operator $*$ can be defined in terms of minimization with respect to a total pre-order over states (Katsuno and Mendelzon, 1992).

An alternative model for reasoning about beliefs is by using a *ranking function* over states to represent the beliefs of an agent (Spohn, 1988). A ranking function κ is a function that maps every state to a natural number, with the constraint that $\kappa(s) = 0$ for at least one state. Informally, if $\kappa(s) \leq \kappa(t)$, we interpret this to mean that the agent considers it more likely that the actual state of the world is s as compared to t. A ranking function representing beliefs is sometimes called an *epistemic state*, as it includes the current beliefs as well as information about how these beliefs will be changed when new information is obtained.

In our software, we internally use the ranking function approach for modeling epistemic states. However, we will see that our software also permits users to enter beliefs as a set of formulas, which is then extended to an epistemic state using a default ranking.

2.2 Trust Graphs

A *trust graph* is a representation of the trust that an agent holds in an information source.

Definition 1 ((Hunter, 2021)). *Let S be the set of states over V. A* trust graph *over S is a pair $\langle S, w\rangle$, where $w : S \times S \rightarrow \mathbf{N}$.*

Hence, a trust graph is just a weighted graph where the nodes represent states, and the weights are distances between states. Informally, the distance between two states represents how strongly we trust the reporting agent to be able to distinguish them.

Example Suppose that an agent gets information about the weather from the radio. They strongly trust the announcer to be able to tell if it is sunny outside (S). On sunny days, they also have moderate trust that the announcer can tell if it is humid (H); however, they have no trust at all in the announcer's ability to know the humidity when it is not sunny. This can be captured by the trust graph in Figure 1.

2.3 Update Rules

A trust graph is not static; it should be updated based on the accuracy of reports provided by a particular

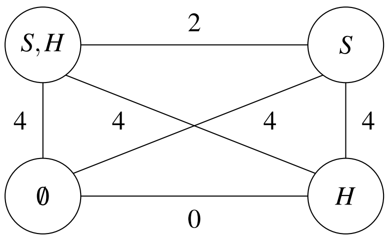


Figure 1: A Trust Graph for Weather Reports.

agent. We can define a *report* to be a pair (ϕ, m) where m is either 0 or 1. If $m = 0$ (resp. 1), this is interpreted to mean that ϕ was falsely (resp. correctly) reported in the past.

Suppose that an agent reports ϕ, and we subsequently learn that ϕ is false. In this case, we should now have less trust in the reporting agent's ability to know if ϕ is true. This means that, following a false report, we should *increase* the distance between states where ϕ is true and states where ϕ is false. Similarly, if an agent provides an accurate report of ϕ, then we should *decrease* the distance between such pairs of states.

There are many different ways to update the distances on a trust graph. As an illustration, we consider the following simple *additive* update rules.

Update Rule 1. *Given an initial trust graph over S and a report $(\phi, 0)$, update the graph as follows:*

- *For each pair of states s_1, s_2 such that $s_1 \models \phi$ and $s_2 \not\models \phi$ decrease the value $w(s_1, s_2)$ to $w(s_1, s_2) - 1$.*

Update Rule 2. *Given an initial trust graph over S and a report $(\phi, 1)$, update the graph as follows:*

- *For each pair of states s_1, s_2 such that $s_1 \models \phi$ and $s_2 \not\models \phi$, increase the value $w(s_1, s_2)$ to $w(s_1, s_2) + 1$.*

According to the first rule, a false report of ϕ makes an agent have less trust in the reporting agent's ability to distinguish ϕ-states from $\neg\phi$-states. According to the second rule, a true report of ϕ makes an agent have more trust in that distinction.

Example Consider the weather reporting example. Suppose that the announcers says it is sunny outside, but then we go outside and we find that it is not sunny. This report is formally represented as (S,0). According to Update Rule 1, we need to decrease the distances on states where S is true and those where S is false. The new trust graph is given in Figure 2.

Note that Update Rules 1 and 2 are simply intended to provide an example of the process; we do not intend to assert that these are the most appropriate update rules in practice. In fact, there is clearly a problem with Update Rule 1 in that it can actually

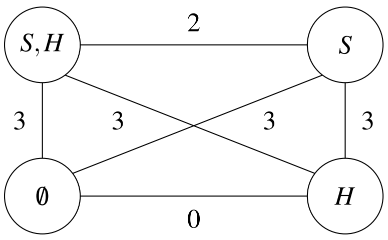


Figure 2: Updated Trust Graph.

lead to negative edge weights. This problem can be fixed by allowing negative weights during the update, and then normalizing at the end. This problem can also be fixed by adding constraints to the weights on the initial trust graph, or by introducing more sophisticated rules. We return to this problem later in our discussion of the our implemented system. We will see that our software allows for very flexible update rules to be defined. But for the purpose of motivation and illustration in this section, we believe it is useful to give concrete, simple rules that are easily understood.

3 IMPLEMENTATION

3.1 Functionality

We describe *T-BEL* , a Java application for modeling the dynamics of trust and belief. The core functionality of *T-BEL* is as follows. It allows a user to create an initial trust graph over different information sources, and it then allows a user to enter a series of reports which might be correct or incorrect. These reports trigger an update to the trust graph. Finally, the user can calculate the result of belief revision, in a manner that accounts for the influence of trust.

Note that the steps listed above need not be done sequentially. The interface for the software provides several panels for different actions: initializing a trust graph, manipulating the trust graph, visualizing the trust graph, and performing revision. The only constraint is that the vocabulary needs to be provided to initialize the trust graph. After the initial trust graph is constructed, a user can jump between different panels. For example, one could add new information about past reports at any time, even after revision has been performed.

3.2 Constructing a Trust Graph

In order to perform belief revision using *T-BEL* , we first need to initialize a trust graph. This is done

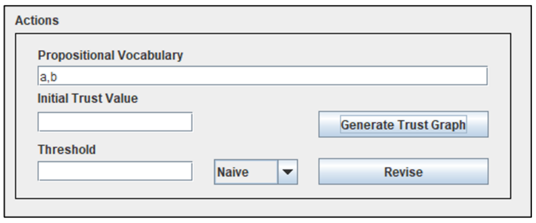


Figure 3: Initializing a Trust Graph.

through the panel in Figure 3. The user simply enters a propositional vocabulary as a comma delimited sequence of strings. Optionally, one can specify an initial trust value; this is the weight that will be assigned to all edges in the trust graph. If it is not specified, it will default to 1.

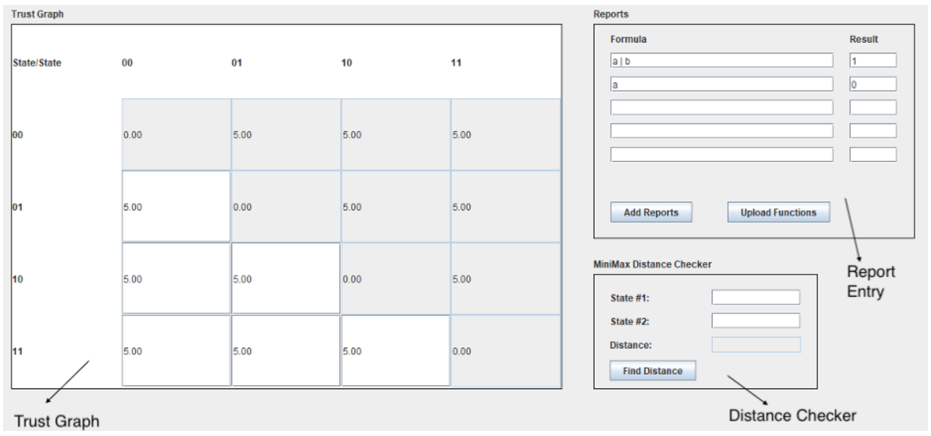
Note that, with the initial trust value of 1, the weights can become negative after as few as two revisions using the default update rules. As such, it is recommended that users set an initial trust value that is larger than the number of reports that will be entered. However, this constraint is not enforced; we would like to leave open the possibility that the user does not know the number of reports to be entered. Internally, weights are additively normalized to get a minimum value of zero. This is not a problem for the most basic problems, but it can be questionable when we move to different update rules.

The trust graph is displayed in Figure 4 as a matrix. In the image, the initial trust value has been set to 5, and the user is shown the weights between every state. Note that the propositional variable names do not appear here. The state 10 is listed to indicate the state where **a** is assigned true and **b** is assigned false. The order of variables for these states is just the order that they were entered when defining the propositional vocabulary.

The main goal of *T-BEL* is to allow trust to be built incrementally by adding reports. This is done through the *report entry* section in Figure 5. Reports are entered as formulas in a simple variant of propositional logic, using the keyboard-friendly symbols & (conjunction), | (disjunction) and − (negation). The reports are tagged with 1 (positive) and 0 (negative). By default, when the *Add Reports* button is pressed, the matrix on the left updates the values in accordance with Update Rule 1 and Update Rule 2.

There is one remaining feature shown in Figure 4: the *Distance Checker*. We will see in the next section that we actually do not use the values in the trust matrix directly; we use the minimax distance generated from these values. As such, we provide the user with a simple mechanism for checking minimax distance. This is useful for testing and experimentation.

While the default operation of *T-BEL* assumes


Trust Graph

State/State	00	01	10	11
00	0.00	5.00	5.00	5.00
01	5.00	0.00	5.00	5.00
10	5.00	5.00	0.00	5.00
11	5.00	5.00	5.00	0.00

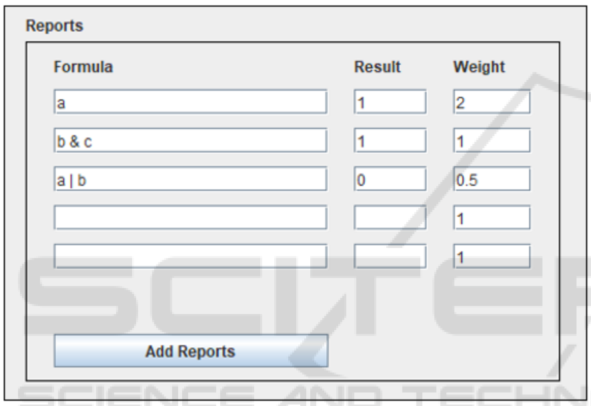
Reports

Formula	Result
a | b	1
a	0

MiniMax Distance Checker

State #1:
State #2:
Distance:

Figure 4: The Trust Panel.


Reports

Formula	Result	Weight
a	1	2
b & c	1	1
a | b	0	0.5
		1
		1

Figure 5: Initializing a Trust Graph.

that reports update the trust graph in increments of 1, there is actually an optional feature that can be used to change this behaviour. We can run the *T-BEL* in an enhanced mode, where the report entry panel has the form displayed in Figure 5. The difference here is that there is a third column for specifying a weight for each report. By using this column, the user can indicate that different reports should increase (or decrease) the weights in trust graph by a different amount. For example, with the values in Figure 4, the report of ’a’ will increase all affected edges by 2. This weighting can be used to indicate that a particular report carries a different weight in terms of building the new trust graph.

3.3 Implications of the Implementation

It is possible to validate that the trust graphs produced by *T-BEL* are actually correct. The trust panel in Figure 4 presents a matrix giving all of the weights explicitly. In order to validate, we performed a series of tests with graphs with different initial vocabularies and compared the matrix with the graphs generated by hand. Unfortunately, when the vocabulary is larger than 4 variables, the matrix becomes small and difficult to read. In order to validate these examples, we need to script the output to print all weights line by line for validation. The process here can be tedious, but all tests were successfully passed.

The bigger problem with the implementation is that we are interested in using these graphs to perform belief revision in the next section. While the trust graph over ten variables can be computed and manipulated quickly, even at that scale the revision calculations in the next section become prohibitively slow. Performance issues are discussed after introducing our approach to revision in the next section.

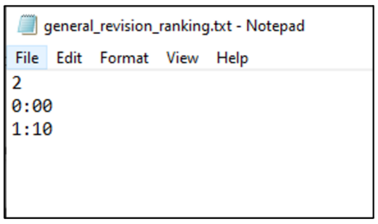
4 REVISION AND TRUST

4.1 Specifying an Epistemic State

Generating a trust graph is just the first step in implementing an approach to revision that incorporates trust. We now discuss the next two steps: defining an epistemic state, and then implementing suitable approach to belief revision.

As noted previously, epistemic states are represented in *T-BEL* using ranking functions. The software provides two different ways to specify an epistemic state.

The first way to specify an epistemic state is by explicitly specifying a total pre-order over all states. This is done by creating an external text file that lists a “level” for all states starting from 0. For example, if we had two variables A and B, then one example input file is shown in Figure 6. In this file, the first


general_revision_ranking.txt - Notepad
File Edit Format View Help

```
2
0:00
1:10
```

Figure 6: File Input.

line indicates that there are 2 variables. The second line says that the state where A and B are both false is the most plausible, so it is the only state in level 0. The next line specifies the states in level 1. Any states not listed are put in level 2. A ranking over states specified in this manner gives us enough information to perform belief revision.

Manually specifying a complete ranking in this manner can be problematic, because it is time consuming and it is easy to make mistakes. As such, we also give the user the ability to experiment with revision simply by entering a belief state as a set of formulas through an input box in the main interface. For example, we could enter the beliefs by giving this list of formulas:

```
A&B
A|-B
```

To generate a ranking function from such a list, *T-BEL* finds all satisfying assignments of the formulas specified. *T-BEL* then uses the Hamming distance from the set of satisfying assignments to create a full ranking. In other words, the default approach defines a ranking that corresponds to Dalal's revision operator (Dalal, 1988). This approach is suitable for many applications, and allows users to get the software running without a detailed ranking over states. The problem, of course, is that this approach requires the software to solve a satisfiability problem. However, this is not a significant problem for the prototype software as it has only been tested on problems with a small set of variables. For larger problems, a competition-level SAT solver could be used for this step. The effectiveness of this approach for belief revision problems has previously been demosntrated in (Hunter and Agapeyev, 2019).

4.2 Naive Revision

T-BEL implements two different approaches for incorporating trust into the belief revision process; the user chooses the mechanism to be used in the menu in Figure 3. The first approach is called *Naive Revision*. The intuition behind naive revision is that we consider the rankings for beliefs and trust to be independent. In other words, it is not possible to compare strength of belief and strength of trust, because we assume that they are on different scales. The approach that we use in this case is to use the trust graph and the initial belief state to define a *trust-sensitive revision operator* (Booth and Hunter, 2018). In this section, we briefly describe how this is implemented in *T-BEL* .

A trust sensitive revision operator is defined with respect to a trust partition Π over states. Informally, the reporting agent is only trusted to be able to distinguish between states that are in different cells of this partition. Trust-sensitive revision works as follows. Given a formula ϕ, let $mod(\phi)$ denote the set of all states where ϕ is true. Then let $\Pi(\phi)$ denote the union of all cells of Π that contain an element of $mod(\phi)$. To calculate the result of the trust-sensitive revision by ϕ, we actually revise by a formula ϕ' with $mod(\phi') = \Pi(\phi)$. So we are essentially revising by the set of all states that are indistinguishable from models of ϕ, from the perspective of the trust partition Π. We refer the reader to (Booth and Hunter, 2018) for a complete description of this operation.

For our purposes, it is sufficient to note that *every* partition over the set of states defines a trust-sensitive revision operator. In order to obtain such a partition, we can use the following result.

Proposition 1 ((Hunter, 2021))**.** *Let T be a trust graph, let d_T be the minimax distance between vertices, and let m be a natural number (the threshold value). For any state s, let $X_s = \{t \mid d_T(s,t) \leq m\}$. The collection of sets X_s over all states forms a partition of S.*

This result holds because the minimax distance defines an ultrametric. This is in fact why we use the minimax distance, because it defines a collection of partitions that are suitable for reasoning about trust.

Following Proposition 1, we know that we can define a trust-sensitive revision operator from the trust graph by specifying a threshold value to define the partition. This is why Figure 3 includes a field for a threshold value. The user simply enters a threshold here, and that number is used to define the revision operator for Naive Revision.

Example Returning to the weather example. Suppose that we set a threshold of 3. This defines a partion with two cells Π_1 and Π_2:

$$\begin{aligned} \Pi_1 &= \{\{S,H\},\{S\}\} \\ \Pi_2 &= \{\{H\},\emptyset\} \end{aligned}$$

This partition intuitively indicates that the radio announcer is trusted to determine if it is sunny or not, bu they are not trusted to determine if it is humid. If the announcer says that it is sunny and humid, we represent this with the formula $\phi = S \wedge H$. But when we perform trust-sensitive revision, then we would actually revise by $(S \wedge H) \vee (S \wedge \neg H)$ because the announcer is not trusted to tell the difference between these states.

Informally, specifying a high threshold value will mean that the reporting agent is not trusted to distinguish between many states; in this case, the reports they provide will not result in drastic changes of belief. On the other hand, a low threshold value will mean that the reports will be taken at closer to face-value. The recommendation for users with no knowledge of trust-sensitive revision is to set an initial threshold value of 1, which means that reports are trusted unless the reporting agent has previously provided a false report on the same topic.

T-BEL is able to perform all of this automatically. The user simply enters the threshold vavlue in the panel in Figure 3, and then clicks revise. They will be prompted to enter a formula for revision, and the computation will be done based on the partition generated. The result of revision is displayed as a formula, capturing the minimal states in the new ranking.

4.3 General Revision

In the *General Revision* approach, we assume the rankings for trust and belief are on the same scale, so they are comparable. This allows us to model situations where an agent weighs how strongly they believe something against how strongly they trust an information source. In other words, if we *strongly* believe ϕ to be true, then we will not believe $\neg\phi$ when reported by an agent that is only *weakly* trusted. In this case, we want to consider the interaction between strength of belief and strenght of trust. Roughly, *strength* here is indicated by the values assigned by different ranking functions.

We can specify that we want to use general revision in the dropdown menu in Figure 3. In this case, we essentially have two ranking functions that we would like to combine. On one hand, we have κ - the ranking function over states representing the beliefs of an agent. On the other hand, we have a minimax distance function d over states defined by the trust graph. When this function d is paired with a formula ϕ for revision, it defines a second ranking function over states. Using this notation, we define a new function through the following formula:

$$\kappa_d^{\phi}(s) = \kappa(s) + \min\{d(s,t) \mid t \models \phi\}.$$

Of course, this may not be a ranking function because it is not guaranteed to take the value 0. The new ranking function following revision is obtained by normalizing κ_d^{ϕ} by subtracting the minimimum value from the ranks of all states.

Example We return to the weather reporting example. Suppose that we start with the initial values from Example 1, and then we receive 5 correct reports of sunshine in a row. So the new distances from $\{S\}$ in the trust graph are as follows:

$$\begin{aligned} d(\{S\},\{S\}) &= 0\\ d(\{S\},\{H\}) &= 9\\ d(\{S\},\{S,H\}) &= 2\\ d(\{S\},\emptyset) &= 9 \end{aligned}$$

Now suppose that our initial belief state κ has the property that $\kappa(\{S,H\}) = 2$ and $\kappa(s) = 0$ for all other states s. So we initially believe it is definitely not sunny and humid. Suppose further that the current weather report says that it is $S \wedge \neg H$, indicates that it is sunny and dry. We calculate $\kappa_d^{\phi}(s)$ for all states s in the following table:

s	$\kappa(s)$	$d(\{S\},s)$	$\kappa_d^{\phi}(s)$
$\{S\}$	2	0	2
$\{H\}$	0	9	9
$\{S,H\}$	0	2	2
$\emptyset$	0	9	9

Since the first and third rows both have minimal values, it follows that the new belief state is $\{\{S\},\{S,H\}\}$. This means that we now believe it is sunny, and it may or may not be humid. This result balances our initial strength of belief with the fact that the reporter is strongly trusted to know when it is sunny.

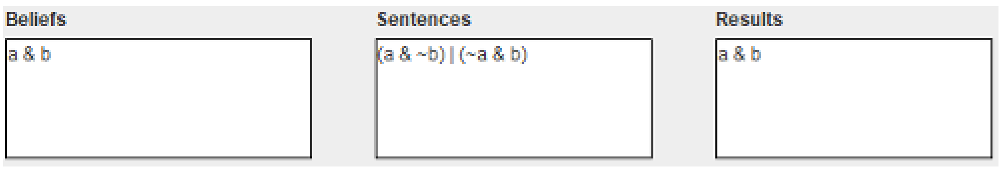
We refer the reader to (Hunter, 2021) for a more detailed discussion of rationale behind this approach. For the moment, we simply indicate that the most plausible states obtained with this operator will be those with the lowest aggregate of strength of belief and strength of trust.

4.4 Step by Step Example

In this section, we walk through a complete example using all features of *T-BEL* to define a trust graph, add reports, and then calculate the result of revision.

Assume we want to work with the vocabulary $\{a,b\}$, as well as past reports of $(a \vee b, 1)$ and $(a,1)$. Assume further that we would like to start with the belief state $(a \wedge b)$ and then revise by $(a \wedge \neg b) \vee (\neg a \wedge b)$. Using *T-BEL* , then can solve this problem through the following steps:

1. Enter the vocabulary a,b and a default value of 5.
2. Enter reports $(a|b,1)$ and $(a,1)$ then click *Add Reports*.



a & b

(a & ~b) | (~a & b)

a & b

Figure 7: Revision Output.

3. Select Naive revision with threshold 3.
4. Enter the belief state $a\&b$ and formula $(a\&-b)|(-a\&b)$.
5. Click *Revise*.

The default value in step 1 should be set so that it is at least as high as the number of reports. However, beyond that constraint, it will not impact the results. After step 2, the values in the matrix representing the trust graph will be as follows:

	00	01	10	11
00	0	6	7	7
01	6	0	6	6
10	7	6	0	5
11	7	6	5	0

The revision panel following the example is in Figure 7, showing the input and the fact that the beliefs are unchanged after revision. It can easily be verified that this is correct.

4.5 Iterated Revision

The problem of iterated revision is well-known in the belief revision literature. The highly-influential AGM approach to revision suffers from the fact that it can only be used for a single revision, because the result of belief revision does not include the full ordering required for subsequent revisions. For our software, users can certainly calculate the result of iterated revision by pressing the revision button several times. However, we need to be clear on how well this captures an approach to rational belief change.

For Naive Revision, the situation is slightly complicated. If the user has specified the initial belief state as a formula, then they are performing Dalal revision. This is an operation that can be applied repeatedly, so *T-BEL* can be used to solve iterated belief change in this case. However, if the user has used a custom initial ranking through the file-based interface, then this is a problem. The result of Naive Revision is a set of states, not a full ranking. As such, Naive Revision with a custom ranking is not suitable for iterated belief change. At a practical level, *T-BEL* will still let the user perform several revisions. However, it is important to note that subsequent revisions will default to Dalal revision regardless of the initial ranking.

For General Revision, the situation is better. While *T-BEL* displays the result of General Revision as a formula, this is just for ease of readability for the user. The formula displayed specifies the minimal states in the new ranking, but the full ranking is internally maintained. As such, iterated revision can be performed using the new orderings as they are modified.

5 DISCUSSION

5.1 Performance

The question of run time is a challenging one to address for any implemented belief revision system, due to the well known compexity of revision (Eiter and Gottlob, 1992). The problem is even worse when we add trust graphs, which become very large as the vocabulary size increases.

We made many implementation choices in order to optimize performance. For example, we represent a trust map internally as a hashmap of hashmaps; the lookup time is very fast. Another place where we focus on efficiency is in the translation from formulas to belief states, where we use a DPLL solver to find satisfying assignments. However, the run time for *T-BEL* still becomes slow as the vocabulary size increases. It is a useful prototype for reasoning about small examples, and demonstrating the utility of trust graphs. In future work, we will look to improve run time by integrating a competition level ALLSAT solver for the hard calculations (Toda and Soh, 2016).

5.2 Related Work

Fundamentally, this work is about software to support reasoning about knowledge-based trust; this problem has previously been explored in the practical context of evaluating web sources (Dong et al., 2015). There has also been related formal work on the relationship between trust and belief (Booth and Hunter, 2018; Liu and Lorini, 2017), as well as emerging work on truth discovery (Singleton and Booth, 2020). In terms of implemented systems, *T-BEL* can be seen as an extension of the GenB system (Hunter and Tsang, 2016). GenB is a general solver for revision with a limited capacity to capture trust; *T-BEL* is significantly more sophisticated when it comes to representing and reasoning about the dynamics of trust and belief.

6 CONCLUSION

In this paper, we have described a tool for solving belief change problems influenced by trust. The focus is on building trust from reports, and then performing belief revision.

Our software provides a simple interface that can be used to build a trust graph iteratively, and then this graph is used to adjust the behaviour of a formal belief change operator to account for trust. We suggest that this tool is an important step towards demonstrating the utility of belief change operators for solving practical problems with partially trusted information sources. In future work, we intend to improve run time performance, apply the tool to concrete problems in the evaluation of web resources, and connect our approach to related work on learning with respect to trust.

REFERENCES

Alchourrón, C. E., Gärdenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic*, 50(2):510–530.

Booth, R. and Hunter, A. (2018). Trust as a precursor to belief revision. *J. Artif. Intell. Res.*, 61:699–722.

Dalal, M. (1988). Investigations into a theory of knowledge base revision. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 475–479.

Dong, X., Gabrilovich, E., Murphy, K., Dang, V., Horn, W., Lugaresi, C., Sun, S., and Zhang, W. (2015). Knowledge-based trust: Estimating the trustworthiness of web sources. *Proceedings of the VLDB Endowment*, 8.

Eiter, T. and Gottlob, G. (1992). On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence*, 57(2-3):227–270.

Hunter, A. (2021). Building trust for belief revision. In *Proceedings of the Pacific Rim Conference on Artificial Intelligence (PRICAI)*, pages 543–555.

Hunter, A. and Agapeyev, J. (2019). An efficient solver for parametrized difference revision. In *Proceedings of the Australasian Conference on Artificial Intelligence*, pages 143–152.

Hunter, A. and Tsang, E. (2016). GenB: A general solver for AGM revision. In *Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA)*, pages 564–569.

Katsuno, H. and Mendelzon, A. (1992). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(2):263–294.

Liu, F. and Lorini, E. (2017). Reasoning about belief, evidence and trust in a multi-agent setting. In An, B., Bazzan, A. L. C., Leite, J., Villata, S., and van der Torre, L. W. N., editors, *PRIMA 2017: Principles and Practice of Multi-Agent Systems - 20th International Conference, Nice, France, October 30 - November 3, 2017, Proceedings*, volume 10621 of *Lecture Notes in Computer Science*, pages 71–89. Springer.

Singleton, J. and Booth, R. (2020). An axiomatic approach to truth discovery. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 2011–2013.

Spohn, W. (1988). Ordinal conditional functions. A dynamic theory of epistemic states. In Harper, W. and Skyrms, B., editors, *Causation in Decision, Belief Change, and Statistics, vol. II*, pages 105–134. Kluwer Academic Publishers.

Toda, T. and Soh, T. (2016). Implementing efficient all solutions sat solvers. *ACM Journal of Experimental Algorithmics*, 21(2):1–44.