TrustMe 1.0 ©

A Social Simulation of Trust, Advice and Gossip

MJ PRIETULA
Goizueta Business School
Emory University

Draft: Jan 10, 2006

Supports Version: TrustMe1r7

Working Paper

gbspapers.library.emory.edu

© 2006 by MJ Prietula

Project supported by a grant from The National Science Foundation, Directorate for Computer, Information Science & Engineering Computational Social Systems Program, Award IIS-0084508
Table of Contents

Table of Contents ..................................................................................................................................................................................... 1

Acknowledgements .................................................................................................................................................................................. 3

Installing TrustMe Software .................................................................................................................................................................. 2

Quick Guide ....................................................................................................................................................................................................... 4

TrustMe: Trust, Advice and Gossip .................................................................................................................................................. 5

TrustMe Theory ........................................................................................................................................................................................................ 6
Trust ............................................................................................................................................................................................................. 6
Advice ....................................................................................................................................................................................................... 7
Gossip ........................................................................................................................................................................................................... 7
Induced Simplicity Hypothesis ............................................................................................................................................................. 8
Modeling as Method ...................................................................................................................................................................................... 8

TrustMe Problems .................................................................................................................................................................................................. 8
Example 1 – Physical Locations .......................................................................................................................................................... 9
Example 2 – Virtual Locations ............................................................................................................................................................. 9

TrustMe Metrics .................................................................................................................................................................................................. 9
Effort Metric 1 .................................................................................................................................................................................................... 10
Effort Metric 2 .................................................................................................................................................................................................... 11
Effort Metric 3 .................................................................................................................................................................................................... 11

TrustMe Theory .................................................................................................................................................................................................. 13

Bounded Rationality .................................................................................................................................................................................................. 13
Limited Search .................................................................................................................................................................................................. 13

Limited Memory .................................................................................................................................................................................................. 14
Location Memory ................................................................................................................................................................................................... 14
Fallible Memory ................................................................................................................................................................................................... 14

Limited Attention .................................................................................................................................................................................................. 15
Advice ............................................................................................................................................................................................................. 15
The Task Environment ............................................................................................................................................................................. 15

TrustMe Interactions .................................................................................................................................................................................................. 16
Role of Communication .............................................................................................................................................................................. 16

Selecting Trust Models .................................................................................................................................................................................................. 17

Advice Resolution Strategies ........................................................................................................................................................................ 20
Strategy 1: Random .................................................................................................................................................................................................. 22
Strategy 2: Maximum Number ................................................................................................................................................................. 22
Strategy 3: Random from Trusted ............................................................................................................................................................. 22
Strategy 4: Maximum from Trusted ............................................................................................................................................................. 22

Example: Advice and Trust ........................................................................................................................................................................ 23

Trust ............................................................................................................................................................................................................. 27
Trust Models .................................................................................................................................................................................................. 29
Model 1: Trusting .................................................................................................................................................................................................. 29
Model 2: Distrusting .................................................................................................................................................................................................. 29
Model 3: Random Judgment ................................................................................................................................................................. 29
Model 4: Forgiving .................................................................................................................................................................................................. 29
Model 5: Unforgiving .................................................................................................................................................................................................. 30

Honesty Models .................................................................................................................................................................................................. 32
Model 1: Honest .................................................................................................................................................................................................. 32
Model 2: Trusted .................................................................................................................................................................................................. 32
Model 3: Trusted with Gossip ................................................................................................................................................................. 32
Model 4: Trusted with Gossip, Deceptive .................................................................................................................................................. 32
Acknowledgements

TrustMe software is provided “as is” for pedagogical purposes. I do, of course, appreciate feedback and ideas. Updates will be made available on web (see below). Thanks are provided to the several members of the Computational and Mathematical Organization Theory group who have helped shape this model and approach, including Rich Burton (Duke University), Kathleen Carley (Carnegie Mellon), Ray Levitt (Stanford) and many of the faculty and graduate students who have participated in the INFORMS and NSF/CMU workshops on Computational Mathematical Organization Theory and Carnegie Mellon University’s CASOS summer institute (http://www.casos.cs.cmu.edu/).

Funding for this project is provided by the National Science Foundation, Computational Social Systems Program, Award IIS-0084508.

Have fun.

MJ Prietula
Professor
Decision and Information Analysis
Emory University

date: prietula@bus.emory.edu
website: www.bus.emory.edu/prietula
Installing TrustMe Software

The TrustMe 1.0 software is written in Visual Basic 6 for Windows. To install it, simply copy the TrustMe folder to your PC.

Although TrustMe 1.0 is a delivered as an executable, Microsoft’s Visual Basic 6 has an interpretive component to it. What this means is that there could be some software you may need to have in order to properly run the program.

The easiest way to ensure the program running (if it indeed fails) is to install Microsoft’s newest service pack (as of the indicated release date below) for Visual Basic 6 runtime programs:

```
VBRun60sp5.exe Release Date: February 27, 2001
```

VBRun60sp5.exe is a self-extracting executable file that installs the latest versions of the Microsoft Visual Basic run-time files required by all applications created with Visual Basic 6.0. This is available from Microsoft’s download center:

http://support.microsoft.com/

There you simply enter the correct information in the upper left search box:

The files loaded include the fixes shipped with Visual Studio 6.0 Service Pack 5:

```
<table>
<thead>
<tr>
<th>File Name</th>
<th>Version</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acropack.dll</td>
<td>4.71.1013.0</td>
<td>73.2 KB (74,660 bytes)</td>
</tr>
<tr>
<td>Vbrun60.inf</td>
<td>Not Applicable</td>
<td>1.04 KB (1,069 bytes)</td>
</tr>
<tr>
<td>W98inf16.dll</td>
<td>4.71.704.0</td>
<td>2.21 KB (2,272 bytes)</td>
</tr>
<tr>
<td>W98inf32.dll</td>
<td>4.71.15.0</td>
<td>4.5 KB (4,608 bytes)</td>
</tr>
</tbody>
</table>
```

…as well as these supporting files for the VBRun60sp5.exe self-extracting file:

When you start TrustMe, it displays a panel (see Appendix A).
As the TrustMe panel display is rather large, you may also have to adjust the resolution of your screen (up) so that you can view the entire interface panel. This is accomplished by...

1. Clicking on the Display icon in your control panel (Start ➔ Settings ➔ Control Panel). This will show the Display Properties box (as below, for Windows98).


3. You can manipulate both the color settings (Colors) and the screen resolution (Screen area) until an acceptable setting is achieved.
Quick Guide

Note that there are many default values applying to Step 3 and beyond.

For all conditions…

1. Specify the Trust model(s), the associated Honesty model(s), and the Number of Agents for each Trust Model….or simply select the Trust Model(s) and press the [Randomize] button and go to Step 3…
2. Specify how many Tasks per Agent…consider using the Balance mechanism.
3. Specify the Memory Limits, Memory Decay (if any), and task Search Limits for the Agents. (These selections are applied to all agents in all Trust Models)
4. Define Task Turbulence (if any).
5. Specify if you wish to have agent Communication (advice, gossip). If not, go to Step 11.

Set Advice options…

6. Select how to handle advice…
   a. Take the first advice acceptable (Restrict Advice Memory), or Select an Advice Resolution Strategy to handle multiple advice.
   b. Determine if advice not take should be ignored or if the agents should Assess Ignored Advice to see if it was bad advice.

Set up Gossip options…

7. Select how to handle Gossip Assertion (conditions under which gossip will be generated).
8. Select how to handle Gossip Belief (conditions under which gossip will be believed) perhaps specifying, if necessary, an explicit Gossip Count.

Set up Traces…

10. Trace Agent events with communication: (Communication)
   a. Trace Gossip (if any)
   b. Trace Messages (posted by Agents)
   c. Trace Agents (goals, results, steps)
   d. Trace Trust (changes in trust states)
11. Trace Agent Goals and Moves without communication -- Trace Moves.
12. Write all Traces to trustlog_iii.txt file, not Trace Window (for big traces): Logged

Set up automatic Replications of the simulation…

13. Repeat with initial item/agent locations randomly assigned: Replications

Set up output Data file (different from trustlog_iii.txt file – for subsequent data analysis)…

14. Define a new data file (Create Data File)
   a. Place data names (text) in first row of new data file (Add Column Titles)
15. Add to an existing data TrustMe data file (Append Data File)

16. Generate Trust Matrices for the new data file (uses new data file name, cannot use with Append…).elect to also export these as UCINET or NETMINER files.
   a. Generate Detailed Data trust matrices
TrustMe:
Trust, Advice and Gossip

This is a manual to accompany the pedagogical software, TrustMe 1.0: A Social Simulation of Trust, Advice and Gossip. Underlying this simulation is a simple model of what we call “TAG models” of social interactions.1 TAG models of social interaction are those that involve information dissemination within the context of task performance (advice), information dissemination regarding agent behavior with respect to advice (gossip), and submodels of believability of information from those dissemination sources (trust) – Trust, Advice, and Gossip. The clearest examples of such models arise in chat-rooms and posting architectures of various forms and functions, such as those found associated with organizations in eBay or Amazon; those arising as independent sources of collective opinion, such as Epinions; a wide array of professional advice forums, such as codeguru or javaworld; and the immense spectrum of IRC-based chatrooms, ranging from the mundane to nefarious exchanges for hacking information, malware, warez or identities. TrustMe software is useful in discussing and exploring this simple model, but also in using TrustMe as a “data generator” for learning about social network analysis metrics and software, as it generates files readable for both Steve Borgatti’s UCINET and Cyram’s Netminer.

Simplicity is a relative evaluation based on the number of perceived concepts, constructs and interactions of the model.

Representativeness is a relative evaluation based on “how much” of the real world (or, “which important elements”) is modeled.

Many social contexts are based on simple interactions; furthermore, the nature of a “social context” is being broadened such that disparate communication links, such as email and chat rooms, are considered “social.”2 The important thing to keep in mind is that the interesting elements of the investigations you can make with TrustMe are based less on the complexity of those engaging in the social interaction, and more on how those interactions evolve over time. TrustMe weaves simplicity and representativeness. Let us look at the type of TAG tasks, agents, and interactions supported by TrustMe.

Details of the options available in TrustMe as defined by the interface are explained in Appendix A and the output details (files, traces, and data) are described in Appendix B.

1 See Prietula and Carley (2002a, 2005).

2 Reeves and Nass (1998) provide evidence that under certain, but quite broad, circumstances even interactions between computers and humans constitute social interactions based on how humans react to those interactions.
Although the fundamental TrustMe model is intuitively appealing, TrustMe has substantial theoretical roots that contribute individually and collectively to the core formulation. TrustMe focuses on understanding the relationship among humans, agents, tasks, and the social situations in which they are engaged. From this, we seek to establish the elemental basis of social behavior and group phenomena, and make predictions about them. Our guide for this effort is the Model Social Agent matrix.

The Model Social Agent matrix (Carley & Newell, 1994) provides a two-dimensional categorization scheme that specifies the kind of knowledge required by the agent(s) (in terms of increasingly complex social situations), and the kind of information processing capabilities required by the agent(s) to operate with that knowledge, in order to exhibit various kinds of individual and collective phenomena. The scheme is summarized in the following figure.

Agent information processing capability range from an agent that is omnipotent as Laplace’s demon (i.e., can process all knowledge relevant to the specific situation), to an agent that is both cognitively and emotionally defined (and thus constrained). The categories for this dimension are based on general differing critical assumptions of agent models and would certainly allow subtypes and variations. All agents have the capability to define and alter goals, and interact with other agents and objects. What differs is their capability to exploit the knowledge available within the processing (and sometimes emotional) constraints available. As agents’ processing capacity diminishes, different, and perhaps more complex, behavior emerges.

TrustMe focuses on the types of situations and agents indicated by the grey circle: multiple boundedly-rational agents doing real-time tasks with no explicit social structure.

Trust

Trust has a host of definitions and interpretations ranging from colloquial and informal, to highly specific and formal. The concept of “trust” has been applied to individuals, groups, institutions, abstract systems, markets, societies and nations. Trust has been interpreted as a psychological construct, a psychological trait, a collective attribute, a lubricant (social, economic, governmental), a regulatory mechanism and a public good (e.g., Kramer & Tyler, 1996; Misztal, 1996; Schneider, 1999).

---

3 This section is adapted from several sources (Carley & Newell, 1994; Carley & Prietula, 1994; Prietula, 2002; Prietula & Carley 2002a, b).
In TrustMe, the concept of trust is viewed as an internal construct of each agent that underlies certain decisions involving cooperation and, in fact, even deception. These constructs are not “hot” cognition constructs (i.e., involving emotions), but rather simple “cold” cognition models that reflect decision states based on a) external events and b) how those events are interpreted by the particular internal trust construct.

For TrustMe, trust is defined by a set of alternative decision models that respond to events (messages and experience) with differing levels of tolerance and forgiveness. The trust construct impacts behavioral choices (e.g., to provide advice) as well as interpretations of others’ behaviors (e.g., to believe advice).

Essentially, trust is an algorithm that defines how task-specific events influence task-specific behaviors of these agents.

**Advice**

As noted, the coin of the realm in TrustMe is *advice*. With the development of email standards and access to – and growth of – the commodity Internet, organization and societal members have experienced substantial increases in connectivity. Furthermore, the growth of rapid communication is greatly facilitated by the Internet and linkages expand well beyond the artificial or geographical boundaries of the organization, group, or coalition. This permits diverse sets of participants to engage in information exchanges and, depending on the particular context, commerce. Consequently, social behavior does occur as many forms of commerce-based coalitions arise opportunistically on the Internet, where advice from strangers is often sought.

For example, Constant, Sproull and Kiesler (1997) studies the advice patterns via e-mail of “weak ties” (strangers or mild acquaintances) and found that most of the information providers (responding to posted requests) did not know the person who posted the message. Cooperation, as social phenomena of optional advice provision, existed via electronic mail.4

Advice in TrustMe are posted messages bout the task and about the participants.

**Gossip**

In TrustMe, the agents represent collections of individuals who are gathered (virtually) in a common (virtual) place for a common purpose. Participation in these groups may be ephemeral (perhaps task- or event- based) and formal membership or extensive historical-social ties do not exist, as the set of participants may be complete (and even anonymous) strangers. What is found in these groups is that strangers exchange information (experience and rumors) about the quality of advice from other strangers – they gossip.

The distinction between gossip and rumor is difficult at best in complex situations. For example, Koenig’s (1985) study of commercial rumors considers them related, with gossip as “communication about people known to the persons involved in the communicating” (p. 2), and a form of social control, while rumor is a “story or report current within any known authority for its truth” (p. 2), thus the difference is less of kind and more of context or extent. Little research has been done in the context of the Internet, though rumor/gossip are known phenomena that have resulted in lawsuits and

---

4 Krackhardt and Hanson (1993) provide an insightful piece on the difficulty and importance of informal networks in an organization.
international incidents. In TrustMe, the context is simple so rumor and gossip will be considered equivalent.

Gossip is taken to be information conveyed about the quality of advice from a specified source.

**Induced Simplicity Hypothesis**
This hypothesis proposes that, for many social and organizational settings, much of the available set of decisions (say, the problem space for the task) is relatively restricted and this simplicity is induced by a confluence of the task, the situation, and the individual (Prietula & Carley, 2006).

These three factors act as constraints that often severely restrict the behavioral options of the individual, such that models of individuals behaving in those contexts can be sufficiently representative to account for parameters underlying most of the variance. Furthermore, because of the plasticity of the adaptive nature of individuals, much of their behavior can be explained in terms of task and situational components; conversely, examining the nature of how humans adapt to tasks and situations can yield insight into the specific influences of those factors. Behaviors are barometers to the tasks and situation; tasks and situations are drivers of behavioral adaptations. Tasks and situations define “what kind of behaviors” a human may engage or define; defined behaviors reflect demands of the tasks and situations.

**Modeling as Method**
Computational modeling is a relatively recent (as methods go) methodology. In fact, computational modeling is more and more being defined as the “third branch” of various sciences (i.e., theoretical, applied, computational).\(^5\)

Despite the innovative and influential early work of such authors as Cyert and March (1963) and Cohen, March and Olsen (1972), the use of computational modeling in the social sciences has been less popular, with few departmental (e.g., organizational behavior, management, sociology, social psychology, psychology, economics) requirements or courses affording substantial instructional and research support for the approach, either at the undergraduate or graduate levels.

**TrustMe Problems**
TrustMe involves a set of virtual agents exchanging information and performing an abstract problem – searching for information objects scattered around their world. A problem is comprised of one or more objects to be found by the agents. Each assigned object to be located is called a task of the problem.

Problems are made up of tasks to complete. The world of TrustMe agents is comprised of a 12 x 12 grid of “locations,” where objects and agents are collocated. This world can be viewed as a set of physical locations and objects (e.g., a warehouse, a neighborhood), as a set of virtual locations (e.g., a WWW URL space), or as a more abstract representation of concepts, distributed architectures, markets, or tasks.

Consider three simple examples.

Example 1 – Physical Locations
A large retail chain that sells compact discs is faced with a recurring item-location problem in response to customer requests. The computer shows that the item is in stock, but the item itself is not located where the computer says it should be shelved. This results in physical search for the item in store locations, such as a backend staging warehouse that is often unorganized and fraught with turnover. They are also considering the implications of worker-to-worker instant communication to help.

Example 2 – Virtual Locations
An organization has set up a voluntary chat room that is visited by employees to exchange information where items of interest can be found in the electronic archives (e.g., URLs) on a particular topic. How does reputation and trust evolve and how is it maintained? How difficult is it to destroy trust in this type of environment?

Example 3 – Virtual Locations2
On the internet, various types of chat rooms exist for various topics. Assume that individuals are seeking the specific Internet Relay Chat (IRC) channel locations for specific topics. Thus, “topic i” will be found on “IRC k” where k can be an particular path:port or channel location.

TrustMe presents its output as Example 3…chat rooms. The scenario is simple. There is a primary chatroom (sometimes called a scatter, or router, room), set up by an organization, that acts as a “meta chat room” that is used to discuss and direct topics that occur in “channels” identified by # (e.g., in #5 they talk about java graphics, in #32 they talk about VB2005, in #21 they talk about Oracle DB 10g).

A trace of the logfile for the meta-chat room would look like this:

```
------------------starting log file output------------------
...
M-Q---->Agent 16: Where is Topic  61?
M-Q---->Agent 7: Where is Topic  25?
MA---->Agent: 7 answering Agent 16
M Agent: 7 says Topic  61 is in #3
M-Q---->Agent 30: Where is Topic  117?
M-Q---->Agent 24: Where is Topic  93?
MA---->Agent: 24 answering Agent 23
M    Agent: 24 says Topic  89 is in #93
...
M------>Agent 16 reviewed answer from Agent 7
M    Agent 16 is taking ADVICE from 7
M    Agent 16 is moving to #3 for Topic  61
...
JC---->Agent: 16 JOINING # 3
M-F------------->Agent: 16 found Topic  61 with GOOD advice from Agent 7
```

TrustMe Metrics
The key component of how TrustMe conceptualizes a problem is that there is a concept of “distance between locations” and that agents’ choices are defined (operationally) in terms of these distances. Distance can be interpreted as a traditional physical space, or on a more abstract level, such as time or similarity.
Regardless of the particular interpretation, the general effect is that the problem is generally characterized as search, and search has an associated effort defined as a function of distance.\(^6\)

In TrustMe, there are three interpretations of distance and these are used as surrogates for three metrics of effort:

<table>
<thead>
<tr>
<th>Effort Metric</th>
<th>Distance</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort Metric 1</td>
<td>Count</td>
<td>Views world locations as being a constant distance apart, so effort is a function of the number of locations visited.</td>
</tr>
<tr>
<td>Effort Metric 2</td>
<td>Euclidean</td>
<td>Views world locations as a “traditional” physical space with effort as a function of distance traveled in the search.</td>
</tr>
<tr>
<td>Effort Metric 3</td>
<td>Logistic</td>
<td>Views world location search as a growth model metaphor based on the number of locations searched, where shorter task visits are rewarded and longer task visits are penalized.</td>
</tr>
</tbody>
</table>

**Effort Metric 1**

In this perspective effort is viewed as logical effort, where the distance (as a logical step) between any two locations is defined as a single unit. Therefore, the effort for any given agent is simply the integer count of the number of steps taken over the task searches. This does not presume that NO effort is required, simply that the effort is a constant and a linear function of the steps taken.

If we imagine this perspective as a problem in searching the Web, the interpretation would be that the URL decisions (and typing) required to go from between arbitrary sites is essentially equivalent (and trivial). If we imagine that each step is a deliberation effort of a cognitive problem, the interpretation would be that the number of “deliberation steps” for each task measures the problem difficulty or complexity.

Although calculated basically as an agent-level property, it is easy to see how this can be used to calculate group/organizational level values. The sum of all agents’ effort is the total organizational effort,

\[
E_{ijk} = \sum_{i=1}^{Agents} \sum_{j=1}^{Tasks} \sum_{k=1}^{Steps} E_{ijk}
\]

where the maximum effort across agents (for each task) is a good surrogate for the time it takes to solve the overall problem, as the agents are presumed to work in parallel.

---

\(^6\) Search has a long and venerable history as both a distinct problem characterization as well as a metaphor in both cognitive science (e.g., Newell & Simon, 1972) and computer science (e.g., Knuth, 1998).
**Effort Metric 2**

In this perspective effort is viewed as search for objects in a physical location, so effort is considered as *physical* effort based on the distances covered in the search.

The effort performed by an agent on a given step is defined as the Euclidean distance in 2-space:

$$E_i = \text{distance}(\text{location}_0, \text{location}_1)$$

where the distance function is articulated for normal rectangular coordinates,

$$E_i = \sqrt{(x_1-x_0)^2 + (y_1-y_0)^2}$$

Therefore, the effort for any given agent $i$ can be defined as the simple sum of the steps $j$ required for each of the tasks $k$ in the problem.

$$E_i = \sum_{k=1}^{\text{Tasks}} \sum_{j=1}^{\text{Steps}_k} E_{ijk}$$

**Effort Metric 3**

In this perspective effort is seen as a variation of Effort Metric 1. Specifically, it is defined in terms of a logistic growth model, where effort in the “low end” is reward by undercounting the actual “steps” and tasks that take longer at the “high end” are penalized by over counting, thus giving it the familiar s-shape.

The equation used to calculate an agent’s effort for a given task is the following:

$$E_i = \left( \frac{S_{\text{max}}}{1 + e^{-a(S_{\text{mid}} - S_{\text{max}})}} \right)$$

where $S_{\text{max}}$ is the upper bound values (respectively) for search effort on a given task, $S_{\text{mid}}$ is the inflection point at which the effort curve begins to “flatten” out, $S_{k-1}$ is the effort value on the prior search step, and $a$ is a constant that describes characteristic duration (i.e., the time [steps] between which the effort curve is at 10% and 90% of its maximum value). $a$ is calculated as $\ln(81)/4S_{\text{max}}$

For TrustMe, $S_{\text{max}}$ is set at the maximum search effort allowed as defined by the user via the TrustMe interface, and it is the only parameter under the control of the user for Effort Metric 3.

The nature of the different performances of Effort Metrics 1 and 3 can be demonstrated in the following graphs. In Figure 1, both metrics are graphed with the $S_{\text{max}}$ set at the maximum allowed (and the default value) of 144. That is, each agent devote 144 search “resources” (i.e., steps) toward each task of finding a particular object. If the task is not completed after 144 steps, the agent quits the task and begins the next search task (assuming there is another one).
Figure 1
Maximum Search Steps Allowed per Task (Set by User)

Figure 2
Maximum Search Steps Allowed per Task (Set by User)
As can be seen, if an agent completes a task with steps \(< S_{\text{mid}}\) it uses “less effort” under Effort Metric 3 than Effort Metric 1. Conversely, when an agent takes longer than \(S_{\text{mid}}\) to complete a task, then it is penalized under Effort Metric 3 (relative to Effort Metric 1).

When an agent’s search limit \((S_{\text{max}})\) is reduced, the resulting Effort Metric 3 is “squashed” accordingly, as shown in Figure 2. Note that there are no individual values for the function beyond the specified \(S_{\text{max}}\) as the agent quits when that limit is reached.

**TrustMe Agents**

As noted, agents in TrustMe solve problems by doing tasks. However, how they do their tasks, as well as how well they do them, is strongly influenced by the bounded nature of their rationality. Recall that they are not omniscient agents, but restricted within the context of the task in very specific ways.

**Bounded Rationality**

Each agent is *boundedly rational*, that is, each agent is architecturally constrained in primary ways in the specific context of the task: informationally and computationally. The specific ways in which these agents are constrained are specified by the agents’ micro-architecture. These two types of bounds on rationality are realized in several ways in a typical TrustMe agent – and several of these bounds can be directly manipulated at the user interface (see Figures 4 and 5).

**Limited Search**

Agents have a limit of (search) effort -- how many steps (locations, deliberation cycles) they will engage to solve a task. You can manipulate this limit. When this limit is reached, the agent ceases to address the current task (i.e., object to locate) and proceeds to address a new task. The default limit of search is set equal to the search space of the problem (144).

---

7 The strong form of bounded rationality was originally conceived to characterize the effects of a restricted rational agent on the assumptions and conclusions of economic and administrative theory (Simon, 1976), and was highly influenced by research from sociology, social-psychology, and cognitive psychology. Bounded rationality ranges from strong forms that hypothesize the underlying cognitive (e.g., Newell, 1990) or institutional (Cyert & March, 1963) apparatus to less restrictive derivatives that address various organizational and economic issues at micro and macro levels.

8 A micro-architecture is a description the components of the agent’s deliberation mechanisms that contribute to the behaviors under study. Micro-architectures differ from full architectures, such as Soar (Newell, 1990) in that they are not theories of full intelligent deliberation, but reflect on some level the dominant elements of interest that plausibly impact task behaviors. Thus, TrustMe does not propose to be a generally intelligent architecture, but a model of a form of architecture that accounts for the behaviors under examination. They specify the assumed components of consequence for the task.

9 The user interface is the primary display panel explained in Appendix A.
Limited Memory

There are two aspects of limited memory for these agents: locations and errors in recall.

Location Memory

Agents have a limited memory of locations. Each time an agent visits/considers a location, it stores the contents of that location in memory. This pair of location and object is called a chunk. However, there is a limit to how many of these chunks an agent can recall. You can manipulate that limit. It is applied as a FIFO – first in, first out – aged knowledge queue.

Fallible Memory

Agents may encounter errors in recall of memory locations. Not only do agents have a limited number of chunks they can recall, but they also may encounter errors in recall of those chunks. You can manipulate whether the agents do have recall errors, and if they do, you can manipulate the nature of the error via decay according to a time-based trace degradation parameter.

The decay is a simple function of the relative length of time (age) a particular memory chunk (i.e., object-location pairing) has been resident in memory, where the decay is expressed as a likelihood of accurate recall:

\[
\text{Likelihood of accurate recall} = \text{Age}^{\log_\beta}
\]

The effect of this function can be seen in Figure 3.

A chunk in this sense is a bond established between two symbolic knowledge references – location and object. The concept of a chunk in a full architecture can be found, for example, in Soar (Newell, 1990).
In the actual user control on the panel (explained in Appendix A), the slider values are from 0 (no degradation) to 10 (maximum degradation) and result in changes in $\beta$ (Figure 3) in the following manner:

<table>
<thead>
<tr>
<th>Slider Value</th>
<th>Decay ($\beta$ value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0 (no decay)</td>
</tr>
<tr>
<td>1</td>
<td>.9</td>
</tr>
<tr>
<td>2</td>
<td>.8</td>
</tr>
<tr>
<td>3</td>
<td>.7</td>
</tr>
<tr>
<td>4</td>
<td>.6</td>
</tr>
<tr>
<td>5</td>
<td>.5</td>
</tr>
<tr>
<td>6</td>
<td>.4</td>
</tr>
<tr>
<td>7</td>
<td>.3</td>
</tr>
<tr>
<td>8</td>
<td>.2</td>
</tr>
<tr>
<td>9</td>
<td>.1</td>
</tr>
<tr>
<td>10</td>
<td>.05 (maximum decay)</td>
</tr>
</tbody>
</table>

### Limited Attention

**Advice**

Agents have a **limited attentional capacity for advice**. When an agent requests advice, it has a restricted capacity to process all of the information sent to it and this limit is expressed in terms of an integer limit of sources it can consider. This limit is the “memory of locations” parameter previously described, and is under your control.

- You can restrict the capacity to the *first answer* received or use the limit of the memory selected (see Figure 4). There is no error in recall/consideration trace in this capacity, as it is a singular event that is processed at one time (i.e., all answers are immediately reviewed and a decision is made).

- There is a set of heuristics for handling *multiple sources* of information. This is described as the **Advice Resolution Strategy** (see Figure 5) and is discussed in the next section on TrustMe Interactions.

### The Task Environment

Agents have a **limited view of their world**. Agents can only perceive what a location contains if it goes to (considers) that location. For a physical location analogy, think of this as having to go to a cabinet or a drawer and having to open it to see the contents. For an Internet analogy, think of this limit as having to go to the website to determine its contents. For an abstract analogy, think of it as having a particular focus of attention. You have no control over this constraint.
Limited Memory of Locations (searched) for each Agent.

Limited Memory of Advice to Consider from other Agents

Restrict Advice Memory to First Advice Received.

Figure 4

Error in Recall

Limit of (search) Effort for each Agent.

Defines which model to use to resolve multiple sources of advice.

Figure 5

TrustMe Interactions

Agents in TrustMe can do their tasks non-cooperatively. Specifically, you can define some set of $N$ agents, where $1 \leq N \leq 144$, such that these $N$ agents are doing the task in parallel and not interacting which, in TrustMe, means no communication. However, this is not realizing a TAG model.

Role of Communication

TAG models in TrustMe require, by definition, communicative interaction among agents. That is, agents have to communicate advice and gossip. As you will see, this is where things get a little complicated, as the simple TAG model actually embodies several constructs of consequence.

In TrustMe, there is no explicit concept of well-defined subgroups such as coalitions, friendship groups or alliances. These concepts are handled by another program (called TrustUs©) and are not required for the type of interactions addressed by TrustMe. The subgroups defined by TrustMe are solely those associations defined by trust based on communication events. TrustMe agents know each other solely by their behaviors (advice) and judgments are made based on those behaviors, directly or indirectly. In a sense, these agents are forming an “information structure” or “advice network” that may grow, shrink, or change depending on the parameters of the task, the agents, and the environment.

---

11 In TrustMe, agents do not “interfere” physically with one another, such as queuing up at locations causing wait conditions. Remember, these are viewed as chat room exchanges and searches.

12 This is not an arbitrary distinction, but based on a theoretical model called the Social Agent Matrix (Carley & Newell, 1994). The Social Agent matrix provides a two-dimensional categorization scheme that specifies the kind of knowledge required by the agent(s) (in terms of increasingly complex social situations), and the kind of information processing capabilities required by the agent(s) to operate with that knowledge, in order to exhibit various kinds of individual and collective phenomena.
Selecting Trust Models

In order to define how many agents and how many items there are, you must select from a set of Trust Models (explained later), define how many agents for each model you wish, and how many items for each agent there will be. Consider Figure 6, which depicts the “default” settings for Agent Models used by TrustMe when you begin:

TrustMe is designed to facilitate such selections and to ensure a few simple constraints:

1. No more than 144 Agents can be defined in total.
2. No more than 144 Task requests can be defined per Agent.
3. Only integral and equivalent assignments will be made to Agents (i.e., each agent will have the same number of tasks assigned for each problem).

To facilitate adherence to these constraints, you should first consider selecting the Trust Models you wish to include (without specifying the number of agents), then clicking on the Balance button in the lower right hand corner (Figure 7). TrustMe will automatically assign the number of tasks and the number of agents for each Trust Model type selected within the allowable constraints. The Balance button will only be selectable if more than one Trust Model is selected.

For example, if you select Trusting and Distrusting Trust Models, then hit Balance, the resulting display should look like the one depicted in Figure 7.
The two Trust Models are maxed out at 10 agents each…

Total Items are within the 144 Task limit…20 Agents total.

Each agent is automatically assigned 7 Tasks to find (all unique).

Check the Trust Models you want, then hit the Balance button to automatically calculate the Number of Agents and Tasks per Agent…

Total Items are within the 144 Task limit…20 Agents total.

Note: You can also simply check the Agent Models you wish to use, and then click the Randomize button. This will automatically set (randomly) the specific model parameters (e.g., type of Honesty model for each agent type) and the number of agents.

To run the TrustMe simulation, simply click on the Initialize button in the bottom right of the display (lower output panel). This will do some checks on your selections (though, there should be no problem at this point). The lower output panel should look something like Figure 8.
The upper output panel, however, will display the objects to be found (in their 12 x 12 grid of locations) along with the agents. Each agent’s Trust Model is indicated by the color of the Agent icon (see Figure 9).

Once you click **Start**, the simulation will begin. To alter settings, click on the **Reset** button. When the simulation has ended, the upper output panel will look something like Figure 10.
Each agent is depicted explicitly and configured as edges, with all possible trust links defined between the depicted agents. The particular type of agent (i.e., its Trust Model) is defined by its color.\textsuperscript{13}

Links define the nature of the trust relationships.

- **green** links are symmetric trust dyads (Agent \(i\) trusts Agent \(j\) and Agent \(j\) trusts Agent \(i\));
- **blue** links indicate asymmetric trust links (Agent \(i\) trusts Agent \(j\) and Agent \(j\) does not trust Agent \(i\)); and,
- **red** links indicate symmetric distrust dyads (Agent \(i\) does not trust Agent \(j\) and Agent \(j\) does not trust Agent \(i\)).

In the lower output panel, a list of summary data and computations give the results of the simulation. The output and the output mechanisms are defined in detail in Appendix B.

Let us consider advice, trust, and gossip in a little more detail.

**Advice**

The form of the TAG model in TrustMe, as was noted, is based on a question-answer framework characterized as a public forum, such as an organizational chat room, where one agent asks if any other agent knows where some topic is located (in a channel). When an agent is to search for a particular topic, it first checks its own memory. If it does not recall if it has seen where the topic was discussed, then it will post a question to the group. Although an agent prefers its own memory to advice from other agents, advice is the cornerstone to this model of group behavior.

**Chat Room Messaging**

As noted, communication is posted to a globally accessible communication structure in a broadcast form – a meta-chat room (Figure 11). Thus, all agents can see the posted question, where Agent 16 posts: “Where is Topic 61?”

\begin{figure}[h]
\centering
\scalebox{0.8}{
\begin{tikzpicture}
\node[draw, circle] (A20) at (0,0) {A20};
\node[draw, circle] (A4) at (1,0) {A4};
\node[draw, circle] (A23) at (2,0) {A23};
\node[draw, circle] (A16) at (3,0) {A16};
\node[draw, circle] (M-Q-20) at (0,-1) {M-Q---->Agent 20: Where is Topic 77?};
\node[draw, circle] (M-Q-4) at (1,-1) {M-Q---->Agent 4: Where is Topic 13?};
\node[draw, circle] (M-Q-23) at (2,-1) {M-Q---->Agent 23: Where is Topic 89?};
\node[draw, circle] (M-Q-16) at (3,-1) {M-Q---->Agent 16: Where is Topic 61?};
\end{tikzpicture}
}\caption{Figure 11}
\end{figure}

\textsuperscript{13} Output files explicitly identify the classes of agents and do not rely on color. The color is simply a visual guide for the immediately displayed output on the screen.
Agents always check the chat room for questions they may be able to answer. If an agent knows the answer (based on its memory contents and error conditions), it will (under basic conditions) post an answer (assuming it has not done so before). Each agent can post one answer for each question and, once posted, it cannot retract or alter that answer.

Answers to these questions are also posted for all agents to see (Figure 12). An answer posted by Agent 7 to the question posed by Agent 16 would look like this:

```
M-Q---->Agent 20: Where is Topic 77?
M-Q---->Agent 4: Where is Topic 13?
M-Q---->Agent 23: Where is Topic 89?
M-Q---->Agent 16: Where is Topic 61?
MA---->Agent: 7 answering Agent 16
M Agent: 7 says Topic 61 is in #3
```

Because agents continually monitor the posting (unless the are executing a specific search driven by specific knowledge), agent A16 will notice the answer posted by A7 and incorporate that knowledge into its search strategy. Other things equal, if this is the only post about agent A16’s particular task (finding a chat room channel covering Topic 61) and that task is not yet achieved, A7 will go to channel #3 to see if it covers Topic 61, relevant to the task it wants to achieve.

If no answers are posted, the asking agent (e.g., agent A16) will engage a default strategy of a random search of channels.

- If more than one agent answers and those answers are different, then the asking agent has to determine which one to choose (Figure 13).

```
M-Q---->Agent 20: Where is Topic 77?
M-Q---->Agent 4: Where is Topic 13?
M-Q---->Agent 23: Where is Topic 89?
M-Q---->Agent 16: Where is Topic 61?
MA---->Agent: 7 answering Agent 16
MA---->Agent: 9 answering Agent 16
M Agent: 7 says Topic 61 is in #3
MA---->Agent: 9 answering Agent 16
M Agent: 9 says Topic 61 is in #22
MA---->Agent: 2 answering Agent 16
M Agent: 2 says Topic 61 is in #67
```

```
Answer Answer Answer (Information)
A7 A9 A2 A16
```
Advice Resolution Strategies

In TrustMe, there are heuristic decision procedures to address what are called multiple advice resolution problems, which are addressed by the Advice Resolution Strategy selection (recall prior Figure 5). This presumes that there are $N > 1$ sources of advice to resolve. In fact, you can restrict advice to take the first acceptable advice encountered by checking the appropriate check box:

If you wish to endow the agents with the capacity to consider multiple sources of advice, then there are four specific strategies that can be chosen.

**Strategy 1: Random**

*Random* – an answer is selected randomly from the set.

**Strategy 2: Maximum Number**

*Maximum* – an answer selected is based on the most frequently occurring suggestion from the set.

**Strategy 3: Random from Trusted**

*Random, Trusted* – an answer is selected randomly from the set of advice from trusted agents only.

**Strategy 4: Maximum from Trusted**

*Maximum, Trusted* – an answer is selected based on the most frequently occurring suggestion from trusted agents only.

You can set which one to incorporate for the agents. The result is that multiple responses must be pruned down to a single solution that can inform the agent’s moves. Table 2 illustrates the set of models that define the advice resolution strategy selection alternatives.

<table>
<thead>
<tr>
<th>Advice Resolution Strategy</th>
<th>DO NOT USE TRUST MODEL</th>
<th>USE TRUST MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANDOM SELECTION</td>
<td>Model 1</td>
<td>Model 3 (default)</td>
</tr>
<tr>
<td>MAXIMUM VOTES</td>
<td>Model 2</td>
<td>Model 4</td>
</tr>
</tbody>
</table>
Model 1 or 2 could be selected (we have not yet discussed the Trust Models) to yield the decision without regard to any trust-based knowledge.

Recall that as agents move from channel to channel, they build up a memory of their visits in the form of chunks, where a chunk is defined as: \(<\text{object: item}>\). As we have noted, the micro-architecture of these agents permits both how many chunks can be retained as well as the veracity of the retained memory (in terms of recall errors as a function of time, see prior Figure 4).

**Example: Advice and Trust**

To illustrate the role and function of advice, we can look at a simple example that asks two basic questions.

**Example.** In this model, how does the number of agents working on the problem impact the group performance? And how does communication (as advice) impact group performance?

The solution is found by running TrustMe with the following settings defined in Figure 14 (and accepting all other defaults):

a. Define one Trusting agent (Step 1).

b. Define 144 objects to be found (Step 2).

c. Disallow any communication (Step 3).

d. Show Details of the run (Step 4).

e.

![Diagram of TrustMe settings](image)

This defines a single agent with 144 task assignments…
When you press **Initialize**, the upper panel will display 144 objects and one (trusting) agent. As no objects can be located at the same location, what you will see is a 12 x 12 matrix of 144 black circles. The single agent will be shown as a green circle at one of the 144 locations (see Figure 15).

When you run this configuration (click **Start**), you will see a summary listing in the dialogue box in the lower right of the panel. Though your results will likely differ (because of random elements in the problem), they should look like this (scroll through the box and find these terms):

**Effort Metrics.**

- **Ave Agent Effort1 (per Problem):** 11,189.
- **Ave Agent Effort1 (per Task):** 77.7
- **Max Agent Task Effort1:** 11,189.
- **Min Agent Task Effort1:** 11,189.
- **Std Dev in Tasks, Effort1:**

- **Ave Agent Effort2 (per Problem):** 27,653,641.
- **Ave Agent Effort2 (per Task):** 192,839.17
- **Max Agent Task Effort2:** 27,653,641.
- **Min Agent Task Effort2:** 27,653,641.
- **Std Dev in Tasks, Effort2:**

- **Ave Agent Effort3 (per Problem):** 10,492.
- **Ave Agent Effort3 (per Task):** 72.86
- **Max Agent Task Effort3:** 10,492
- **Min Agent Task Effort3:** 10,492
- **Std Dev in Tasks, Effort3:**

**Total Tasks Quit by Agents:** 37

Focusing on the Effort1 metric, one agent took 11,189 search steps to complete 144 tasks, with an average of 77.7 complete each one. Because there is only one agent, this is also the “time” it took to complete the entire problem.
However, there is a question as to whether the problem is actually “solved”, as only 107 of the 144 objects were found! In searching for 37 of the objects, the agent had to quit as the search limit (of 144) was exceeded. In a sense, one agent solved approximately 74.3% of the problem.

Let us do the same task, except making the following changes:

a. Define two trusting agents.

b. Define 72 items per agent (split up the problem).

Still allowing no communication, this yields the following results:

<table>
<thead>
<tr>
<th>Problem Completion Cycles (Time): 11,189</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Problem Effort Metrics:</td>
</tr>
<tr>
<td>-- Total Problem Effort1: 11,189</td>
</tr>
<tr>
<td>-- Total Problem Effort2: 27,653,641</td>
</tr>
<tr>
<td>-- Total Problem Effort3: 10,492</td>
</tr>
</tbody>
</table>

Again, let us focus on the Effort1 metric. Expectedly, the average effort an agent expends for the problem has dropped (11, 189 → 6,553) as has the average effort to complete a task (77.7 → 45.51), but also note that the amount of tasks completed has also dropped slightly (60.4% of the tasks completed successfully here versus 74.5% found previously). What has changed radically was the total problem time, dropping from a previous 11, 189 to 7,043.

<table>
<thead>
<tr>
<th>Problem Completion Cycles (Time): 7,043</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Problem Effort Metrics:</td>
</tr>
<tr>
<td>-- Total Problem Effort1: 13,106</td>
</tr>
<tr>
<td>-- Total Problem Effort2: 33,327,382</td>
</tr>
<tr>
<td>-- Total Problem Effort3: 12,186</td>
</tr>
</tbody>
</table>
Problem time is defined as the maximum time any agent takes to complete the task. Therefore, assuming parallel execution by the agents, this task time is defined by the slowest agent. The total organizational effort defined by Effort1 (sum of all agents' Effort1) remains roughly the same (13,106 ≈ 11,189), as the output indicates.

Now, make the final adjustment by allowing communication – check the box you unchecked in the beginning (prior Figure 14). In this case, communication will take the form of posting questions and receiving advice on locations of objects (see prior Figures 12 and 13) for the 2-agent situation. This generates the following results:

<table>
<thead>
<tr>
<th>Effort Metrics.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-- Ave Agent Effort1 (per Problem): 4,269.5</td>
</tr>
<tr>
<td>-- Ave Agent Effort1 (per Task): 29.65</td>
</tr>
<tr>
<td>-- Max Agent Task Effort1: 4,889.</td>
</tr>
<tr>
<td>-- Min Agent Task Effort1: 3,650.</td>
</tr>
<tr>
<td>-- Std Dev in Tasks, Effort1: 876.11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Effort Metrics.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-- Ave Agent Effort2 (per Problem): 10,690,779.5</td>
</tr>
<tr>
<td>-- Ave Agent Effort2 (per Task): 74,241.52</td>
</tr>
<tr>
<td>-- Max Agent Task Effort2: 12,392,817.</td>
</tr>
<tr>
<td>-- Min Agent Task Effort2: 8,988,742.</td>
</tr>
<tr>
<td>-- Std Dev in Tasks, Effort2: 2,407,044.52</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Effort Metrics.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-- Ave Agent Effort3 (per Problem): 3,721.</td>
</tr>
<tr>
<td>-- Ave Agent Effort3 (per Task): 25.84</td>
</tr>
<tr>
<td>-- Max Agent Task Effort3: 4,364</td>
</tr>
<tr>
<td>-- Min Agent Task Effort3: 3,078</td>
</tr>
<tr>
<td>-- Std Dev in Tasks, Effort3: 909</td>
</tr>
</tbody>
</table>

| Total Tasks Quit by Agents: 21 |

The situation has improved in several respects. First, the average agent effort has decreased per problem (6,553 → 4,269.5) and per task (45.51 → 29.65). Second, the percent of the problem achieved (i.e., percent of the tasks completed) has risen (60.4% → 85.4%). Third the total problem (organizational) effort has (necessarily) dropped (13,106 → 8,539). Finally, the problem completion time has improved (7,043 → 4,889).

| Problem Completion Cycles (Time): 4,889 |

<table>
<thead>
<tr>
<th>Total Problem Effort Metrics:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-- Total Problem Effort1: 8,539</td>
</tr>
<tr>
<td>-- Total Problem Effort2: 21,381,559</td>
</tr>
<tr>
<td>-- Total Problem Effort3: 7,442</td>
</tr>
</tbody>
</table>

Consequently, adding advice reduces the total organizational effort, the individual effort (not unrelated, of course), the time it takes for the problem, and increases the percent of tasks completed successfully.

We can view the percent of tasks completed successfully as a measure of organizational effectiveness.

We can view the organizational productivity as the amount of task completion obtained per search resource unit used (i.e., tasks completed divided by effort, say, using the Effort1 metric) and multiply that by 100 to normalize. In the above example, two agents without communication have an organizational productivity of .66 (i.e., about...
2/3 of a task is completed with every 100 search steps). Two agents with communication increase the productivity metric to 1.4.

Exploring the average resources used per task completed, it is also possible to generate an extrapolated total effort by simply multiplying the average per task by the number of total tasks in the problem (144). Consider the results shown in Table 3.

<table>
<thead>
<tr>
<th></th>
<th>Effort per Completed Task</th>
<th>Completed Tasks</th>
<th>Uncompleted Tasks</th>
<th>Projected Total Problem Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Agent</td>
<td>77.7</td>
<td>107</td>
<td>37</td>
<td>11,188.8</td>
</tr>
<tr>
<td>Two Agents</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(no communication)</td>
<td>45.1</td>
<td>87</td>
<td>57</td>
<td>6,494.4</td>
</tr>
<tr>
<td>Two Agents</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(communication)</td>
<td>29.6</td>
<td>123</td>
<td>21</td>
<td>4,262.4</td>
</tr>
</tbody>
</table>

As can be seen from the table, extrapolated effort measures reveal a substantial decrease because of shared knowledge.

Recall that there is a default limit to the agents’ search at 144 steps. A single agent, on the average, uses 53.9% of its resources in solving a task. An agent in a dyadic group uses only 31.3% of its resources and when these agents communicate, only 20.5% of their search resources are consumed per task. Slack is growing.\(^{14}\)

Thus, communication in this task allows advice to increase the search efficiency of the agents (an individual result) and decrease the total organizational effort (an organizational result) while increasing the percent of the task completed (an organizational result). Measures of organizational productivity

Trust

In the previous example, any agent does not care which agent is answering. Furthermore, the simple heuristics to deal with a decision space that involves multiple possible solutions proposed by these agents are applied without regard to agent identities.

Nonetheless, TrustMe is a social simulation. Agents in TrustMe indeed have the capacity to consider the source of advice. The provision and acceptance of advice in TrustMe is viewed as a form of cooperation between agents. Specifically, in the context of

\(^{14}\) In this model, slack of this sort is “personal” slack and not directly organizational slack, except that aggregating personal slack yields an organizational level measurement of personal slack (TrustMe generates this). Slack does not accumulate (beyond a task search event) nor can it be used by the “organization.” In TrustMe, the underlying model is one of a loose confederation of individuals, not specific organizational structures.
this task, cooperation (as advice) should reduce search effort for the agents (in particular) and for the entire group (in general).\(^{15}\)

However, as cooperation -- as the provision of advice -- is a decision variable within TrustMe, so must be non-cooperation -- as a choice not to provide advice. In TrustMe, trust is the construct that drives the decision to cooperate or not. In short, cooperation is viewed as a by-product of trust.

The trust construct is realized as augmented transition networks that move from various trust states to other trust states, based on how the current events impact the current state context (see Appendix C).

In TrustMe, agents do not “know” other agents other than their communication events – posted messages. Furthermore, agents cannot differentiate agents on any parameter other than revealed by the communication events. This reduces the complexity of the situation to one of judging the events themselves, with attributions made concerning the source of many of those events – the agents.\(^{16}\)

Now, what events are defined in TrustMe and what attributions are made?

**Events.** Events are those components of the task environment to which agents attend or that impact the task. The task environment is composed of the task elements themselves (i.e., locations, items) as well as the other agents and the properties by which they are known (social elements).

The primary **task events** defined in TrustMe are based on the success of the item search, while the primary **social events** are based on agent communication.

In TrustMe, the driving factors are based on whether the agent found the particular item or not, and whether the advice provided (if any) was correct or incorrect. Note that a basic task event (e.g., the agent did not find item \(k\) and location \(i\)) can generate a social event (e.g., posting of gossip about agent \(l\)).

**Attributions.** Based on the events in the task environment, an agent makes attributions about other agents in the task environment. In TrustMe, the primary attribution is whether an agent is trustworthy or not. In TrustMe, attributions are all about trust. In TrustMe, attributions of trust are binary (trust, not trust).

The mechanism that defines process of trusting is the agent’s **trust model.** The trust model articulates how a series of events in the task environment determines the attributional states toward the agents in that task environment.

---

\(^{15}\) This presumes that there is no explicit cost for the provision of advice (when known) and no explicit penalty for not providing advice (when known). Transmission is costless, though the effects may not be.

\(^{16}\) In fact, the concept of trust itself can be viewed as an adaptive characteristic to reduce the decision complexity of a complex social environment. Thus, the trust constructs could be interpreted as alternative variations on a social simplification heuristic.
So, the decisions impacted by trust models in TrustMe are those related (directly and indirectly) to cooperative or non-cooperative agent behaviors in the context of the task:

1. The decision to believe (or not believe) advice.
2. The decision to provide (or not provide) advice.
3. The decision to assert (or not assert) gossip.
4. The decision to believe (or not believe) gossip.
5. The decision to provide (or not to provide) deceptive advice.

The first two decisions will be explored in the context of trust models in the next section, while the remaining three will be elaborated upon in the section concerning gossip and deception.

**Trust Models**

TrustMe supports the following five general types of general trust models (recall Figure 6, see also Appendix C). On the screen, these types are depicted by the indicated color.

**Model 1: Trusting.**
These agents do not see events as sufficiently important to generate a response to alter trust behavior. They invariantly view other agents as honest and trust them. (Green)

**Model 2: Distrusting.**
These agents do not trust any other agent under any circumstances. All advice is ignored. These agents can be simply isolated performers or can actively disrupt the information quality of the group, depending on how you configure the options. (Red)

**Model 3: Random Judgment.**
In this model, a judgment of whether to trust the advice from an agent or not is solely based on a random choice from a uniform distribution. As opposed to the other models, this is a memoryless model with respect to social events. Each encounter with an agent’s advice is independent of any other encounter. (Orange)

**Model 4: Forgiving.**
These agents exhibit alteration of their trust in other agents, depending on options and situations. These alterations of trust are based on particular sequence of bad advice they have encountered when accepting the cooperation of other agents. TrustMe has four versions of Forgiving agents, each based on the number of bad advice messages it receives in a row for a particular agent, referred to as the tolerance level: 1, 2, 3 and 4. Once that particular limit is reached (e.g., 2 pieces of bad advice from a specific agent in a row), then a judgment of distrust is made. However a judgment of untrustworthiness is not an absorbing state. Agents that are not trusted can supply a series of good advice and redeem themselves. These are symmetric models, so if it takes \( n \) pieces of bad advice to be judged untrustworthy, then it takes \( n \) pieces of good advice to become fully trusted again. (Forgiving 1, Dark Blue; Forgiving 2, Light Blue)
Model 5: Unforgiving. 

These agents exhibit similar alteration of their trust in agents, as these alterations of trust are based on particular sequence of advice they have encountered when accepting the cooperation of other agents – the tolerance level. However, once that particular limit is reached (e.g., 2 pieces of bad advice from a specific agent), then a judgment of untrustworthiness is made and a permanent disposition is made, which is not recoverable. (Unforgiving 1, Dark Lavender; Unforgiving 2, Light Lavender)

When an agent finds some Topic \( i \) it has been seeking is indeed discussed at some channel \( #k \), it then explores the postings in the meta-chat room to see if any other agent suggested that Topic \( i \) could be found at that channel.\(^\text{17} \) That is, when an agent finds an appropriate chat channel, it will reassess its trust with respect to all of the agents that have posted advice that would have proven correct, provided it is using a Forgiving 1, Forgiving 2, Unforgiving 1, or Unforgiving2 Trust Model.\(^\text{18} \)

There is an optional decision that could occur at the time an agent finds an appropriate chat channel. Should the agent also scan the postings to see if any agents have posted advice that would have proven wrong? If you want the agents to do this, check the option entitled “Assess Ignored Bad Advice” (see Figure 16). This will cause the agents to review posted advice that was found to be judged as incorrect when that correct chat channel is found.\(^\text{19} \)

![Assess Ignored Bad Advice](image)

Figure 16.

In TrustMe, you can define up to 10 agents per model, plus there are two groups of Forging and Unforgiving agents each that you may wish to define.

Note that as you select the particular Trust Model, the associated options appear on the display. Consider Trusting and Distrusting Models (Figure 17). You can select the number of agents for each (up to 10) and the associated Honesty Model (discussed in the next section).

![Agent Models](image)

Figure 17.

---

17 If the Agent went to some location because of advice, it will first reassess the trust in that source of that advice if the Agent is using a Forgiving or Unforgiving trust model.

18 When using Unforgiving Trust Models, once an agent has been judged as Distrusting, “good” advice is not assessed any more.

19 Note that the advice may be incorrect, but the agents posting the advice may not be deceptive; rather, a turbulent environment may underlie the incorrect posting or memory errors may have occurred. This will be discussed later (see Figure 22).
In addition, with Forgiving models and Unforgiving models (Figure 18), you must specify the tolerance level. Furthermore, with Forgiving models, you have the option of the initial trust between agents as either trusting (the default) or distrusting.

Note that simple distrusting an agent (the resulting state determined by the Trust Model) does not imply advice from untrusted agents will be ignored.

To trust or not trust an agent’s advice is determined explicitly by the advice resolution strategy selected (see Figure 19 below and prior Table 2).
For example, an agent (A9) may be judged as distrusted (by the Trust Model), and this would filter out A9 leaving the advice of agents A2 and A7 to, in fact, agree on the proper channel (#67).

However, if the answer resolution strategy does not explicitly take trust into account (i.e., Models 1 and 2 in Table 2), then advice may be repeatedly taken from distrusted sources.

Trust, then, is an internal attributional state that can be used to inform and influence specific behavioral decisions. As noted, TrustMe sets the default Multiple Advice Resolution strategy to Model 3, which only considers advice from trusted sources, then randomly selects from the resulting set.

Finally, there is one additional modification you can make – when the agents in the world should provide advice.

**Honesty Models**

As noted, when an agent views the message board, it always checks to see if it can generate advice. That is, it checks to see a) if another agent has a question outstanding, b) if it has answered that question, and c) if not, can it answer that question. Honesty models determine how that decision is made.

Advice generation is determined in TrustMe in terms of six choices from five Honesty Models (see “Honesty Model” slide control in prior Figure 17):

**Model 1: Honest**

These agents will always provide honest advice to an agent. Gossip does not impact this model.

**Model 2: Trusted**

These agents will provide honest advice only to a trusted agent otherwise the request is ignored. Gossip does not directly impact this model.

**Model 3: Trusted with Gossip**

These agents will provide honest advice only to trusted agents. However, even with trusted agents, it will check to see if there is a sufficient amount of gossip. If so, then the agent will ignore the request. If an agent is Untrusted, then the request is ignored. Furthermore, if the Distrusted Requests box is checked, these agents will generate trust-based negative gossip about the untrusted agent (see Figure 20).

**Model 4: Trusted with Gossip, Deceptive**

These agents are like Trusted with Gossip agents, except that if the requesting agent is not trusted, these agents will generate deceptive answers to that agent. Similarly, if Distrusted Requests box is checked, these agents will additionally generate trust-based negative gossip about the untrusted agent (see Figure 20).

---

20 This negative gossip is considered “trust-based” as the generating agent did not directly receive bad advice from that agent, but does not trust the asking agent.
**Model 5: Deceptive**
These agents will always provide *deceptive* advice to any other agent. Furthermore, if Distrusted Requests box is checked, these agents will additionally generate *negative gossip* about the requesting agent (regardless of the trust level).

**Model 6: Random**
These agents will randomly adopt one of the honesty models 1 through 5 each time they consider providing advice.

![Gossip Assertion](image)

This generates “false negative gossip” with Honesty Models 3, 4 and 5 under certain conditions.

![Figure 20.](image)

**When to believe advice.**

With respect to advice, trust models serve as filters in an overall advice resolution strategy that are applied before the random/maximum methods are applied (recall Table 2).

<table>
<thead>
<tr>
<th>DOES TRUST INFLUENCE ADVICE?</th>
<th>MORE THAN ONE ANSWER?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>YES</strong></td>
<td>YES</td>
</tr>
<tr>
<td><strong>NO</strong></td>
<td>NO</td>
</tr>
</tbody>
</table>

Consider the prior example from Figure 19. Agents A7 and A2 were trusted, but agent A9 was not trusted, then the answer resolution was simplified (*Maximum*) because they both gave the same channel (#64). If the answers were different, then the multiple advice resolution strategy engaged would be a *Random* choice between the two (Figure 21). In this case, the random choice resulted in a different channel being selected.
Under many circumstances, advice may not be correct for a variety of reasons. Consider two common types of situations under which advice could be incorrect: error and deception.

First, in error situations, an agent has intent to provide correct advice, but something has happened to invalidate the information passed along. For example, there may have been errors in the agent’s memory or perhaps the information itself has changed (i.e., a chat channel’s topic or the channel number itself).

In TrustMe 1.0, you actually have an option to make the task environment quite turbulent. What this means is that as you select higher turbulence values, TrustMe “rearranges” the items in their location at increasingly frequent intervals, thus invalidating the agents’ item memory (Figure 22).

Second, in deceptive situations, an agent deliberately provides incorrect information to other agents (see the Honesty Models discussed in the previous section).  

In either case, it is important and functional for agents to consider whether or not the source of advice is believable. In TrustMe, this means that if Agent i trusts Agent j, then

---

21 [stuff here on deception]
Agent i will believe any advice from Agent j. The mechanism that defines the trustworthiness of an agent is the agent’s trust model.

There is a variation on this, and this variation occurs in the presence of gossip.

**Gossip**

Gossip is the third element in the TAG model. Gossip functions as both a behavioral choice and as a social construct, for it reflects a very real social mechanism of communication and control (e.g., norm enforcement). Current communication technologies alter the speed and scope of their distribution and (presumably) their organizational effect. Once dispensed by face-to-face whispers, technology has gradually increased the breadth of reach and rate of dispersion over the years, where gossip is now are broadcast real-time on firm’s email or the nation-world Internet in seconds.

Recall that TrustMe can be viewed as an abstraction of a distributed community of simple collaboration as an advice model of search.

As we have seen, the *messages* in this model are simple. Thus far, the messages in TrustMe are these:

1. **Questions** posted to the group about finding where a topic is covered:
   
   Agent 3: Does anyone know where Topic 21 is discussed?

2. **Advice** posted to the group about a topic channel:
   
   Agent 5: Hey Agent 3, I think Topic 21 is covered in #94

3. **Responses** to advice:
   
   Agent 3: Thanks Agent 5, I will look for Topic 21 in #94 ;]

4. Chat-rooms that note agents **Joining** (to search) the channel:
   
   CH #94: Agent 3 HAS JOINED THE CHAT.

5. **Results of a particular search success**:
   
   Agent 3: Cool! I FOUND Topic 21 in #94 <^^>
   Agent 3: Thanks for the help, Agent 5!!! I found Topic 21

6. **Results of search failure**:
   
   M-SX----->Agent: 5: I GIVE UP...I checked 10 channels!! <+>
   M-SX----->Agent: 5: I CANNOT find Topic 41

7. **Notices when an agent leaves** the entire chat-room service:
   
   M-XG----->Agent: 2: I am GONE!! ;}
   M-XG----->CH #0: Agent 2 HAS LEFT THE CHAT.

8.
Gossip is basically about the information exchanges about behavior of advice sources shared among the group. Gossip is information spread about the events directly experienced by an agent.\(^{22}\)

There are certain circumstances, however, when false gossip can be spread as a malicious act!

**Types of gossip.**

If some Agent \(i\) received bad advice from some Agent \(j\), then Agent \(i\) may initiate gossip about the advice provided by Agent \(j\). This is *negative* gossip is generally based on direct experience.

For example, assume that Agent A4 gave advice to Agent A8, but when A8 went to the specified location, the item was not there.

Negative gossip is posted for all to see and is of the form:

“Agent A4 gives bad advice. posted by Agent A8”

On the other hand, if some Agent \(i\) received good advice from some Agent \(j\), then Agent \(i\) may initiate gossip about the trustworthiness of Agent \(j\). This is *positive* gossip also based on direct experience (see Figure 24).

“Agent A4 gives good advice. posted by Agent A8”

You may elect to use no gossip, negative gossip only, or negative and positive gossip together.

---

\(^{22}\) Our interpretation of gossip subsumes that of task-related “institutional gossip” as we see this as an important component of defining and maintaining informational coalitions.
Gossip Assertion Rules

The decision to generate gossip is based fundamentally on a propensity to communicate the occurrence of a negative event (i.e., the reception of bad advice) to others. This is controlled in the area shown in Figure 24. The choices are the following:

- **Never.** Never generate gossip.
- **Experience Only.** Generate gossip when an agent has received bad advice in search for an item.
- **Trust-Based.** Generate gossip when an agent has received bad advice in search for an item and the agent that has provided the bad advice is not trusted.

The latter two primary gossip model selections can be augmented with two additional choices.

- **Positive (Experience).** There is an option, as previously noted, to also generate positive gossip when good advice has been received, thereby creating the rudimentary component for reputation effects. (see subsequent section on gossip influence).
- **Untrusted Requests.** Finally, there is also an option for an agent to generate false negative gossip under certain circumstances when considering answering a request. Recall that under discussions of the Honesty Models, models 3, 4 and 5 allow the agents to generate false negative gossip if the asking agent is not trusted (providing the Untrusted Request option is checked).

Gossip Belief Rules

Given that gossip assertion is defined, the conditions under which gossip is accepted must be specified. There are three general options for belief:

**Never.**

Never believe gossip.
**Gossip Count (Threshold).**
Believe gossip only if there are $k$ gossip assertions (between 1 and 5) about another agent.

**Trusted Count (Threshold).**
Believe gossip only if there are $k$ gossip assertions (between 1 and 5) about another agent asserted by trusted agents.

Thus, one can specify that some level of belief in gossip about an agent is defined by an integer between 1 and 5, and elaborate on that constraint by enforcing a “trusted source” mechanism to the rule. Figure 25 shows the controls for this.

![Figure 25](image-url)
Gossip Impact Rules

A key question is: what do the agents do with gossip? In other words, what effect does gossip have on the behavior (directly or indirectly) of the agents?

Gossip is viewed as information that can play an important part in a group or organizational setting as surrogate experiences and attributional influences. In the model, this means that gossip can impact Trust and Advice situations.

Here are the options in TrustMe (see Figure 26):

**Taking Advice**
When agents consider taking advice, they will do so in accordance with the Trust Models selected. However, this option places an additional constraint on the “filter” depicted in Figure 27. Trust and gossip filters are applied first, and then any multiple answers remaining are subjected to the answer resolution strategy chosen.

**Giving Advice**
When this is checked, Honesty Models 3 and 4 (Trusted with Gossip; Trusted with Gossip, Deceptive) will incorporate gossip in addition to Trust Models in consideration to answer posted questions.

**Impacts Trust Model (State)**
Under this condition, whenever there is a need to check for gossip, the results of that check can impact the Trust Model state. That is, gossip, as well as experience, can alter trust. Note that when Positive gossip is selected (see Figure 24), gossip can have the effect of increasing trust as well as decreasing trust.

---

23 The gossip limit is the number of gossip assertions about a particular agent above which gossip is considered “significant.” You have control of this limit. For example, an agent may not “believe” gossip about an agent if there is only one assertion about the agent; however, an agent may begin to believe gossip of there are three or more such assertions.
References


Appendix A: TrustMe Interface

The overall window for TrustMe 1.0 is shown below as it initially appears…

Located at the lower right of the display are the primary controls for running TrustMe…

- **Initialize** Prepares TrustMe for a run and checks the user’s settings to see if there are any constraint violations or problems with the choices.

- **Start** Begins the simulation.

- **Reset** If problems are discovered or the user wishes to change settings, the user is directed to click on the Reset button.

- **Clear Text Box** The text box (or sometimes called the Trace Window) is the output window directly above these controls. The output is not cleared after runs; rather, the user must clear the window. Although this window scrolls, it is often easier to clear it in order to facilitate reviewing the results. All additional text is added at the “end” (or “bottom”) of this window.

- **Exit** This stops the TrustMe program entirely (i.e., closes the program).

For All Agents…
These choices apply to all agents participating in the simulation, regardless of the type of Agent Model selected.

**Memory.**
This defines the number of chunks an agent can recall, and is the same for all agents in the simulation.

*Impacts:*
- **Agent search.** An agent prefers to examine and believe its memory as a primary strategy. This defines the number of past locations (each as a chunk) an agent can recall.
- **Agent providing advice.** Agents provide advice based on their memory of items (content) viewed in past locations.
- **Agent considering advice.** Agents will consider advice from other agents; however, there is a limit to how much advice can be considered for a particular move. This limit is also determined by the Memory setting.
- **Default:** 14 (10% of search space)

**Restrict Advice Memory.**
By checking this option, the amount of advice to be considered by any Agent (in response to a posted question) is restricted to one (first-come, first-served).

**Decay.**
Decay determines a) if there are any errors in recall from memory and b) how the extent of those errors changes as a function of their “age” in memory, as defined by (see prior Figures 1 and 2):

\[
\text{Likelihood of accurate recall} = \text{Age}^{\log\beta}
\]

*Impacts:*
- All events that depend on memory as described under the Memory option.
- **Default:** 0 (no memory errors or trace degradation)

**Srch Limit.**
This is the Search Limit for all Agents in the simulation run. Specifically, it defines the maximum number of moves an Agent will make before it quits a given search goal. It is a measure of individual Agent “effort.”

*Impacts:*
- Individual and Organizational Effort measures.
- **Default:** 144 (entire search space)

**Turbulence.**
Turbulence is a control over the “validity of the task environment” in terms of the <Item: Location> pairings. With no turbulence, the pairing is fixed. As the environment becomes more turbulent, the more frequently the environment is disrupted (i.e., Items are randomly shifted to new Locations).

- Validity of Memory.
- Validity of Advice (thus Trust judgments).
- **Default:** 0 (no turbulence)
Selected Agent Models…
These define subgroups of agents that have particular characteristics with respect to trust and deception. For each subgroup chosen, all of the agents within that subset are the same.

Agent Models.
There are seven subgroups of five different Trust Models that define how good and bad advice impact the trust in the source of the advice (see discussion in section x). These are selected by the particular checkbox next to the name of the subgroup:

Trust Models
• Trusting. Always trust all other agents.
• Distrusting. Never trust another agent.
• Random Judgment. For each situation in the task, randomly determine whether or not to trust other agents (thus, trust varies within the task).
• Forgiving1. Trust in other agents defined by tolerance level bad (and good) advice. Distrust can be reversed by good advice as defined by tolerance level.
  o Tolerance Level. Defines the effects of sequences of good and bad advice on trust judgments.
  o Initial Distrust. This sets the initial state of Forgiving agents as distrusting all other agents.
• Forgiving2. Same as Forgiving1. You can simply define a second group of agents that are a variation of Forgiving1 agents (e.g., a different tolerance level).
• Unforgiving1. Trust in other agents defined by tolerance level for bad advice. Distrusted Agents cannot be trusted again.
  o Tolerance Level. Defines the effects of sequences of good and bad advice on trust judgments.
• Unforgiving2. Same as Unforgiving1. Similarly, you can define a second group of agents that are a variation of Unforgiving1 agents.

Number of Agents.
This defines the number of Agents in the particular Trust Model subgroup.

Honesty Model.
For each Trust Model subgroup, an Honesty Model must be defined that specifies how Agents in that subgroup respond to requests for advice from other Agents. Note that memory Decay settings can impact the results of these models; that is, there can be memory errors (and subsequent incorrect advice) within the context of answering honestly.
• Model 1. Always answer honestly.
• Model 2. Answer honestly, but only if the asking Agent is trusted; otherwise, do not answer.
• Model 3. Answer honestly if the asking Agent is trusted; otherwise, do not answer, plus…
  o If you have also stipulated that gossip matters in giving advice (Gossip Influence: Giving Advice is checked), if sufficient gossip is found (depending on the Gossip Belief model
selected), then the agent simply will withhold the advice from trusted agents.

- If the Gossip Assertion: Distrusted Request box is checked, then these agents will not only withhold the true answer from distrusted agents, but will spread negative gossip about them.

- **Model 4.** Answer honestly if the asking Agent is trusted; otherwise, be deceptive – post incorrect Advice, plus...
  - If you have also stipulated that gossip matters in giving advice (Gossip Influence: Giving Advice is checked), if sufficient gossip is found (depending on the Gossip Belief model selected), then the agent simply will withhold the advice from trusted agents.
  - If the Gossip Assertion: Distrusted Request box is checked, then these agents will not only be deceptive to the distrusted agents, but will spread negative gossip about them.

- **Model 5.** Answer deceptively, regardless of trust or knowledge of answer, plus...
  - If the Gossip Assertion: Distrusted Request box is checked, then these agents will not only be deceptive to all agents seeking advice, but will spread negative gossip about them.

- **Model 6.** Randomly select an Honesty Model (1 through 5) every time an advice situation arises.

**Tolerance**

For Trust Models Forgiving1 and 2 and Unforgiving1 and 2, tolerance defines the required number of bad advice received in a row to judge the source agent as not trustworthy (Distrusted state). For Trust Models Forgiving1 and 2, tolerance also defines the required number of good advice received in a row to move from a Distrusted state to a Trusted state. See Appendix C.

**Initial Distrust**

For Trust Models Forgiving1 and 2, this creates the initial condition where these agents initially do not trust any other agent. The default situation is that agents (for all Trust Models) initially trust all other agents.

**Balance**

When more than one Honesty Model is selected, you have the option of letting the software assign the tasks automatically, such that each agent (across all Honesty Models) has the same number of tasks to complete and maximizing the number of agents assignable across all models (subject to the equal task number constraint).

**Requests per Agent**

All agents defined in the model must have the same number of tasks, or requests, to complete. This slider control defines the number for each agent. Any constraint violation (i.e., not enough tasks to go around), the system will flag the condition.
To set the number of Requests for agents automatically, try these strategies:

- Select the specific number of agents for each model, and then slide the Requests per Agent slider to the maximum. TrustMe will automatically “scale back” the task requests to an appropriate number for the number of agents selected.
- Select the particular models you want to include, then click on the Balance button (see next section). This will tell TrustMe to automatically maximize the number of agents subject to the constraint of equal requests.

**Advice Resolution Strategy**

This defines how the agents in the simulation react to multiple sources of advice. There are four models that specify how to respond, and all agents in the simulation will use the same model…

- **Model 1**: Random – an answer is selected randomly from the set.
- **Model 2**: Maximum – an answer selected is based on the most frequently occurring suggestion from the set.
- **Model 3**: Random, Trusted – an answer is selected randomly from the set of advice from trusted agents only. (Default)
- **Model 4**: Maximum, Trusted – an answer is selected based on the most frequently occurring suggestion from trusted agents only.

**Assess Ignored Bad Advice**

When this box is checked, each agent will explore the advice posted by other agents, and make trust adjustments as if it had followed that advice.

**Gossip Assertion**

This area of checkboxes defines the conditions under which an agent posts gossip about another agent. Gossip is defined as posted information (positive, negative) about the credibility of a source of advice.

- **Never**. Gossip is never posted about another agent.
- **Experience Only**. Gossip is posted, based on the personal experiences of the agents. That is, the conditions that may lead to gossip (bad advice, good advice) must occur directly to the agent, whereupon the gossip is immediately posted.
- **Trust-Based**. Gossip is posed

---

Positive (Experience). When this is checked, gossip (positive) about agents that submit good advice received will be posted.

**Distrusted Requests**. This is used in accordance with Honesty Models 3, 4 and 5. Under certain conditions, when this is checked it will enable agents to spread negative gossip about distrusted agent without the need of direct experience.
Gossip Belief
This area of checkboxes defines the general conditions under which an agent believes gossip posted by another agent.

- **Never.** Gossip is not believed from any agent under any circumstances.
- **Gossip Count.** Gossip about some agent is believed by an agent only after \( n \) other agents have posted gossip, where \( 1 \leq n \leq 5 \). This \( n \) is specified in the Gossip Count (Belief) listbox.
- **Trusted Count.** Gossip about some agent is believed by an agent only after \( n \) other trusted agents have posted gossip, where \( 1 \leq n \leq 5 \). This \( n \) is specified in the Gossip Count (Belief) listbox.

Gossip Count (Belief) This listbox defines the \( n \) in the Gossip Count and Trusted Count belief models. The default is \( n = 1 \).

Gossip Influence
This area of checkboxes defines the general events that are influenced by gossip posted by another agent.

- **Taking Advice.** When an agent is reviewing advice posted by other agents, this selection will force the agent to engage a “gossip filter” whereby the source of advice will be evaluated with respect to the defined Gossip Belief model. That is, if there is sufficient gossip about an agent that has posted advice, it will be ignored.
- **Giving Advice.** When an agent is considering answering a question posted by another agent, this selection will force the agent to engage in a “gossip filter” whereby the agent that has posted the question will be evaluated with respect to the defined Gossip Belief model. That is, if there is sufficient gossip about an agent that has posted a question, that question will be ignored.
- **Impacts Trust Model.** If this is selected, then any time an agent checks to see if there is gossip about a source (according to the particular Gossip Belief model), the result of that evaluation impacts the Trust Model for that agent. That is, gossip influences the Trust Models as well as direct experience. If there is sufficient negative gossip, then trust my shift down (similar to receiving bad advice); if there is sufficient positive gossip (and this is allowed), then trust can shift up (similar to receiving good advice).

Communication
This section describes what information is displayed in the Trace Window regarding the moves and communication of the agents. There is a primary checkbox associated with Communication that, when checked (the default), allows agents to communicate. When it is not checked, agents do not communicate among each other – which is often used to create “base cases”.
Details of this section are explained in Appendix B.

- **Trace Gossip.** Displays all gossip (good and bad) posted from agents.
- **Trace Messages.** Displays all questions asked, answers posted, results of reviewing answers.
- **Trace Moves.** Displays the item an agent is seeking, where it is searching, and the results of moves in that search task.
- **Trace Agents.** Similar to Trace Moves, but only displays the end result of the search task and not the intermediate moves in the search.
- **Trace Trust.** Displays the events and states relevant to trust (i.e., advice, gossip, and search results impacting the states).

Logged

The Trace Window is used for quick assessments and, in fact, can be “overloaded” (a Windows constraint). By selecting the Logged checkbox, all traces chosen will be written to a specific textfile called “trustlog_iiii.txt” (where iii is a randomly chosen integer) in the path indicated in the Output File Location section below it. Note that you cannot change the name of this file until after it is written, though the random component will make most unique.

- The trustlog_iiii.txt file is very useful in subsequent analysis. You can even write specific routines that automatically do analysis on this text file.

Replications

In doing experiments with TrustMe, it is often useful to make manipulations and incorporate multiple runs that define “within cell” replications for the particular manipulation. This is similar to running a study with human participants, and incorporating “n participants per cell.”

- The default setting is one.
- When more than one is selected, every simulation is repeated with an initial random assignment of agents and items to locations in the grid.
  - The results of \( n > 1 \) simulations (replications) are then listed in the Trace Window at the end of the traces.

For the most part, the variance in TrustMe Replications is derived from the randomly chosen initial positions of the agents and the items.\(^{24}\)

**Show Details**

Selecting this checkbox will cause a listing in the Trace Window of the results of every replication as well as the summary across all replications. This information will also be written to the trustlog.txt file if the Logged option is selected.

\(^{24}\) Further variance can be incorporated by the “random” options in the models and strategies allowed: Advice Resolution Strategy, Trust Model, and Advice Generation Honesty Models.
Show Links
At the end of a simulation (or at the end of the last simulation run if \( n > 1 \) Replications are chosen), TrustMe will generate a connectivity graph that shows all of the agents and their final inter-agent dyadic trust status via three colors:

- **Green.** The two agents trust each other.
- **Blue.** One agent trusts the other, but this trust is not reciprocated.
- **Red.** The two agents do not trust each other.

If the Show Links checkbox is **unchecked** (the default is **checked**), then the last state of problem state (i.e., locations, items and agents) will be displayed.

Output File Location
It is often useful, especially with larger data sets and different simulation settings, to save the data for analysis. The Output File Location provides the mechanisms to define the location of files to be written.

- Note that the displayed path indicates where the trustlog.txt file will be written, if that option is selected.
- The displayed path will indicate where the data files will be written, if that option is selected.

Now the area to the immediate **left** of this section allows you to define the name and type of the data files that will be generated from a TrustMe simulation run.

Create Data File
Selecting this checkbox enables the Output File Location section (as does selecting the Logged checkbox). Once enables, you can define the location and name of the data file. The details of the data file are presented in Appendix B.

Append Data File
If you make multiple runs, it may be desirable to append the runs to a primary data file. If not, you could simply cut and paste the files to Excel, or merge them with a statistics package.

File Name
This is where you can type in the file name for the data file. If you type in a duplicate name (of an existing file in the path selected), you will be warned with an alter box. At that time, you can RESET to define a new name, or START to run the simulation without generating a data file.

Add Column Titles
For many statistical packages and spreadsheets like Excel, it is quite useful to have the first row of the output data file to be the names of the variables that will eventually be the column titles in the analysis software. Selecting this option generates these titles (unless the Append Data File is selected).
Generate Trust Matrices
The core concept in TrustMe is trust as specified by the Trust Models. Operationally, an $N \times N$ matrix defining how all agents trust each other captures this information. The values in the matrix depend on the particular Trust Models involved (see Appendix B for details), but all of the models define values $\leq -100$ as not trustworthy and values $\geq 0$ as trustworthy.

Filename: The name of the trust matrix file is automatically derived from the data filename you have created. It is simply that filename plus “mtx” -- not an extension, but three specific characters added to the filename. For example, if you defined your data file to be “dataset1” then associated trust matrix file generated would be “dataset1mtx”.

Note that you also have the option of generating particular formatted output files that can be directly read by network analysis programs: UCINET and Netminer.

Detailed Data
When this checkbox is selected (not a default), then a trust matrix is generated every time there is a change in a trust value. This obviously can lead to very large data sets, but provides a rich history of matrices that define the dynamics of trust between all agents. The trust matrix identifier in the second column uniquely defines each matrix. The identifier is simply an incremental integer counter.
Appendix B: TrustMe Output

Output from TrustMe can take several forms and includes much data. There are three primary forms of output from TrustMe: traces, trust matrices, and the data file.

Traces.
First, consider the traces you can select via this section of the display shown in Figure B1.

The default setting is the selection of the Communication checkbox. What this does is specifically allow agents to communicate as defined by the other settings (e.g., gossip).

Now, depending on your other selections, you can generate traces that are displayed in the Trace window (i.e., the output window on the bottom right of the TrustMe display).

Although the Trace window is convenient, there are two problems associated with it concerning data output. First, there is actually a limit of how much text it can hold. Second, you cannot “save” what is displayed in the Trace window to a file.

To save the output traces selected, you can select the Logged checkbox (Figure B2). This will inhibit output to the Trace window and redirect it to a special text file called: trustlog.txt.

As you cannot change the name of this special file, it will write over any prior trustlog.txt.

You can, however, alter where this file will be written. This is done via the Output File Location part of the display (see Figure x).

Now, what exactly are the traces indicated?

---

25 If the Communication checkbox is not checked, then no communication between agents is allowed – that is, no postings are permitted.

26 You may get a run time error indicating an invalid procedure call. This is one indicator that the text window is “full”.
Trace Gossip. When gossip is allowed (and then only if gossip occurs), this option will generate a trace of that gossip. This includes GOOD as well as BAD gossip, depending on selection of options (see prior Figure 24).

R---->Agent: 4 posts GOOD GOSSIP about Agent 2
R---->Agent: 3 posts BAD GOSSIP about Agent 5
R---->Agent: 6 posts BAD GOSSIP about Agent 5
R---->Agent: 5 posts BAD GOSSIP about Agent 6
R---->Agent: 6 posts BAD GOSSIP about Agent 4
R---->Agent: 6 posts GOOD GOSSIP about Agent 3

Trace Messages. Messages include a larger group of traces. In a sense, it is an omniscient overview of the events occurring from an observer’s perspective, but not including certain information (i.e., gossip, trust).

**Questions**

M-Q---->Agent 3: Where is Item 49?
M-Q---->Agent 5: Where is Item 97?
M-Q---->Agent 4: Where is Item 73?
M-Q---->Agent 1: Where is Item 1?
M-Q---->Agent 2: Where is Item 25?

**Answers**

MA---->Agent: 2 answering Agent 1
M        Agent: 2 says Item 1 is in Stack 65
MA---->Agent: 1 answering Agent 6
M        Agent: 1 says Item 122 is in Stack 37

**Reviewing Answer Strategy and Move Decisions**

M------>Agent 1 reviewed answer from Agent 2
M        Agent 1 is taking ADVICE from 2
M        Agent 1 is moving to Stack 65 for Item 1
M------>Agent 6 reviewed answer from Agent 1
M        Agent 6 is taking ADVICE from 1
M        Agent 6 is moving to Stack 37 for Item 122

**Advice Impact**

M-F-------->Agent: 6 found Item 122 with GOOD advice from Agent 1
M-B---->Agent: 1 received BAD advice from Agent 2
M-F------------->Agent: 5 found Item 97 without advice.

Trace Moves. This indicates what an Agent is seeking, where an Agent is going to seek it, and the result of that search move.

L----> Agent: 4 needs Item 73. At Stack: 14 that has Item 12
M-B---->Agent: 4 did NOT FIND item 73 at Stack 14
L----> Agent: 6 needs Item 121. At Stack: 52 that has Item 7
M-B---->Agent: 6 did NOT FIND item 121 at Stack 52
L-**-> Agent: 4 needs Item 73. At Stack: 4 that has Item 73
A-F---->Agent: 4 found Item 73
Trace Agents. This option is similar to Trace Moves, but only notes what is sought and the results…

- Agent: 4 needs Item 75
- Agent: 3 found Item 50
- Agent: 3 needs Item 51
- Agent: 4 found Item 75
- Agent: 4 needs Item 76
- Agent: 2 search exceeded.
- Agent: 2 cannot find item: 25

Trace Trust. This option displays the trust states and changes for the agents as well as the events relative to those states – bad advice and gossip. For example,…

- Agent: 1 posts GOOD GOSSIP about Agent 2
- Agent 1 ALWAYS TRUSTS Agent 5
- received BAD ADVICE from Agent 5
- Agent: 1 posts BAD GOSSIP about Agent 5

In this case, Agent 1 received GOOD advice from Agent 2 and posted GOOD gossip about that event. Agent 1, however, received BAD advice from Agent 5, but the Trust Model for Agent 1 turns out to be the Trusting model (i.e., always trusts advice). Nevertheless, the Honesty Model for Agent 1 permits Agent 1 to actually post BAD gossip about Agent 5.

Consider Agent 3’s events…

- Agent 3 now does NOT TRUST Agent 5
- received BAD ADVICE from Agent 5
- Agent: 3 posts BAD GOSSIP about Agent 5

Agent 5 gave Agent 3 BAD advice, but Agent 3 has a Forgiving Trust Model with a Tolerance of 1, so it now does not trust Agent 5. In addition, Agent 3’s Honesty Model results in BAD gossip being posted about Agent 5.

So, is Agent 5 a bad agent? Surprisingly, it is not! Agent 5 is actually the victim of a Turbulent Environment and posts BAD advice unintentionally. This is remedied further in the simulation, when Agent 3 examines if there are any Agent posting that deem reassessment of trust…

- Agent 3 sees GOOD advice posted by Agent 5
- Agent 3 reassessing (+) Agent 5
- Agent 3 AGAIN TRUSTS Agent 5

As with all of these options, it is important to realize how the various settings may interact for the results obtained.
For the example we have just explored, here were the key settings that varied from the defaults...

Six Agents. Two Trusting (Honesty Model 2), two Forgiving (Honesty Model 2), and two Unforgiving (Honesty Model 1).

144 Items. This translates into 24 Items for each Agent.

Gossip Assertion. Experience Only, Positive Experience.

Gossip Belief. Gossip Count.

Gossip Influence. Taking Advice, Giving Advice.

Turbulence. Setting 7.

The listing of the summary data for the run is as follows.

---------- Starting Sim Run Summary:  1
Total Agents: 10
  Trusting: 0
  Honesty: 0
  Distrusting: 0
  Honesty: 0
  Random: 0
  Honesty: 0
  Forgiving1: 0
  Honesty: 0
  Tolerance: 0
  Initial: 1
  Forgiving2: 0
  Honesty: 0
  Tolerance: 0
  Initial: 1
  Unforgiving1: 0
  Honesty: 0
  Tolerance: 0
  Unforgiving2: 0
  Honesty: 0
  Tolerance: 0

Good Advice: 90
Bad Advice: 0
Questions Asked: 100
Honest Answers: 101
Deceptive Answers: 0
Deceptive Answers to Untrust: 0
Advice Ignored: 0
  -- Trust: 0
  -- Rumor: 0
Rumors Asserted: 0
Rumors Taken: 0
Info Withheld: 0
  -- Trust: 0
  -- Rumor: 0
  -- Type: 0
% Items Found: .
Trust Adjustments Down via Rumor:
Trust Adjustments Down via Experience:
Turbulence Level:
Locations Disrupted:
  Ave Locations Searched by Agent: 192
  Ave Agent Effort: 460,704
Total Locations Searched: 1,920
Total Effort: 4,607,043
Conflicts: 0
Conflict Ratio: .
Total Links: 90
Ave Trust Duration: 1,920.
Per Cent Duration: 1.
Ave Conflict Duration: .
Time: 1,920
Max Agent Effort: 1232228
Max Agent Visits: 473
Max Agent Time: 1003
Persistent Agents: 0
Locations Searched under Persistence:
Effort Exerted under Persistence:
Timed Out: 1
----------- End Sim Run Summary: 1

----------- Summary of All Runs -----------
Simulations Completed: 1
Total Agents: 10
  Trusting: 0
  Honesty: 0
  Distrusting: 0
  Honesty: 0
Random: 0
  Honesty: 0
Forgiving1: 0
  Honesty: 0
  Tolerance: 0
  Initial: 1
Forgiving2: 0
  Honesty: 0
  Tolerance: 0
  Initial: 1
Unforgiving1: 0
  Honesty: 0
  Tolerance: 0
Unforgiving2: 0

Summary data across all runs is then listed (last).
**Trust Matrices.**
The trust matrix output file (name mtx file) is a file that defines how all agents trust each other in terms of the trust values assigned during program operation. The values in the matrix depend on the particular Trust Models involved (see Appendix C for details), but all of the models define values $\leq -100$ as not trustworthy and values $\geq 0$ as trustworthy.

**Default Format of Matrix File.** The default format of the matrix file is as follows:

- **Column 1:** The Replication run number (1, 2, …), where the default is a single (1) run.
- **Column 2:** A flag indicating the initial state of the matrix (who trusts whom) and the final state of the matrix. If the default settings are used, this is a “0” for the initial state, and a “1” for the final state.
- **Column 3..N:** When there are $N$ agents in the simulation, the remaining columns represent the $N$ agents, for a total of $N + 2$ columns in total.

**Rows:** For $N$ agents, there is $2N$ rows output with the default settings (i.e., the initial state matrix and the final state matrix). The row defines who the agent trusts; that is, for a given row $i$, the column entries define how agent $i$ trusts the other agents 1 .. $N$.

**Example 1:** In the following trust matrix file, there are 5 agents that a) initially trust each other, and b) still trust each other at the end of the simulation.

```
1 0 0 0 0 0 0
1 0 0 0 0 0 0
1 0 0 0 0 0 0
1 0 0 0 0 0 0
1 0 0 0 0 0 0
1 1 0 0 0 0 0
1 1 0 0 0 0 0
1 1 0 0 0 0 0
1 1 0 0 0 0 0
```

Initial state trust matrix

```
1 1 0 0 0 0 0
1 1 0 0 0 0 0
1 1 0 0 0 0 0
1 1 0 0 0 0 0
1 1 0 0 0 0 0
```

Final state trust matrix

**Example 2:** In the following trust matrix file, there are 5 agents that a) initially trust each other, and b) have trust changes by the end of the simulation.
This is interpreted as follows. In the final state trust matrix (beginning on row 6), agent 1 trusts agents 2 through 4 (column entries in those rows are zeros), but does not trust agent 5 (in the last column, -100). In fact, agents 2, 3 and 4 also do not trust agent 5, though they trust each other.

**Detailed Data.**

When this checkbox is selected (not a default), then a trust matrix is generated every time there is a change in a trust value. This obviously can lead to very large data sets, but provides a rich history of matrices that define the dynamics of trust between all agents. The trust matrix identifier in the second column uniquely defines each matrix. The identifier is simply an incremental integer counter.

Let’s look at the results of the previous Example 2, but with the Detailed Data box checked. When we look at the “mtx” file generated, it looks quite different…

**Example 2a:** In the following trust matrix file, there are 5 agents that a) initially trust each other, and b) have trust changes by the end of the simulation. Here we have checked the Detailed Data box located under the Generate Trust Matrices box.

Note that the initial state trust matrix is first (with the same trust matrix identifier = 0); there are four “intermediate” state trust matrices (with identifiers 2 through 4); and the final state trust matrix is still last (with a different identifier = 5)
The usefulness of these matrices is in the subsequent analysis by statistical software (e.g., the analysis of cliques, distance metrics and neighborhood analysis, emergence of trusted structures) in conjunction with analysis of the manipulations made that generated the matrices.
**Data Files.**

The primary data file that is generated (apart from the trace file and the trust matrix file) by TrustMe is based on each record (row) defining the results of a run (or replication). That is, if you use the default setting for the Replications (which is 1), then your output data file will have one row.

Each *row* of the output data file corresponds to a replication. This is beneficial for subsequent analysis by commercial statistical packages.

The reason for this is that each run/replication is considered, in an experimental sense, a “manipulation” that is defined by the particular group of selected settings (Trust Models, Honesty Models, etc.). Consequently, when you wish to “formally” experiment with a group of settings, the Replication option allows you to fill your manipulation cell with multiple (albeit synthetic) observations that have an initial random component – the location of item/concepts and agents.

Each *column* in the output data file corresponds to a particular output data item associated with a replication.

There are currently 129 columns of data for each replication.

<table>
<thead>
<tr>
<th>Col</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SessID</td>
<td>System assigned random identifier for session for all replications</td>
</tr>
<tr>
<td>2</td>
<td>RepNum</td>
<td>Replication number (user selects how many replications)</td>
</tr>
<tr>
<td>3</td>
<td>NAgents</td>
<td>Number of total agents in the session (all Models)</td>
</tr>
<tr>
<td>4</td>
<td>NTrust</td>
<td>Number of Trusting agents (Model 1)</td>
</tr>
<tr>
<td>5</td>
<td>HonMT</td>
<td>Honesty model for Model 1 agents</td>
</tr>
<tr>
<td>6</td>
<td>NDisT</td>
<td>Number of Distrusting agents (Model 2)</td>
</tr>
<tr>
<td>7</td>
<td>HonMD</td>
<td>Honesty model for Model 2 agents</td>
</tr>
<tr>
<td>8</td>
<td>NRnd</td>
<td>Number of Random Judgment agents (Model 3)</td>
</tr>
<tr>
<td>9</td>
<td>HonMR</td>
<td>Honesty model for Model 3 agents</td>
</tr>
<tr>
<td>10</td>
<td>NForg1</td>
<td>Number of Forgiving1 agents (1st set of Model 4)</td>
</tr>
<tr>
<td>11</td>
<td>HonMF1</td>
<td>Honesty model for Forgiving1 agents</td>
</tr>
<tr>
<td>12</td>
<td>TolF1</td>
<td>Tolerance level for Forgiving1 agents</td>
</tr>
<tr>
<td>13</td>
<td>InitDF1</td>
<td>Initial trust state for Forgiving1 agents (trusting, distrusting)</td>
</tr>
<tr>
<td>14</td>
<td>NForg2</td>
<td>Number of Forgiving2 agents (2nd set of Model 4)</td>
</tr>
<tr>
<td>15</td>
<td>HonMF2</td>
<td>Honesty model for Forgiving2 agents</td>
</tr>
<tr>
<td>16</td>
<td>TolF2</td>
<td>Tolerance level for Forgiving2 agents</td>
</tr>
<tr>
<td>17</td>
<td>InitDF2</td>
<td>Initial trust state for Forgiving2 agents (trusting, distrusting)</td>
</tr>
<tr>
<td>18</td>
<td>NUForg1</td>
<td>Number of Unforgiving1 agents (1st set of Model 5)</td>
</tr>
<tr>
<td>19</td>
<td>HonMUF1</td>
<td>Honesty model for Unforgiving1 agents</td>
</tr>
<tr>
<td>20</td>
<td>TolUF1</td>
<td>Tolerance level for Unforgiving1 agents</td>
</tr>
<tr>
<td>21</td>
<td>NUForg2</td>
<td>Number of Unforgiving2 agents (2nd set of Model 5)</td>
</tr>
<tr>
<td>22</td>
<td>HonMUF2</td>
<td>Honesty model for Unforgiving2 agents</td>
</tr>
<tr>
<td>23</td>
<td>TolUF2</td>
<td>Tolerance level for Unforgiving2 agents</td>
</tr>
<tr>
<td>24</td>
<td>MSize</td>
<td>Item Memory size for all agents</td>
</tr>
<tr>
<td>25</td>
<td>AdvMemSz</td>
<td>Advice Memory size for all agents</td>
</tr>
<tr>
<td>26</td>
<td>MDecay</td>
<td>Memory decay value for all agents</td>
</tr>
<tr>
<td>27</td>
<td>Slimit</td>
<td>Search limit for all agents</td>
</tr>
<tr>
<td>28</td>
<td>SLPCnt</td>
<td>Limit as percentage of total possible locations (144)</td>
</tr>
<tr>
<td>29</td>
<td>NTasks</td>
<td>Total number of tasks (items) for each replication</td>
</tr>
<tr>
<td>30</td>
<td>TskAgent</td>
<td>Average number of tasks per agent</td>
</tr>
<tr>
<td>31</td>
<td>Turb</td>
<td>Environmental turbulence setting</td>
</tr>
<tr>
<td>32</td>
<td>GosNev</td>
<td>Agents never gossip (flag = 1)</td>
</tr>
<tr>
<td>Col</td>
<td>Identifier</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>33</td>
<td>GosExp</td>
<td>Gossip generated only via direct experience (flag = 1)</td>
</tr>
<tr>
<td>34</td>
<td>GosTrust</td>
<td>Gossip generated via trust model (flag = 1)</td>
</tr>
<tr>
<td>35</td>
<td>GosGood</td>
<td>Assert and listen to Good gossip (flag = 1)</td>
</tr>
<tr>
<td>36</td>
<td>GosBad</td>
<td>Assert bad gossip based on trust model (flag = 1)</td>
</tr>
<tr>
<td>37</td>
<td>Detail</td>
<td>Assess ignored advice to adjust trust if possible (flag = 1)</td>
</tr>
<tr>
<td>38</td>
<td>GosLimit</td>
<td>Attend to gossip only after this limit is reached</td>
</tr>
<tr>
<td>39</td>
<td>AnsMult</td>
<td>Advice Resolution Strategy</td>
</tr>
<tr>
<td>40</td>
<td>GosBLF</td>
<td>Listens to gossip (0 = never, 1 = count, 2 = trustmodel)</td>
</tr>
<tr>
<td>41</td>
<td>InfTake</td>
<td>Gossip influences taking advice (flag = 1)</td>
</tr>
<tr>
<td>42</td>
<td>InfGiv</td>
<td>Gossip influences giving advice (flag = 1)</td>
</tr>
<tr>
<td>43</td>
<td>InfTrust</td>
<td>Gossip impacts trust model, when checking gossip (flag = 1)</td>
</tr>
<tr>
<td>128</td>
<td>Comm</td>
<td>Was any communication allowed (flag = 1)</td>
</tr>
</tbody>
</table>

**Dependent values from replication**

<p>| 44  | TotGAdv    | Total good advice (for task) received for all agents |
| 45  | TotBAdv    | Total bad advice (for task) received for all agents |
| 46  | TotQues    | Total questions posted by all agents |
| 47  | AvQues     | Average questions posted by agent |
| 48  | HonPost    | Total honest advice posted by all agents |
| 49  | SolvNAdv   | Items found without any advice |
| 50  | DecPost    | Total deceptive advice posted |
| 51  | DecPCTot   | Average deceptive advice posted (over all advice) |
| 52  | DecRatio   | Deceptive to honest advice ratio |
| 53  | DecUNT     | Total deceptive answers to distrusted agents |
| 54  | TotAdTak   | Total advice taken over all agents |
| 55  | TotAdIg    | Total advice ignored over all agents |
| 56  | IgTrust    | Advice ignored because of distrust of source |
| 57  | IgGos      | Advice ignored because of believed gossip |
| 58  | GosPost    | Total gossip posted (positive, negative) |
| 59  | GGpost     | Positive gossip posted |
| 60  | BGpost     | Bad gossip posted |
| 61  | GosTaken   | Total gossip believed (ignored advice, info withheld) |
| 62  | InfoHeld   | Total information (good task advice) withheld |
| 63  | InfoHTrst  | Information withheld because of distrust |
| 64  | InfoHGs    | Information withheld because of believed gossip |
| 110 | PCGosTkn   | Percent gossip believed: GosTaken/GosPost |
| 65  | TotCom     | Total communication in task (questions+ all answers) |
| 66  | AvePAgent  | Aver communication per agent |
| 67  | AvePTask   | Aver communication per task (items) |
| 68  | ChgTdT     | Total judgments changes: trust → distrust |
| 69  | ChgDtT     | Total judgment changes: distrust → trust |
| 70  | AvTdTAg    | Ave agent judgment changes: trust → distrust |
| 71  | AvDtTAg    | Ave agent judgment changes: distrust → trust |
| 72  | TdTGos     | Total times gossip causes change: trust → distrust |
| 73  | TdExp      | Total times bad experience causes change: trust → distrust |
| 74  | JUP        | Total trust judgments up (excluding non-change) |
| 75  | JDN        | Total trust judgments down (excluding non-change) |
| 76  | JUPpAg     | Ave judgments up per agent |
| 77  | JDNpAg     | Ave judgments down per agent |
| 78  | TLINKS     | Total links possible between all agents |
| 79  | TrstLK     | Number of trusted links at end (of task) |
| 80  | XYTrst     | Number of pure (symmetric) trusted links at end |
| 81  | TRatio     | Trust Ratio, trusted links to total: TrstLK/TLINKS |
| 82  | PTRatio    | Pure trust ratio: XYTrst/TLINKS/2 |</p>
<table>
<thead>
<tr>
<th>Column</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dependent values from replication (cont.)</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>TotCftLK</td>
<td>Number of conflict (distrusted) links at end (of task)</td>
</tr>
<tr>
<td>84</td>
<td>YCflt</td>
<td>Number of asymmetric distrusted links</td>
</tr>
<tr>
<td>85</td>
<td>XYCflt</td>
<td>Number of pure (symmetric) distrusted links</td>
</tr>
<tr>
<td>86</td>
<td>DRatio</td>
<td>Distrust Ratio, distrusted links to total: TotCftLK/TLINKS</td>
</tr>
<tr>
<td>87</td>
<td>PDRatio</td>
<td>Pure distrust ratio: XYCflt/TLINKS/2</td>
</tr>
<tr>
<td>127</td>
<td>AsTRatio</td>
<td>Asymmetric ratio: YCflt/TLINKS/2</td>
</tr>
<tr>
<td>88</td>
<td>NRelPer</td>
<td>Average possible links per agent: NAgents - 1</td>
</tr>
<tr>
<td>89</td>
<td>AvTrAgPP</td>
<td>Ave agents an agent trusts per period</td>
</tr>
<tr>
<td>90</td>
<td>AvDtAgPP</td>
<td>Age agents an agent distrusts per period</td>
</tr>
<tr>
<td>91</td>
<td>LocDisr</td>
<td>Total task environment disruptions</td>
</tr>
<tr>
<td>92</td>
<td>TLocPAgent</td>
<td>Ave search effort per agent: TotSrch/NAgents</td>
</tr>
<tr>
<td>93</td>
<td>TotSLK</td>
<td>Total search (effort) slack: total visits possible – total visits</td>
</tr>
<tr>
<td>94</td>
<td>SLKpAgT</td>
<td>Ave slack per agent: TotSLK/NAgents</td>
</tr>
<tr>
<td>95</td>
<td>SLKPTsk</td>
<td>Ave slack per task</td>
</tr>
<tr>
<td>96</td>
<td>SLKpct</td>
<td>Ave percent of search limit used per agent</td>
</tr>
<tr>
<td>97</td>
<td>MAXTime</td>
<td>Maximum search effort taken over all agents (task time)</td>
</tr>
<tr>
<td>98</td>
<td>AVEef1</td>
<td>Ave Effort1 taken per agent over all tasks</td>
</tr>
<tr>
<td>99</td>
<td>MAXef1</td>
<td>Maximum Effort1 taken by an agent</td>
</tr>
<tr>
<td>100</td>
<td>MINef1</td>
<td>Minimum Effort1 taken by an agent</td>
</tr>
<tr>
<td>101</td>
<td>SDef1</td>
<td>Stdev in Effort1, over all agents</td>
</tr>
<tr>
<td>102</td>
<td>AVEef2</td>
<td>Ave Effort2 taken per agent over all tasks</td>
</tr>
<tr>
<td>103</td>
<td>MAXef2</td>
<td>Maximum Effort2 taken by an agent</td>
</tr>
<tr>
<td>104</td>
<td>MINef2</td>
<td>Minimum Effort2 taken by an agent</td>
</tr>
<tr>
<td>105</td>
<td>SDef2</td>
<td>Stdev in Effort2, over all agents</td>
</tr>
<tr>
<td>106</td>
<td>AVEef3</td>
<td>Ave Effort3 taken per agent over all tasks</td>
</tr>
<tr>
<td>107</td>
<td>MAXef3</td>
<td>Maximum Effort3 taken by an agent</td>
</tr>
<tr>
<td>108</td>
<td>MINef3</td>
<td>Minimum Effort3 taken by an agent</td>
</tr>
<tr>
<td>109</td>
<td>SDef3</td>
<td>Stdev in Effort3, over all agents</td>
</tr>
<tr>
<td>110</td>
<td>PCGosTkn</td>
<td>(defined above)</td>
</tr>
<tr>
<td>111</td>
<td>TasksDone</td>
<td>Total tasks completed (items found)</td>
</tr>
<tr>
<td>112</td>
<td>Quits</td>
<td>Number of times agents search failed</td>
</tr>
<tr>
<td>113</td>
<td>PCProb</td>
<td>Percent of problem solved: TasksDone/NTasks</td>
</tr>
<tr>
<td>114</td>
<td>TotRes</td>
<td>Total possible group effort: agents x search limit x tasks</td>
</tr>
<tr>
<td>115</td>
<td>TotSrch</td>
<td>Total search effort used by group</td>
</tr>
<tr>
<td>116</td>
<td>PCTotRes</td>
<td>Percent of search effort used by group</td>
</tr>
<tr>
<td>117</td>
<td>ProbTime</td>
<td>Largest search effort for an agent over all tasks (slowest agent)</td>
</tr>
<tr>
<td>118</td>
<td>TotEf1</td>
<td>Total problem effort using Effort1</td>
</tr>
<tr>
<td>119</td>
<td>TotEf2</td>
<td>Total problem effort using Effort2</td>
</tr>
<tr>
<td>120</td>
<td>TotEf3</td>
<td>Total problem effort using Effort3</td>
</tr>
<tr>
<td>121</td>
<td>PCTWoAdv</td>
<td>Percent of problem solved without advice</td>
</tr>
<tr>
<td>122</td>
<td>AvEf1</td>
<td>Ave agent Effort1 per task</td>
</tr>
<tr>
<td>123</td>
<td>AvEf2</td>
<td>Ave agent Effort2 per task</td>
</tr>
<tr>
<td>124</td>
<td>ProdTC</td>
<td>Resources used per tasks completed: TotSrch/TasksDone</td>
</tr>
<tr>
<td>125</td>
<td>ProdTA</td>
<td>Resources used per problem: TotSrch/NTasks</td>
</tr>
<tr>
<td>126</td>
<td>TotRel</td>
<td>Total possible relations for group: NAgents * (NAgents -1)/2</td>
</tr>
<tr>
<td>127</td>
<td>AsTRatio</td>
<td>(defined above)</td>
</tr>
<tr>
<td>128</td>
<td>Comm</td>
<td>(defined above)</td>
</tr>
<tr>
<td>129</td>
<td>AvEf1</td>
<td>(defined above)</td>
</tr>
</tbody>
</table>
Generated Trust Matrices

When the Generate Trust Matrices option is selected with the file output options, you have two choices for the format of the output trust matrices: UCINET and NetMiner. These are two excellent tools for analysis and visualization of network structures. UCINET is available at a reduced cost from Analytic Technologies (www.analytictech.com) and NetMiner is available at a reduced cost for academic users (www.netminer.com). Once output, these files can then be directly imported to the particular software package.

For example, consider a problem that is configured as follows:

The resulting visualization shows definite differences in trust symmetries. However, the output file (Example_UCINET) can be imported to UCINET and results in an interesting visualization itself.

The Example_UCINET output file actually contains TWO matrices: INITIAL and FINAL. These are the trust matrices resulting from the initialization specifications (INITIAL) and the final one at the end of the run (FINAL).

First, consider how the agents are identified in the output. Each agent is identified as:

<id><trust model><honesty model>

where the <id> is an integer assigned by the program, the <trust model> is a letter code that refers to the five options noted in the Trust Models discussion on page 28.

The <trust model> code is as follows (related to the Trust Model descriptions):

T = Model 1 (Trusting)  F = Model 4 (Forgiving)
D = Model 2 (Distrusting)  UF = Model 5 (Unforgiving)
RND = Model 3 (Random)
The <honesty model> is a code that refers to the five Honesty Models discussed in that section on page 41.

Hon  = Model 1 (Honest)
TGNL = Model 2 (Honest, no deception
TNGNL = Model 3 (Honest, possible deception)
TGL = Model 4 (Honest, deceptive)
LiarG = Model 5 (Deceptive)
Rand = Model 6 (Random)

Thus, the following identifiers would be interpreted as described:

1-T-Rand  Agent 1 is a Trusting agent that Randomly selects its honesty model
8-D-Liar  Agent 8 is Deceptive and Dishonest
16-F-TGNL Agent 16 is Forgiving and is Honest to those it trusts
18-RND-TGL Agent 18 Randomly decides if it trusts an agent and is Honest to those it trusts, otherwise it is Deceptive

Let’s look at a visualization of the FINAL trust matrix using the Netdraw facility in UCINET. The matrix was adjusted by applying a multidimensional scaling model (MDS) in Netdraw to layout to impose distance metrics based on links.

We purposely set the “difficult” agents to be outliers in this group…indeed they are.
Appendix C: Trust Model Details

The following augmented transition networks describe the primary trust models in the context of how good (+) and bad (-) advice, and sequences of good and bad advice, impacts their trust judgments for each of the Trust Models defined in TrustMe. Trust judgments are in either of one of two states: Trust, or Distrust. For each state, the numerical value for the associated trust state as it appears in the trust matrices is provided. Recall that state values \( \leq -100 \) are considered in a state of Distrust (i.e., not trustworthy) and state values \( \geq 0 \) are in a state of Trust (i.e., trustworthy).

**Trusting.**

**Distrusting.**

**Random Judgment.**
Forgiving1, Forgiving2. (Note: You can set the initial state to Distrust or Trust.)

Tolerance = 1

Tolerance = 2
Tolerance = 3

Tolerance = 4
Tolerance = 5
Unforgiving1, Unforgiving2.

Tolerance = 1

Tolerance = 2

Tolerance = 3
Tolerance = 4

Tolerance = 5