Addressing the Issues of Coalitions and Collusion in Multiagent Systems

by

Reid C. Kerr

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In the field of multiagent systems, trust and reputation systems are intended to assist agents in finding trustworthy partners with whom to interact. Earlier work of ours identified in theory a number of security vulnerabilities in trust and reputation systems, weaknesses that might be exploited by malicious agents to bypass the protections offered by such systems. In this work, we begin by developing the TREET testbed, a simulation platform that allows for extensive evaluation and flexible experimentation with trust and reputation technologies. We use this testbed to experimentally validate the practicality and gravity of attacks against vulnerabilities. Of particular interest are attacks that are *collusive* in nature: groups of agents (*coalitions*) working together to improve their expected rewards. But the issue of coalitions is not unique to trust and reputation; rather, it cuts across a range of fields in multiagent systems and beyond. In some scenarios, coalitions may be unwanted or forbidden; in others they may be benign or even desirable. In this document, we propose a method for detecting coalitions and identifying coalition members, a capability that is likely to be valuable in many of the diverse fields where coalitions may be of interest. Our method makes use of clustering in *benefit space* (a high-dimensional space reflecting how agents benefit others in the system) in order to identify groups of agents who benefit similar sets of agents. A statistical technique is then used to identify which clusters contain coalitions. Experimentation using the TREET platform verifies the effectiveness of this approach. A series of enhancements to our method are also introduced, which improve the accuracy and robustness of the algorithm. To demonstrate how this broadly-applicable tool can be used to address domain-specific problems, we focus again on trust and reputation systems. We show how, by incorporating our work into one such system (the existing Beta Reputation System), we can provide resistance to collusion. We conclude with a detailed discussion of the value of our work for a wide range of environments, including a variety of multiagent systems and real-world settings.

Acknowledgements

First and foremost, I must thank my wife, Victoria Larke, for the astonishing amount of support she has provided in this process (and in the rest of life, too!) Embarking on a project such as this, and at this stage, is far more than an academic decision; it impacts every aspect of life for the entire family. In the most literal sense, this work could never have come to fruition without her support and encouragement, and her willingness to carry so much of the load for our family, so that I could devote time to this project. Of course, Vicky means far more to me than this—as I said in my Master's thesis, I can't possibly express how she has enriched my life, and I would not be the person that I am today without her. Victoria is an extraordinary person, and I am grateful every day that she chose to share her life with me.

A work such as this takes a significant amount of time, and a great deal of life 'happens' in the process. Most notable in our lives were the death of my father, Boyd; the birth of one beautiful daughter, Catherine; and the death of a second beautiful daughter, Elizabeth, who didn't quite make it into the world. I can't properly acknowledge those who have contributed to this work, and to our lives, without noting this context.

I need to thank Catherine, although she would not yet understand why. There is no way I could begin to describe what she has brought into our lives. Regarding this work, I will only say thank you for being so understanding on those many occasions when Daddy had to work on his 'pieces', and for bringing so much comfort to Mummy and Daddy in times of great sadness.

No words can adequately express my gratitude to my parents, Boyd and Laura Kerr. Their love and support, their strength of character, and their guidance, have shaped the person that I am, and the life that I am so fortunate to have. I marvel today at how they were able to raise me to have high expectations for what I might accomplish, but without imposing the slightest pressure to pursue any path other than the one that I felt would make me happy. In my life as a parent, I aspire to their example. I will return to say more about my father at the end of this section.

As I noted in my Master's thesis, John and Lorraine Larke are a wonderful counterexample to the in-law stereotype. I am thankful for their love and generosity of spirit; their willingness to help at times when the demands of life mount has made this process, and life in general, easier for us.

To the many people who reached out to us in difficult times, we will be forever grateful. Your love and concern have helped more than you could know. In particular, I wish to thank Graham and Isabel Stephens, who just days after suffering the devastating loss

of their newborn son Nathaniel, reached out to help us cope with our own loss. To this day, I stand in awe of their strength, compassion, and generosity. Although I have never met Isabel in person, our ongoing dialog by email has been central to my own healing. My gratitude is inexpressible.

Of course, I owe immense gratitude to my supervisor, Professor Robin Cohen. Robin is well known for being unmatched as a coach and mentor, and here I can only add my complete agreement, and my thanks. I have been incredibly fortunate to benefit from her dedication, and the amount of time and energy she puts into both helping her students achieve success, and ensuring the quality and value of our research. Those who have not worked directly with Robin may be less aware of her deep concern for the personal well-being of her students, which runs counter to the stereotype of the graduate supervisor. I am among the many students who have witnessed, and benefitted from this concern, and for this too, I am grateful.

I would also like to thank the members of my committee, Professor Beth Jewkes, Professor Audun Jøsang, Professor Kate Larson, and Professor Pascal Poupart, for their insights and careful consideration of my work. Professor Jøsang's research has provided inspiration for this project, and his words of encouragement have been greatly appreciated. I have been fortunate to take graduate classes with both Kate and Pascal, and I have benefited from their intelligence, dedication, and academic leadership. Kate has also been very supportive on a personal level, especially in our times of difficulty, and for this I am very grateful.

Universities are immense bureaucracies, and simple graduate students cannot navigate them successfully without a great deal of help from members of the administrative staff. I would particularly like to thank Jessica Miranda, Paula Roser, Wendy Rush, and Margaret Towell for their assistance and patience.

I would like to acknowledge the generous support of the Natural Sciences and Engineering Research Council of Canada (NSERC); this funding made it possible for me to entertain this project. I would also like to thank David Cheriton, both for his generous financial support, and for taking an interest in my work, and for his words of encouragement.

This work could not have taken place without flexibility and accommodation from my employer. I am deeply grateful to my academic Chairs, first Angela Zigras, and more recently Marianne Marando, for helping make this possible. Marianne has also gone above and beyond the call of duty to be supportive of our family during challenging times over the last few years; thank you.

My father, Boyd, was the only family member who read my Master's thesis in its entirety. Of course, I didn't expect my family members to read it—it was long and technical, and outside their spheres. My father, though, a retired police officer, had both the intellectual curiosity, and the deep love for me and interest in what I was doing, to read it cover to cover.

Dad died while I was completing the research for this project. I feel his loss tremendously, every day, but especially today.

My father did not have the good fortune of higher education, but he was a deep thinker, and had great wisdom. In the last weeks of his life, as cancer was ravaging his body, he wrote a document describing what he thought might await him, and all of us, when we die. Knowing Dad, he might have written it more to comfort us, than out of a need for self expression. In any case, I know that his words have brought great comfort to those who have read them, when faced with the mortality of their own loved ones. I think it would make my Dad happy to have his words published, and possibly read by those few souls brave enough to wade into this thesis. Thus, I include them here.

Theory on Death and Dying
Heaven and Hell

Boyd Walter Kerr

When I was a young boy I became aware that people died. When I learned that we were all going to die I started to have very serious concerns and an interest in the subject. This was partly because of curiosity but mostly because of fear.

When I started to Sunday school they talked about dying but not the actual act. Even to a child they painted some unrealistic pictures of Heaven and Hell. No one ever wanted to talk about death and dying, probably because they didn't know themselves. As I grew older I realized that I would have to come to my own conclusions about actually dying and the Hereafter.

As a teenager I had a discussion with my Mother (a very religious woman) about the Hereafter and she strongly believed in it. She knew it wasn't this idealistic place where everything was beautiful and the streets were paved with gold. She believed that when we died our souls left us and went to Heaven to remain with God in a place free of pain and sorrow. I told her that I thought that after death there was no Hereafter,

only nothingness. My Mother became very upset with me for thinking like that and told me it is not enough to lead a good life, but that I had to believe in the Hereafter too to be a Christian, and believe in going to Heaven as part of believing in God. This got me to thinking, not just about dying but about the Hereafter. I felt that people were so egocentric they had to believe in the Hereafter because they couldn't think of a world without them in it.

As I got older and saw a lot of dying and death I began to realize that the body had just been a shell that held the real person. The real person was their brainwaves, personality, their thoughts, how they lived. These things make up the soul that leaves the body at death. Where do they go? The Bible is quite vague on this, except for a few clues. It seems to say that if you are a good person, in action, thoughts and deeds, believe in God and Heaven your spirits and energy will join God in Heaven to do good. If you are a person who has not always lead a sin free life (like most of us), you can repent and turn your life around in thought and deed and believe in God. Your soul, spirit and energy can also go to Heaven and work in harmony with God, to do good deeds.

It is not difficult to believe in God, prayer and the Hereafter if you notice all the unexplained things that happen in our lives every day. We can only guess, what our souls, spirit and energy are. Because we cannot see our souls, spirit and energy it makes it difficult to believe that they are there. We know there is all kinds of energy that cannot be seen but has been proven to be there. I believe this energy is forever and ever, and like the universe, goes on for eternity. It is a very small step to believe that the soul, spirit and energy of man joins a much larger power, in a place called Heaven and joins together to do things with other souls we have known and loved.

The Bible is also very vague about Hell. It would lead us to believe that people who are committed to a lifetime of sin and never repent are destined to go to Hell. Their bad spirits and energy will join with Satan and continue on to do bad deeds, but their spirits will clash and there will be disorder and disruption, causing pain and sorrow to themselves and everything they influence.

Now, what will we find in Heaven when we cross over? The Bible has been vague there as well. I believe we will find the spirits of our loved ones, people we knew and people we would like to have known. They could be in the form of the person we once knew or some beautiful energy form. We will all join our spirits and energy with God's and do good deeds on earth and in Heaven. Our combined energy and force will be on a mission to persuade people to lead better lives, think better thoughts, do better deeds and by doing so rid the world of Satan and all his followers. I am sorry the biggest mystery of life is death. The answer can never be given to the living.

Whatever God has in store for me, I am not afraid of crossing over and joining all those who have gone before me, and starting the next part of my existence. I will be there to watch over you and meet you when you cross over.

Dedication

For Catherine, who brings indescribable joy to our lives.

For Elizabeth, who couldn't stay with us, but who fills our hearts with so much love.

For the little boy and little girl whose arrival we anticipate with such great hope.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

In the field of Artificial Intelligence, the term *agent* refers to a piece of software that performs actions on behalf of a user, typically with a high degree of autonomy [78]. The area of *multiagent systems* is concerned with environments where multiple, independent entities (primarily agents) interact with one another [67, 78].

While multiagent systems are often classified as either cooperative (where agents work together towards a common goal) or non-cooperative/competitive (where self-interested agents pursue their own agendas, possibly at the expense of other agents) [67], the reality is often more complex. For example, electronic marketplaces (composed of agents that buy and sell goods) are prominent examples of multiagent systems. Such environments might be classified as non-cooperative—each agent is a self-interested utility maximizer—but each agent also depends on other agents, its buying/selling partners, to achieve its goals. In such a scenario, being cooperative (in the form of behaving honestly) may be critical to success—honest agents may be more likely to find trading partners in the future. (While we focus on agent-oriented scenarios here, the same discussion often applies to human actors.)

In multiagent systems such as the one described above, an agent's success may also depend to a large degree on the reliability or trustworthiness of the agents with whom it chooses to interact. In the marketplace example, success may hinge on trading with reliable agents. For this reason, there has been considerable research in the area of *trust and reputation*. Trust and reputation systems (TRSes) are tools that are intended to aid agents in selecting trustworthy partners. These systems typically provide means for an agent to identify potential partners who are reliable, and/or to avoid agents who are expected to be unreliable.

While TRSes are intended to provide protection from dishonest or unreliable agents, it is often the case that malicious agents can bypass those protections. In earlier work [37], we identified a catalog of theoretical vulnerabilities in existing TRSes, weaknesses that can allow dishonest agents to cheat other agents without the TRS preventing or punishing the action. Such vulnerabilities are of grave concern, potentially rendering TRSes ineffective or even counterproductive. In the early chapters of this thesis, we investigate the feasibility of attacks that prey on such vulnerabilities.

In Chapter 2, we begin by examining the principal approaches taken by TRS designers, and considering a number of noteworthy proposals. (We also explore numerous other fields related to the work presented in this document.)

Then, beginning in Chapter 3, we embark on an experimental investigation of vulnerabilities in TRSes. In Chapter 3, we develop TREET, a testbed for experimentation and evaluation of TRSes. Such a testbed is a critical piece of infrastructure in exploring the security (or lack thereof) of TRSes, and TREET is unique in its ability to allow a wide range of general-purpose investigation. TREET provides a flexible simulated marketplace environment, suitable for conducting a wide range of trust and reputation experiments. Agents (making use of arbitrary behaviours/strategies) and TRSes (from a wide range of formulations) can be implemented in TREET, and their behaviour/performance can be evaluated under a variety of conditions.

Using TREET, in Chapter 4 we develop agents that make use of malicious strategies, designed to take advantage of vulnerabilities commonly present in TRSes. A variety of such strategies are tested, against a number of noteworthy TRSes; the malicious agents are overwhelmingly successful. These experiments reveal both the pervasiveness and the profoundness of vulnerabilities, and the practicality of attacks against them.

Of the attacks demonstrated in Chapter 4, of particular interest are those that are *collusive* in nature—strategies executed by a group of multiple agents working in cooperation with one another (a *coalition*) to circumvent the system. The issues of cooperation, coalitions, and collusion are especially interesting, for two reasons:

- First, vulnerability to coalitions is widespread in TRSes, as has been frequently noted by trust and reputation researchers (e.g., [4, 13, 31]). While some methods have been identified to counter other vulnerabilities, the problems posed by coalitions have been largely unsolved.
- Second, coalitions present issues for many domains, far beyond trust and reputation. (We discuss examples later in this chapter.) Tools to cope with these issues may have value in a variety of fields.

For these reasons, we focus our attention on these issues for the remainder of this document.

Cooperation is a complex phenomenon. In situations like the marketplace example described above, agents may display a high degree of mutually beneficial behaviour, despite being autonomous and independent; such behaviour may derive from agents' strategic decisions, as a consequence of tools designed to induce honest behaviour (e.g., TRSes), and/or from the fact that agents have been designed to be intrinsically trustworthy. Such behaviour is commonly observed, and is often desirable.

Cooperation between agents may go further, however. Beyond the independent-but-mutually-beneficial activity described above, agents may seek to coordinate their activities in some way, with the expectation that coordinated effort will further their goals (whether shared or independent). Coordinated activities may seek to increase agents' scores in a game, enhance profits, improve competitive positions, provide protection from other agents, damage competitors, etc., depending on the scenario. We refer to a group of agents working together to increase the benefits they receive as a *coalition*.[1]

There are (at least) three ways in which coalitions might aid one another, and in so doing increase the benefits accrued by each:

1. Coalition members might simply favour each other (for example, by selecting each other as partners more than outsiders, or by engaging in acts that have aid each other);

2. Coalition members might follow strategies/tactics of coordinated actions, which increase rewards to the group (for example, executing 'plays' in sporting events or games);

3. Coalition members might share information with one another, which might otherwise be unavailable (for example, insider trading, or revealing the cards that are held during a poker game).

[1] A coalition is often defined in the literature as simply a group of agents (particularly a group of agents that are cooperating in some way), without further specification [2, 3, 14, 44, 21, 53, 67, 82].

In some cases, definitions are more elaborate and domain specific. For example, in the study of coalition formation, a full specification for a coalition might include details such as the resources possessed and how those resources might be allocated amongst different agents. The issue of what a coalition *is*, however, is still ultimately 'a subset of agents' [66].

A definition of *coalition formation* (and implicitly of a coalition) which is appropriate for our problem: "Coalition formation (CF) is the coming together of a number of distinct, autonomous agents in order to increase their individual gains by collaborating." [12]

(In a coalition of rational self-interested agents, it should be the case that each entity expects to increase his own, individual benefit by participating in the coalition. It may not be the case, however, that each coalition member is *observed* to directly benefit—some members may be seen to profit greatly, while others are not. We make the common *transferable utility assumption* [67], however: that members of a coalition are free to share the gains from their activity, in this case outside the system (i.e., unobserved).)

Coordinated activity by a team may be benign, or even desirable. For example, consider a scenario where each agent is assigned large objects that it must move from one location to another. While there is no requirement that agents form teams (and no need to 'register' teams with any authority), teams of agents may be able to move the objects much more efficiently than individual agents, benefitting all agents involved. Further, there is no harm caused to others in the environment by agents coordinating their activities. A similar example is the 'neighbourhood watch' program, where individuals cooperate for their mutual security, but no harm is done to others.

In many other scenarios, however, agents may cooperate to further their own interests, despite the fact that this may be unwelcome or forbidden. For example, agents may cooperate in a game such as poker (which is intended to be played individually) to increase their winnings, at the expense of other players. Such harmful activities are often considered forms of cheating; we refer to them specifically as *collusion*.

Whether the activity of coalitions is seen to be benign or harmful, welcome or forbidden, will vary from domain to domain. Similarly, the steps one might hope to take—for example, to encourage or prevent coalitions—are also likely to be domain specific. Capabilities that are likely to be valuable in many domains, however, are the ability to *detect* the presence of teams, and/or to *identify* team members.[2] One might, for example, need to allocate security resources, and a knowledge of the the presence of 'neighbourhood watch' cells might make this allocation more effective. In contrast, one might wish to know which agents are colluding in a game, so that they can be penalized, expelled, etc. (We discuss more detailed examples later in this document.)

The goal of later chapters in this document is to establish techniques to detect the presence of coalitions, and identify team membership. Our aim is to develop methods that are broadly applicable to a range of activities and scenarios. It is worth noting that identifying teams is likely to be more difficult (and potentially more valuable) where such teams are unwelcome; colluding agents are unlikely to advertise their membership

[2]For brevity, we informally use the term 'detect' to refer to both the detection of coalitions and the identification of members, together. Where the distinction is important, we note it specifically.

in a team, and may actively seek to conceal their cooperation. Our work is particularly interested in such scenarios.

One can envision many real-world situations where detection of coalition activity, and identification of coalition members, is important. Notable examples include:

- As discussed above, collusion is a major problem for trust and reputation systems, and our inspiration for conducting this research. Trust and reputation thus serves as the example domain in which we apply our techniques in this document. Two forms of collusion are well-known here [13]:
  - ***Ballot-stuffing*** occurs when coalition agents give false positive reviews to their teammates, in order to inflate the reputations of the recipients. This inflated reputation is then used to induce other (outsider) agents to select the agent with the inflated reputation, instead of a competitor.
  - ***Bad-mouthing*** occurs when coalition agents give false negative reviews to competitors of teammates, in order to damage the reputation of the competitors. By doing so, the coalition again hopes to increase the likelihood that a coalition member will be selected, rather than a competitor.
- 'Shilling' and 'Astroturfing', activities designed to create the (false) impression of widespread public support (or opposition) for a position, product, etc. In many online communities, participants rely on the opinions of others when attempting to choose (or avoid) products to purchase, politicians to support, etc. False opinions offered by coalition members attempt to influence purchasers/voters/etc. into taking the desired action [26, 35]. While we can observe the postings/reviews, we cannot directly observe whether the posting or the writer is legitimate.

  This activity may occur in settings where a formal ratings systems is used, such as Amazon or TripAdvisor. In this context, the rating system is similar to a reputation system, and the 'shilling' activity is similar to ballot stuffing. The term astroturfing, however, refers more broadly to forum messages, blog postings, free-form textual reviews, etc.
- Insurgent activity in a military setting, or terrorism. Battlefield scenarios have been examined by recent work by artificial intelligence researchers in behaviour and plan recognition [68, 71]. Members of such a coalition may attempt to 'blend in' with the population, so we cannot directly observe group membership.

- In games such as 'first-person shooters' (FPSes) and 'massively-multiplayer online role-playing games' (MMORPGes), individual players may join into teams to gain competitive advantage over others. These venues may share similarities to the military scenarios noted above. In some games, teams are formal constructs, and no detection would be necessary. In others, however, no such recognized team structure exists.

- Collusion in games, particularly forms of gambling such as poker, is a significant problem. In fact, given the boom in online gambling, such cheating has been reported by the mainstream media (e.g., [33]).

- Within an economy, certain businesses may cooperate in order to further the goals of each. Some forms of such cooperation may be desirable (e.g., strong relationships between suppliers and purchasers), while others may be illegal (e.g., anti-competitive behaviour such as price fixing).

- One can conceive of applications in cooperative multiagent systems as well. For example, cooperating agents (e.g., robots) may have very limited communication capacity. It may be useful for an agent to be able to identify those other agents that are coordinating to accomplish a task; this may inform both its choice of action (which may depend on what the coalition of agents is doing) and its choice of potential partners (which may depend on the availability of agents, and hence on their membership in the team).

This list illustrates both the richness, and the real-world importance of the issue. Note that we do not prescribe any particular solution based on the identification; domain expertise is likely required to determine an appropriate course of action. One might seek, for example, to penalize coalitions, to 'undo' the effects of collaboration, to expel coalition members from the system, to shift resource allocations to take into account the activities of coalitions, etc.

While considering this range of scenarios, it is worthwhile to note certain features of the problem. In certain scenarios, while group membership is unknown, both the goal and the tactics might be understood in advance. If we have a complete set of possible tactics (a *plan library*), the problem of identifying teams might make use of multiagent plan recognition techniques (e.g., [68, 70, 72]). These proposals consist largely of pattern matching the actions of individuals against the known libraries of team plans. When the actions of a subset of agents provide a strong match against a plan in the library, we can infer that the agents constitute a coalition.

In many cases, however (including the types of scenarios with which we are primarily concerned) we cannot presume possession of a comprehensive known plan library. This issue was highlighted during the investigation of security vulnerabilities documented in Chapter 4. During our research, we uncovered new attacks, highlighting the difficulty of developing a comprehensive library, and similarly, the danger in depending on the fact that a library is complete. Indeed, passing familiarity with news reports on scams and fraud, on computer crime and security breaches, etc., reveals an astonishing ability for adversaries to find new strategies to prey on the unwary—the strategy space is so large that it makes potential attacks difficult to foresee. When unknown plans exist, existing multiagent plan recognition techniques are not directly applicable.

Other distinguishing characteristics of our scenarios should be noted. First, we can observe each individual identity in the environment (although we might not know who is actually controlling the identity, e.g., in the case of a user account). Second, while certain actions are observable, others are not. In particular, we have no access to communications between colluding parties, nor knowledge of their sharing of resources or benefits outside the system. (This has the consequence that our detection methods must be based on points 1 and 2 from the list presented earlier in this chapter (favouring one another, and engaging in coordinated efforts); the third (sharing information) is not part of our analysis.)

Given this understanding of the problem, in Chapter 5 we explore the nature of cooperation, arriving at the conclusion that benefit is the defining feature of coalitions, and that *similarity* of benefit is a powerful indicator of coalition activity. We introduce a *benefit space* representation of agents, where agents are mapped into the high-dimensional space according to how they benefit other agents. This representation is not domain specific, and is likely to be applicable to a wide variety of applications. Clustering can then be used in benefit space to identify groups of agents that are similar in terms of who they are benefiting. The clusters obtained constitute a set of *candidate coalitions*; we then apply a statistical technique to characterize those clusters, determining which of them constitute actual coalitions. The steps of mapping to benefit space, clustering, and then characterizing constitute our basic detection algorithm. This algorithm is evaluated using the TREET platform. Groups pursuing collusive strategies, in a variety of configurations, are embedded into larger populations. A wide variety of tests, exploring a number of issues and parameters, are used to demonstrate the accuracy of the method.

While the algorithm introduced in Chapter 5 achieves impressive results, there is still room for improvement, and still cases which prove challenging. In Chapter 6, we introduce a number of refinements that improve the accuracy and robustness of the al-

gorithm. Included here are: a recursive method of refinement, which enhances the algorithm's ability to separate multiple coalitions from one another; an improved method for determining the appropriate number of clusters to choose, based on our own characterization method; a means of applying multiple clustering methods in parallel, and being able to automatically choose the best result; and finally, an improved characterization method, which improves upon the one introduced in Chapter 5. Again, the refinements are subjected to a variety of tests, demonstrating significant improvement over the basic algorithm from Chapter 5.

In Chapter 7, we introduce the time dimension to our analysis. First, we investigate the speed of convergence using our technique, revealing that detection accuracy is high even when limited information has been accumulated. Then, we explore the important issue of dynamic coalitions. In many domains, coalitions are likely to be unstable, and membership may change often; we would like to be able to identify coalition members accurately, despite this. We address this issue by introducing a 'forgetting factor' as data is accumulated. Our tests show this approach to be effective. Finally, we incorporate the timing and situation in which transactions occur into our analysis, creating a measure of benefit modelled after the TF-IDF ('term frequency–inverse document frequency') measure [64] used in information retrieval. This measure improves detection accuracy in some circumstances, over the 'raw' benefit measures used earlier.

In Chapter 8 we return to the problem that originally inspired our work: the vulnerability of TRSes to collusion. We introduce an enhanced version of the well-known Beta Reputation System [29], in which we apply our coalition detection techniques, and take corrective action when coalitions are detected. This approach proves extremely effective, neutralizing and even punishing collusive activity.

This work constitutes an accomplishment of critical value towards addressing an issue that, despite its importance, has seen very little progress to date. In Chapter 9 we conclude with a discussion of the applicability of this work, and of future research directions.

Our work represents a major step towards addressing the issues of coalitions and collusion in multiagent systems. In contrast to research in areas such as coalition formation (which considers the decision-making of coalition members from an economic perspective), we instead address the challenging problem of coalition detection, presenting what we believe to be the first broadly-applicable technique to accomplish this. Given the lack of real-world datasets with labelled colluders, we offer an extensive set of tests using the TREET testbed to validate our method. We introduce a number of refinements, which further improve the accuracy and robustness of our method. We are able to demonstrate

how our work can be of use in a specific scenario (in this example, trust and reputation systems) in order to address domain-specific issues stemming from coalitions and collusion. In all, however, our methods have broad applicability for many scenarios in which the possible presence of unknown coalitions within a larger population may be of concern, including a variety of multiagent systems and real-world settings. As such, our work offers contributions for a wide range of researchers.

Chapter 2

Related Work

Although our problem has received little direct study, there is a broad range of areas that have some relation to our work.

Trust and reputation systems are both the inspiration and the example application for our work, and so we begin this chapter with an overview of that field. Afterward, we highlight a number of other areas that share some relationship with our topic.

2.1 Trust and Reputation

Trust and reputation systems (TRSes) in multiagent systems aim to help agents choose trustworthy partners and/or avoid untrustworthy ones. Typically, agents provide reviews of their experiences with other participants; when deciding whether or not to trust a potential partner, an agent can make use of the information in these reviews.

In earlier work, we shed light on the importance of security in trust and reputation systems [38] and identified a number of theoretical security vulnerabilities that are pervasive in TRSes [37]. (These vulnerabilities are explored experimentally in Chapter 4.) In particular, vulnerability to collusive attacks is ubiquitous, and serious. There is widespread acknowledgement by researchers of the vulnerability of trust and reputation proposals to coalitions/collusion (e.g., [4, 13, 31]). While efforts are made to cope with unreliable reviews, systems are often susceptible to two well-known forms of collusion [13]:

- ***Ballot-stuffing***, where coalition agents give false positive reviews to their teammates, in order to inflate the reputations of the recipients.

- ***Bad-mouthing***, where coalition agents give false negative reviews to competitors, to damage the targets' reputation.

The goal of both of these attacks is to improve team members' reputations relative to competitors, and thus improve the members' chances of being selected by other agents.

Unfortunately, trust and reputation researchers have made little progress against this problem. Because of the importance of collusion to TRSes, and because such systems have been well-studied, we use them here as the example domain in which we demonstrate our techniques.

In this section, we survey the range of approaches taken by TRS designers, and examine a number of specific proposals, with particular attention given to the issues presented by coalitions.

2.1.1 Direct experience

In one category of systems, described as *direct experience* models [63], agents rely solely on their own past interactions with a target agent in evaluating its likely future trustworthiness. Examples of such systems include the work of Marsh [46], Griffiths [22], and Tran and Cohen [75, 76].

The work of Marsh [46] was seminal in the area of trust and reputation, proposing that agents might estimate the trustworthiness of others, based on their own experience of the other agents' behaviour. In Marsh's model, an agent a will cooperate with another agent b if a's trust value for b exceeds a certain threshold. After the experience (should a choose to cooperate with b), a will update his view of b's trustworthiness based on the new information. Marsh also proposed that such modelling might be situation dependent—that a may view b's trustworthiness differently, depending on the nature of the task. For example, a might trust b for minor tasks, but not enough to cooperate on more important tasks.

Griffiths [22] expanded on Marsh's model by decomposing trust into multiple dimensions. For example, a may have a high degree of trust that b will deliver goods on time, but may have a low degree of trust in the quality of product that will be delivered.

The Tran and Cohen model [75, 76] makes use of reinforcement learning, as agents seek to learn over time who is trustworthy. Each agent maintains a set of expected outcomes for each possible action (here, choice of product and partner), based on past

experience. An agent chooses from among the available actions so as to maximize the expected value. After an action is taken, the real outcome is used to update the expected outcome for that action, before the next such choice is made. With high probability, agents will *exploit* the marketplace, choosing partners from among the agents that are believed to be trustworthy. Occasionally, however, an agent will choose to *explore* the market instead, choosing an agent of unknown trustworthiness, with the goal of identifying new trustworthy partners. Over time, the buyer will learn which agents can (and cannot) be trusted, and which ones give the best value for any given product.

Direct experience systems have a number of advantages; most relevant here is that they are essentially impervious to forms of collusion like those we have noted. Because an agent relies only on its own direct experience when evaluating a potential partner, other members of a coalition cannot directly impact that evaluation. Direct experience models are relatively uncommon, however, because they suffer a number of important limitations. In particular, agents in such a system are much slower to learn who is (un)trustworthy, because they do not have the benefit of other agents' experience. Accordingly, these systems are largely outside our focus.

2.1.2 Witness information

In comparison, the majority of trust and reputation proposals are *witness information* systems [63], where agents incorporate information provided by others (referred to as *reviewers, witnesses, recommenders,* etc.) when evaluating the trustworthiness of a target agent (e.g, using a probabilistic model). Representative examples of such systems include the work of Yu and Singh [83], the Beta Reputation System [29], and Travos [73].

Typically, agents in witness information systems can learn the trustworthiness of potential partners much faster than in direct information systems, because they have a much greater pool of experience upon which to draw. Unfortunately, they are also quite vulnerable to inaccurate information provided by reviewers.

Reviews may be unreliable for a number of benign reasons: for example, the reviewer may have limited experience itself, it may have very different tastes or expectations than the agent requesting reviews, etc. Reviewers may also provide intentionally misleading reviews, whether for strategic reasons, or simply out of malice. Attempts have been made to compensate for the varying credibility of reviews, as discussed below.

The Yu and Singh model [83] makes use of referral networks in which agents, when evaluating a target agent's trustworthiness, can request recommendations from neigh-

bouring agents. Each agent has a set of other agents (its *acquaintances*) for which it models trustworthiness, based on its own experiences of those agents' behaviour. When a request is made to an agent a for a recommendation for target agent t, a may answer from its own experience (if t is an acquaintance), or may forward the request to a neighbour. The Yu and Singh model is novel in its use of the Dempster-Shafer theory of evidence to evaluate the likelihood that a target agent is trustworthy, which allows agents to distinguish between the case where an agent t is known to be untrustworthy, and the case where t is simply unknown to be trustworthy (due to lack of evidence).

The Beta Reputation System (BRS) [29] makes use of the well-known beta probability distribution to estimate the probability that a target agent t will be trustworthy. Given counts of the previous number of successes r (i.e., honest actions) and failures s (dishonest actions) by target agent t, the mean of the resulting beta distribution estimates the probability that t will be honest on future transactions:

$$P(\text{honesty}) = \frac{r+1}{r+s+2} \tag{2.1}$$

(A transaction that is partially satisfactory might be reflected by using fractional 'counts' for r and s. For example, if a transaction was 70% satisfactory, this may be reflected by adding 0.7 to the r total, and 0.3 to s.)

To incorporate the experience of others, the evaluating agent a can request reviews of t from other agents with which a has relationships; each review consists of the counts r and s that the reviewing agent has accumulated. The simplest way of incorporating these reviews is to simply sum the r and s values from each of the reviews received (along with a's own counts).

Noting that the reviews received may not be entirely reliable, the authors propose a system called *reputation discounting*, where reviews received by a from b are adjusted depending on a's experience of b's trustworthiness. A subsequent proposal [79] introduces a system where potentially-unreliable reviews are filtered out based on statistical properties. When considering an entire set of reviews, those reviews that are extremely positive or negative (relative to the bulk of reviews) can be considered suspicious, and are disregarded.

The TRAVOS system follows BRS's example, making use of the beta distribution in a very similar manner. To this foundation, TRAVOS introduces two features. First, agents only look to other agents for reviews of a target agent t if they do not have enough direct

experience to judge t's trustworthiness themselves. Second, TRAVOS uses a different method of evaluating the trustworthiness of reviews. For each reviewing agent b, agent a maintains a record of b's past accuracy. When a receives a new review of target t from b, the weight given to that review is determined by the accuracy of the reviews b provided in the past. After a has completed its interaction with t, it updates its records of b's accuracy based on the result.

As noted, these proposals provide methods of dealing with potentially unreliable reviews. However, these defences seem intended primarily to cope with unreliable reviewers that act independently, rather than as coordinated teams. For example, the later BRS proposal [79] acknowledges its inability to cope with large proportions of unreliable reviews (even from independent reviewers); coordinated teams might overwhelm the system. Under TRAVOS, an accurate recommendation is one which corresponds closely to the ultimate behaviour of the target; in the case of ballot-stuffing, for example, both the recommendation and the behaviour of the target may be positive, which actually serves to increase the credibility of the ballot-stuffer. While these proposed defences are quite effective for their intended purposes, the do not stretch well to the coalition problem we now address.

2.1.3 Mechanism design

The classes of trust and reputation systems described above are essentially predictive in nature; whether or not they take an explicitly probabilistic perspective, they attempt to project the future behaviour of agents based on their past behaviour. In contrast, a third, and much smaller, group of proposals attempt not to evaluate the trustworthiness of agents, but rather to use mechanisms to elicit honest behaviour in a marketplace trust scenario. One example is the work of Braynov and Sandholm [7, 8] (which attempts to set the rules of the marketplace so that an agent's profit-maximizing strategy is to honestly declare its trustworthiness).

A mechanism that ensures trustworthiness would be ideal—if no agent had reason to be dishonest, then predictive models would be unneeded. Unfortunately, the mechanisms proposed, while provably correct, depend on assumptions and conditions that make them inapplicable for real-world use. For example, the Braynov and Sandholm work requires that agents know each others' cost functions, a highly unrealistic prospect. Further, this work does not ensure that agents behave in an honest and trustworthy manner,

only that they honestly declare how trustworthy they are; it is difficult to envision this being applied in real marketplaces. Further, such proposals typically make no mention of collusion, giving no reason to believe that they are resistant.

Important work on the problem of ensuring honest reviews has been done by Jurca and Faltings. In early work [30], a mechanism is proposed where a reviewer is paid a small sum for a review, only if his review matches the previous review (by a different agent) of the same good/vendor. The idea is that if an agent gives honest reviews, it is more likely that its review will match the previous one, increasing the payment expectation. Unfortunately, the authors note that the system is subject to collusion: if many agents give dishonest reviews, then dishonesty yields the highest expected payment.

In a later proposal [31], a collusion resistant scheme is developed. Similarly to the earlier work, agents are paid for reviews in some cases. Here, however, payments are only made if the number of positive reviews so far is a specific value. For an agent to be paid for a positive report, the number of positive reviews so far (from all agents) must be the number with the maximum probability given that you actually had a positive experience (relative to the probability of having that number of positive reviews if you had actually had a negative experience). It is suggested that the system is collusion resistant because a coalition would find it problematic to time their reviews in order to be paid, without complete information about the reviews given by non-coalition agents. Under such circumstances, an agent may not be paid for most reviews, but still maximizes its expectation by reviewing honestly. The authors go on to consider a number of scenarios, and while the system is not perfectly collusion-proof, it does provide significant protection when a portion of the population tells the truth.

While this work represents a significant step, it has limitations. First, no consideration seems to have been given to collusion involving reviewees, in addition to reviewers (which is a standard part of many collusive attacks). It requires assumptions such as that the quality of the product being reviewed remains constant; in a case where reviewees are part of the coalition, they can manipulate product quality, undermining the system. Further, it is not budget balanced, requiring input of funds to work.

Earlier work of ours, on the Trunits systems [37, 43], has a mechanistic effect. The original Trunits system [37] (subsequently labelled 'Basic Trunits', to distinguish it from a later proposal) introduced the idea that trust could be represented using numerical units, similar to a currency. The quantity of trunits possessed reflects a seller's trustworthiness. A seller can only engage in a sale if she is considered sufficiently trustworthy (i.e., has sufficient trunits). If the seller enters into the sale, then the required quantity of trunits is

placed in escrow, pending completion of the sale and a review by the buyer. If the review is negative, then the seller loses the trunits, and hence, the seller's ability to engage in future sales is reduced. If the review is positive, the trunits are returned to the seller, along with a reward of a small additional number of trunits, increasing the seller's ability to engage in future sales. The effect on future sales provides the incentive for honesty.

Basic Trunits faces certain issues. In particular, there is the startup problem: how do agents obtain an initial quantity of trunits? (If they are provided with an initial sum when they create their account, a dishonest agent can simply open many accounts, and cheat freely with the trunits.) Further, when an agent possesses more trunits than she needs (e.g., when an operating surplus is acquired, or when an agent plans to exit the marketplace), then she can use the unneeded trunits to cheat without consequence.

Commodity Trunits [43] was introduced to address these problems. Under this system, trunits are tradable: they can be bought and sold on an open market. This addresses the 'start-up problem' (because agents can purchase trunits when needed), and the 'surplus trunit' problem (because the system is designed so that selling trunits is more profitable than using them to cheat.) However, it requires a trunit marketplace to be created and maintained.

Unfortunately, both of these systems are vulnerable to collusion. Ballot-stuffing can be used to generate additional trunits, which can be used to engage in sales (or under Commodity Trunits, sold)[1]. Bad-mouthing can remove potential competitors by reducing their trunit balances (or cost them money, under Commodity Trunits).

2.2 Coalition formation and stability

An area of long-standing and continuous research (e.g., [18]), and one with obvious topical relationship to our work, is that of coalition formation and stability within the field of multiagent systems. The problem of coalition identification differs fundamentally in nature from our own, however.

The issues of coalition formation and stability are often approached from a game-theoretic perspective [53, 66], exploring the conditions under which coalitions form, algorithms for formation, and requirements for a coalition to persist.

[1] Commodity Trunits can, in fact, provide some protection against ballot-stuffing, but this requires the reward for honesty to be set very low. This reduces the incentive for honesty, and makes it difficult for sellers to grow their business without the input of large quantities of capital to acquire trunits.

Researchers consider coalitional games, where outcomes consist of the coalition(s) that form, and the joint actions taken by the coalition(s) as a whole. More directly related to our work, researchers also investigate coalition activity within non-cooperative games: self-interested agents may find it advantageous to cooperate with other such agents, in the pursuit of their own interests [66].

The solution concepts fundamental to much work in coalition formation and stability are based in the idea that a coalition forms to improve the prospects of its members; a coalition is stable if no agent has an incentive to defect based on the utility it receives [44]. There are a variety of solution/stabililty concepts used to evaluate coalitions, including [53]:

- *Nash Equilibrium*, a configuration in which each agent is maximizing its own utility, given the strategies of the other agents.
- *The Core* (a coalitional analog to Nash Equilibrium), where no (sub)coalition can deviate from a coalition and improve the total payout to the subcoalition's members.
- *The Stable Set*, a set of payoff configurations Y for a coalition, where no payoff configuration within Y makes every member better off than another configuration in Y, and for any configuration outside Y, there is a configuration in Y where every member is better off.
- *The Bargaining Set*, where for any unsatisfied coalition member x who makes an argument claiming that all of the coalition members are better off if they exclude y, y can make a counter argument claiming that by excluding x instead, the remaining members would be better off than by excluding y.
- *The Kernel*, where for any argument agent x makes that agent y is receiving too great a payout (and hence, by kicking out y, x could claim a greater payout), y can make a similar counterargument against x.

While this list is not exhaustive, it illustrates the nature of work in this field. One might consider that, in order to detect coalitions, one might look for sets of agents satisfying these concepts. Unfortunately, there are a number of requirements and common assumptions made in this work, which are not met in the scenarios with which we are concerned. Examples include:

- It is commonly assumed that the capabilities of agents are known to one another [66]. This is not necessarily true, for example, in a marketplace (where a buyer may not know what quality of products (or even which products) a seller produces), or on a battlefield (where a combatant's capabilities may not be evident until they are demonstrated).

- It is commonly assumed that the value earned by a coalition does not depend on the actions of any agents outside the coalition [44]. This is clearly not the case when, for example, coalition agents are trading with non-coalition agents.

- The definitions above show that we often need information about the distribution of payouts within a coalition to apply these concepts. In the scenarios with which we are concerned, such payments outside the playing field are explicitly unobservable. Moreover, the 'payout' may come in non-monetary terms (e.g., increased reputation, improved security, etc.) Unfortunately, we have no information about agents' utility functions, so we cannot evaluate these 'payouts'.

- Because it is difficult to evaluate benefits such as reputation and improved security, it is difficult to apply solution concepts requiring us to, for example, determine the value of a coalition that excludes a particular agent.

While these game theoretic principles may not be applicable as we focus on detection of coalitions, they may be useful in future work: for example, moving past detection into the development of systems that take corrective action.

2.3 Multiagent Plan Recognition

Another area which appears, at first glance, to be closely related to our work is that of multiagent plan recognition. Work in this area considers scenarios where multiple agents are attempting to execute a joint plan; the actions of the agents can be observed, but the plan cannot. The goal is to infer the plan being executed from the observations. If the agents break into subteams in order to execute portions of the plan, it may also be desirable to detect the splitting and merging events, as well as the composition of the subteams.

Multiagent plan recognition follows closely on a large body of work in the area of (single agent) plan recognition (e.g., [10, 34]). The work of Kautz [34] develops a common theme: the observed actions of the subject are matched against a known library of

plans, to determine the best match. Here, *events* (both plans and actions) are hierarchically structured, with plans containing actions and subplans. The recognition task is to describe the end event (the 'root' of the plan tree) based on the observed events (actions). Kautz's approach is entirely logical, making use of first order predicate calculus and reasoning to reach conclusions from observed facts.

While there has been extensive work in (single agent) plan recognition, it has been noted (e.g., [71]) that there has been a limited quantity of work on multiagent plan recognition.

The work of Tambe [72] is a noteworthy example. In this paper, the author proposes that explicit use of team models enables more effective team tracking. Tambe notes that tracking using joint models has advantages over tracking each individual separately: it can reduce the search space in recognizing actions, and it can more easily cope with changes in team membership.

Tambe's RESC ('REal-time Situated Committments') approach, originally developed for tracking individuals, makes use of a runnable model of the target. Given a starting state, the model is run, and the prediction compared with the actual actions of the target. It may be the case that the action corresponds with multiple possible execution paths; in this case, one is chosen heuristically. If the later observation shows a deviation (i.e., that the wrong choice was made at this point), then a 'real-time repair' operation is executed, correcting for the error.

Applied to teams, RESC uses models consisting of both team state and team operators. The joint team state consists of a shared part (common to all participants, e.g., mission), and a divergent part (applicable to individuals, e.g., position). For ease of tracking, rather than monitoring the divergent components for each member individually, a single paradigmatic agent is selected to represent the team. If a team splits into subteams, a paradigmatic member is selected to represent each. Tracking then proceeds similarly to the single entity case, with one change: if a tracking failure occurs, it must be determined if the failure relates to the entire team, or to a subteam, before repair can take place.

While described somewhat differently, this method follows the same basic principle outlined above: a library of known plans (in this case, embodied in a model of the target) is searched (in this case, in a real-time, backtracking manner).

Later work in this area was conducted by Sukthankar and Sycara. In one proposal [68], the authors address plan recognition of a small combat team in a battlefield scenario. In this scenario, the identities and positions of team members are known, as

well as the positions of other environmental features. A library of all possible tactical behaviours is known. Two techniques are then used:

- Hand-authored spatial templates are created representing the relative positions of team members at some point in each tactic. These spatial templates can be scaled, rotated, etc. For efficient search, a randomized technique is used, RANSAC. A simple overview of how a single position is evaluated: 1) Choose two entities at random; 2) Find all templates that match the two entities, considering possible transforms; 3) For each matching template, calculate the expected positions of the other entities; 4) For each entity that is close (within some threshold) to its expected position, consider it a 'vote'. 5) The matching template (and hence, tactic) is the one with the most votes.

- In some situations (particularly, in two-person teams) spatial relationships are insufficient to recognize a behaviour. Here, a temporal component is added, considering the relative positions of the entities over a window of time. A hidden Markov model is then used to match these observations to a behaviour.

Subsequently, the authors consider the issue of dynamic teams, where teams split to perform subtasks, merge, etc. [71]. Here, traces of agent positions over time are used to identify behaviours. A known set of team behaviours is used to derive a set of movement constraints. The goal is to determine the behaviour of each team, and the assignment of each agent to a team, for each time step observed. Exhaustively matching all possible behaviours and team assignments against all agents in every time step would be infeasible. Instead, the authors start with a static state, and find potential teams by spacial arrangements alone, as described above. Then, the movement of agents over time are examined to confirm or discard the potential assignments.

The approach outlined above might best be described as behaviour recognition, rather than plan recognition. To identify plans, the authors use an approach derived from the hierarchical plan matching used in single-agent plan recognition [70]. This is essentially a search task, but the search space is reduced by pruning branches based on resource requirements, temporal dependencies, etc.

The authors also address something they call *policy recognition* [69]. Here, it is recognized that plan recognition based on long traces of activity (for example, using hidden Markov models) may be problematic: often plans are interrupted, reconsidered, etc. Instead, they seek to identify policies, which they describe as broad (over all possible states), but shallow (not detailed), as compared to plans (which are deep (looking far

into the future) and narrow). Here, they use a simple probabilistic model, assigning evidence from observed events to support known policies. They also use supervised learning (a support vector machine), using training data with known policy labels.

Existing multiagent plan recognition work, while important, focuses on different scenarios than those with which we are concerned. For example, such proposals typically assume that the membership of the team (or teams) is known, and seek to identify the plans used by the team (e.g., [32]), or how the team has organized itself internally to execute a plan. In our work, team membership is unknown, and discovering it is our goal. As outlined above, existing work typically presumes the existence of a model of the team or known library from which the agent(s) in question draw their plans; our work assumes that we have no model or knowledge of plans. Moreover, in the scenarios we target (e.g., marketplaces), there may be a very large number of participating agents; there may be zero, one, or many coalitions at work—big or small—within the larger population, and we have no advance knowledge of the numbers or of the configuration. In contrast, work in this area typically centres on a single (known) team, or environments where two (known) teams constitute the entire population.

2.4 Community finding/Social network analysis

An area which has received a great deal of attention is that of *community finding* in social networks. This work attempts to identify subsets of populations that constitute 'communities' unto themselves. The most prominent work in this area appears to be that of Girvan and Newman (e.g., [19, 51]). This work, which is representative of the area, is based on the observation that social networks consist of smaller groups (*communities*) that are densely connected; these communities are connected to the larger network by a small number of links. To identify communities, one can eliminate the (few) links to the outside network, leaving the connected communities intact. To identify the links to cut, the authors make use of a number of 'betweenness' measures, which reflect the role of each link. For example, to compute *edge betweenness*, the shortest path from every vertex to every other vertex is computed, and the number of times each link is traversed is counted. Links that connect communities to the outside network will be traversed far more often than links within a community.

It is tempting to think of coalition detection as an instance of the community finding problem: essentially, we are seeking to find groups of affiliated agents within a larger

population, based on their interactions. In reality, though, the problem of coalition detection differs in fundamental ways from that of community finding, making it difficult to apply proposals from that area to our problem. In fact, the properties that are fundamental to community finding (e.g., frequency of interaction, connectivity, etc.) can be of little use, or even entirely misleading, when attempting to identify coalitions. Recall that in our marketplace scenario, when coalitions engage in strategies such as bad-mouthing and ballot-stuffing, they are attempting to improve the reputations of member agents, relative to that of competitors. They do so in order to win additional sales, from agents outside own group. If ballot-stuffing or bad-mouthing is successful, a relatively small number of collusive actions may win many sales from outsiders. Here, the frequency of interaction will be high between members and non-members (perhaps even higher than *amongst* members); connectivity with outsiders will similarly be high. Measures such as these, which are so useful for community finding, provide little insight into coalition activity.

The difference is very clear when one considers bad-mouthing. Figure 2.1 depicts a simple situation where one coalition, consisting of agents A, B, and C is present. Two agents, coalition member C and non-member D, are offering the same product for sale, which agent E (also an outsider) seeks to purchase. E will evaluate the reputations of each agent in selecting whether to buy from C or D.

A and B, seeking to support their teammate C, engage in bad-mouthing: they give false negative reviews to D, in order to damage the reputation of C's competitor. Based on this, E is likely to choose C instead of D.

Note that the coalition members have (very successfully) engaged in collusive activity, *despite the fact that there is no interaction or connectivity between any of the coalition members*. This illustrates an essential difference between community finding and coalition detection: in community finding the key issue is who is interacting with whom, while in coalition detection (as explored in Chapter 5), the important issue is who is *benefitting* whom.

These properties are examined in greater detail in Section 5.2.

Palshikar and Apte present what they identify as a coalition detection method for stock market trading [54]. This work is probably best classified as community finding, however: they seek groups with high levels of interaction between them. Essentially, the authors seem to have made the assumption that in the context of stock trading, communities should be viewed as coalitions.

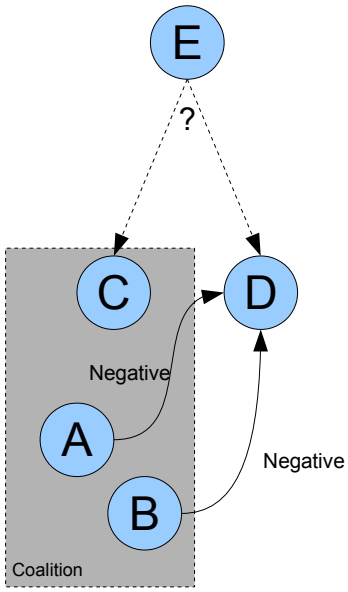

Figure 2.1: A bad-mouthing situation.

2.5 Recommender systems/Collaborative filtering

The research that is perhaps most similar to ours appears to have come in the field of *recommender systems* and *collaborative filtering*, an area with a long research history (e.g., [20, 57, 59]). Recommender systems aid users in making selections, by making recommendations based on the opinions of others. Recommender systems are commonly encountered on the internet today, providing recommendations on books, music, movies, etc. While such systems might make use of actual purchase data, most often they are based on the stated opinions of users: each user provides explicit opinions on a range of items, with these opinions being used to make future recommendations.

Two primary approaches are used [11]:

1. User-based algorithms find other users (*neighbours*) with similar opinions to the user in question (i.e., they have offered similar opinions on some of the same

items), and then find items that the neighbours have liked or disliked, under the assumption that the user might share the neighbours' tastes.

2. Item-based algorithms find, for each item, a set of similar items, based on their receipt of similar ratings. When a user expresses like or dislike for an item, the assumption is made that they may also like/dislike the similar items.

Clear similarities exist between recommender systems and reputation systems. In particular, both rely on the opinions of others to help an agent make selections. Similarly, both types of systems share vulnerability to malicious users that express false opinions in the hopes of manipulating others; further, in both cases malicious users will prefer to avoid detection.

Recently, much research has targeted the problem of *shilling*—the creation of false user profiles/accounts containing ratings intended to manipulate the results of the recommendation algorithms (e.g., [9, 11, 48, 49, 47, 50, 65, 80]). Two general types of attacks are noted: *push* attacks, intended to increase the recommendations of an item, and *nuke* attacks, intended to decrease recommendations. These correspond roughly to ballot-stuffing and bad-mouthing, respectively. There are important differences, however, stemming from differences in scenario, which we outline below.

In both reputation systems and recommender systems, agents make use of the opinions of others. These opinions are used in quite different ways, however. In a reputation system, an agent might make use of all of the ratings rendered by other users; alternatively, it might select or weight ratings based on a number of possible factors: its relationship with and confidence in the reviewers, the past accuracy of the reviewers, the statistical properties of the reviews (e.g., is the review an outlier from the other reviews), etc. In contrast, the recommendations of a recommender system are based on the reviews of those considered to have similar tastes to the user (i.e., those having rendered similar opinions). These are fundamentally different criteria. Note, too, that in a reputation system, it is often up to the individual agent to determine which reviews/reviewers to consider, and how to use them in the decision-making process. In contrast, recommender systems make the recommendations for the user, without the user needing to (or being able to) evaluate the different reviews.

These differences lead themselves to different attack strategies. To ballot-stuff or bad-mouth in a reputation system, a coalition might (for example): offer large numbers of reviews to sizeably increase reputation; create many new accounts for the sole purpose of making a single ballot-stuffing/bad-mouthing review; offer many honest reviews to

build credibility before engaging in ballot stuffing; recommend other teammates as reviewers (in a referral network) so that others might transitively trust the teammates. By comparison, an attacker in a recommender system seeks to build shill profiles that will be as similar to (honest) users as possible, so as to strongly impact their recommendations [48].

A shill profile will typically consist of ratings of the following items [50]:

- A *target item*, the item chosen to be pushed or nuked;
- *Filler items*, other items rated by the profile. Filler items/reviews are selected in an attempt to have strong influence (i.e., strong similarity) with users whom the attacker wishes to influence.

To avoid being obvious, filler items are typically rated randomly, using several strategies, including [50]:

- Random attacks, where each item is rated randomly (using a Gaussian distribution) around the overall mean rating across all items;
- Average attacks, where each item is rated randomly around the mean for that particular item;
- Bandwagon attacks, similar to random attacks but with certain popular items rated maximally, to increase correspondence with honest raters.

2.5.1 Combatting shills in recommender systems

We note several important contributions to countering the problem of shills.

Chirita et al. [11] present an early attempt at coping with this issue. Their approach makes use of statistical properties to identify shill profiles, so that they can be removed from the recommendation process. Among the measures they use are *number of prediction differences* (the number of changes in the predictions a system would make if the profile was removed from the system), standard deviation in a user's ratings, and degree of agreement with other users. They also construct a new measure, called *Rating Deviation from Mean Agreement* (RDMA). RDMA for a given user j is calculated as follows:

$$RDMA_j = \frac{\sum_{i=0}^{N_j} \frac{|r_{i,j} - Avg_i|}{NR_i}}{N_j}$$

a) for each product *i*, the deviation between the user's rating and the average rating is computed, and then divided by the total number of ratings for the product (with the goal that highly reviewed items, for which a single additional review will have little influence, contribute less to RDMA); b) the values for all products are averaged. Chirita et. al then propose a simple algorithm where accounts with very high (or low) values of these key metrics are labelled as shills. As noted in [48], this approach has some success in detecting attacks dense with profiles, but less effective in detecting small or sparse attacks.

Burke et al. [9] make use of similar statistics to Chirita et al., but instead of using an ad hoc algorithm, make use of supervised learning to detect shills. While this method achieves some success, and copes well with sparse/small attacks, it has obvious limitations. First, it requires sufficient labelled training data, which is difficult to acquire. Second, it fails to detect behaviours that may be valid attacks, but were not present in the training set. This is especially important where attackers are actively trying to disguise their activity.

Mehta et al. [48] take a different approach. They note that shill profiles (by construction) are similar to many real users; however, in contrast to the normal variability in the opinions of real users, shill profiles are *extremely* similar to one another. The authors make use of Principal Component Analysis (PCA). PCA reduces dimensionality by projecting data (in this case, profiles of opinions) to a low dimensional space, where the axes selected capture the maximum variation. By taking the first principal component, and selecting the variables with the lowest coefficients, one can select profiles with the lowest covariance with other users. These profiles are most likely to be shills. In experiments, by eliminating the top 5% of profiles, they achieved over 90% accuracy in eliminating shills. (We note that in this case, the only consequence in 'eliminating' a shill was that their opinion would not be used to generate recommendations. This has little consequence for the owner of the profile, so a false positive is not of grave concern. In our scenarios, where more serious action might be taken, 90% accuracy is probably inadequate.)

Sandvig et al. [65] propose the construction of a recommender system that is resistant to shills. Rather than making recommendations based directly on users' ratings, they instead use the *apriori* algorithm to generate *association rules*, sets of items that commonly are rated together. When a user has rated one item, recommendations are generated based on the association rules. This approach was shown to be fairly resistant to shills. However, because some items do not occur frequently enough with others to generate association rules, many items can never be recommended.

Resnick and Sami [58] also introduce a system intended to limit the effects of shills, which can be applied to existing recommender systems. In this proposal, each rater j has a reputation score, computed based on the degree to which future raters of products agree with the ratings given by j. The recommendations given by the system are a weighted average of the predictions given before j's rating is considered, and the prediction made when j's rating is incorporated; the weight given to the latter is dependent on j's reputation. The authors show that influence is maximized by giving accurate ratings, and that the impact of shills is limited, subject to certain assumptions (such as the use of an optimal recommendation system, and limits on the number of accounts that can be created).

Mehta and Nejdl [50] make use of a 'soft clustering' technique, *Probabilistic Latent Semantic Analysis* (PLSA) to cope with shills. PLSA is a Bayesian network technique using latent variables to explain co-occurence. In this case, when users have given many similar ratings to many of the same items, this coincidence of opinion will be seen as likely to have been generated by a common 'source', represented by a latent variable. PLSA has been used as a means of collaborative filtering; here, each latent variable might represent a shared interest or common community. (Note that this is a form of clustering; all of the profiles related to a latent variable constitute a cluster.) Used in this way, PLSA has been seen to be quite resistant to shills, although the reason for this has been debated.

The authors note, however, that this technique can also be used to detect shills: a latent variable representing a shared 'interest' might instead by a shill. Noting, again, that shill profiles have very little covariance in their ratings, they measure the 'tightness' of each cluster (using Mahalanobis distance) [50]. The cluster with the smallest distribution can be considered to consist of shills. Whether or not shills are present, or how many groups of shills, appear to be unanswered questions.

These approaches have important similarities to our work, in that they focus on detecting coordinated attacks (at times even using clustering). Indeed, there is much in this work that can inform our own. There are important differences, however. Some of those differences have been noted in the discussion above; we note others here:

- Leading approaches to detecting shills in recommender systems focus on the extreme consistency of composition (behaviour) of the shill profiles. Shill accounts are only effective if they are similar to real users; outliers will have little influence. In contrast, in the scenarios we target, members of a coalition may have behaviours that are very different from other users, and very different from other coalition members. For example, within a marketplace some coalition members

may actively ballot-stuff in large quantities, some may only sell products without giving reviews at all, and some may balance ballot-stuffing with other activities. In a battlefield scenario, some agents may be active attackers, while others take on supporting roles.

- To achieve similarity to real users, a shill profile often must have a significant number of filler items, which then form the basis for detection. In our scenarios, no such 'filler' is necessary; for example, an agent can offer a review (which might carry significant impact) without having made any other recommendations.

- Typically, each (shill) account can only rate a given item once. In reputation systems, one account might provide many reviews of another agent. Considering our broader range of scenarios, one agent might engage in many acts that benefit another agent.

- Similarly, a group of shill accounts typically targets only one (or perhaps a small number of) items; targeting too many items might reduce similarity to real users, and hence effectiveness. In contrast, within a marketplace members of a coalition might freely ballot-stuff for many of its members, while bad-mouthing others. (Further, this activity need not be symmetric or 'fair' in any way, because the agents can share revenues 'behind the scenes'.)

- Following from the above point, the recommender systems work seeks to cope with a single 'attack' launched by multiple accounts, whereas our work attempts to detect coalitions engaged in broader cooperative action. Each account/profile in a shill attack is typically only used for one attack, while in our work, we are also interested in longer-term coordinated activity.

- In the recommender systems work, users do not interact with one another (at least, not as part of the filtering/recommendation system itself). Instead, each user is simply a data point, used during computation of recommendations. In many of the scenarios we target (e.g., reputation systems for marketplaces), there is substantial interaction between participants. This type of domain allows for repeated interaction, acquisition of direct experience, the ability to react to other participants behaviours, etc.

- Very specific attack profiles have been identified for recommender systems, because the systems are well known and the goals of attackers are very specific. In our scenarios, we cannot be confident that we have a thorough set of known attacks.

2.6 Sybil attacks

A topic that relates directly to our own is that of Sybil attacks [15]. In a Sybil attack, a single entity makes use of multiple identities or accounts, in order to improve its situation in some way. Sybil accounts can be used, for example, to engage in ballot-stuffing or bad-mouthing—the collusive reviews would thus appear to be coming from multiple identities rather than a single entity, which can make them more effective.

Sybil attacks have received substantial research attention. Levine et al [45] present a survey of approaches to dealing with the problem of Sybil attacks. Many proposals depend on certification of identity by some trusted authority. While this provides strong protection against Sybil attacks, it is impractical for many of the application domains that we target.

There has been a particular focus on Sybil attacks in the context of peer-to-peer networks. This focus has inspired proposals that may be directly applicable to that domain, but of limited usefulness elsewhere. For example, many have proposed systems where the capabilities of a network node (e.g., its computational power or network bandwidth) are tested, based on the assumption that an entity sharing its resources amongst numerous Sybil identities will demonstrate lesser capabilities than an authentic node (e.g., [5, 62]).

Many approaches focus on differences between Sybil identities and authentic entities. This would include the 'resource test' proposals mentioned above, but extends beyond such approaches. For example, the authors of SybilGuard [85], note that authentic entities form relationships with one another, but that it is much less common for relationships to exist between authentic users and Sybil accounts—it takes deception on the part of the Sybil identity to lure an authentic user into such a relationship. This leads to graph structures where groups of Sybil agents are connected to the larger population by a relatively small number of links (called *attack edges*). SybilGuard (and the later SybilLimit [84]) limit the effectiveness of creating Sybil accounts by, in simple terms, treating all accounts that traverse the same attack edge as part of the same group of Sybil identities. Attack edges are identified using a form of random walk. SybilDefender [77] applies a very similar method. SRNC ('Sybil Resisting Network Clustering') [81] identifies attack edges by computing the shortest path between each pair of nodes, and identifying the edges that are most frequently traversed. (Note that this is closely related to techniques for community finding, discussed above.) Then, SRNC prevents communication along those edges, to inhibit the activity of Sybils.

Other approaches rely on specifics of the application domain. For example, Piro et

al [55] propose a system to detect Sybil accounts in mobile ad hoc networks. A key feature of such networks is that entities move in space. Essentially, this proposal treats identities that are usually seen to be geographically close to one another, as likely to be Sybil accounts—one entity (in one place) masquerading as multiple identities.

Sybil attacks are important to our own work: they constitute attempts by multiple identities to improve the mutual benefit of the group, exactly the type of behaviour with which we are concerned. However, Sybil attacks are limited in ways that we cannot expect our coalitions to be, and likewise, existing approaches to coping with Sybil attacks do not address our more general problem. For example, coalitions in our problem are likely to be real entities, and thus have relationships with many other authentic identities. (It was noted earlier that our coalitions are likely to be very well connected.) This fact renders the common graph-based approaches ineffective for our problem. Further, our goal is a system that is applicable across a range of domains; solutions that are dependent on specific scenario properties will not accomplish this.

In fact, we consider Sybil attacks[2] to be special cases of the larger issue of coalitions: the Sybil accounts, being controlled by a single entity, can be viewed as a very well-coordinated, very loyal coalition.

2.7 Conclusion

This survey highlights the importance of coalitions in areas such as trust and reputation; it also illustrates that related work in various research fields does not offer a solution to the problem we seek to address.

In the next chapter, we begin our investigation by describing the experimental platform used to explore key issues, and to validate our work.

[2]More specifically, we refer here to Sybil attacks where the attacker attempts to use certain of the accounts to benefit other of the accounts, e.g., by manipulating reputation scores. In Chapter 4, we identify uses of Sybil accounts that are not collusive in nature.

Chapter 3

TREET: The Trust and Reputation Experimentation and Evaluation Testbed[1]

The area of *multiagent systems* is concerned with scenarios where a number of agents (who may be acting on behalf of different users) must interact in order to achieve their goals; often, an agent must depend on other agents in order to achieve its objectives. In such scenarios, trust can be an important issue—an agent's ultimate success may depend on its ability to choose trustworthy agents with which to work. For this reason, *trust and reputation systems* (TRSes)[2] have received much attention from researchers. Such systems seek to aid agents in selecting dependable partners (or in avoiding undependable ones).

A particular focus for researchers has been on the electronic marketplace scenario, a well-established and important example of a multiagent system. In this setting, agents act as traders, buying and selling amongst one another. The ability to find trustworthy partners is critical to an agent's success, because an untrustworthy agent may deliver an inferior good (or fail to deliver at all), or may not pay for goods purchased. The nature of electronic marketplaces complicates the evaluation of trustworthiness: identity is difficult to establish (because new accounts can be created easily), agents may not engage

[1] Earlier versions of the work in this chapter have appeared at IFIPTM [40], and in the journal, *Electronic Commerce Research* [42].

[2] For convenience, we use the abbreviation TRS, for 'Trust/Reputation System', in reference to both trust systems and reputation systems.

in repeated transactions together (because of the size of the market and the diversity of products), and an agent may have an advantage over another during a transaction (for example, when a buyer must pay in full before a seller ships (or fails to ship) the good).

Along with the multitude of TRS proposals have come a similarly large number of methods to evaluate the proposals. It has been widespread practice for researchers in the field to develop their own tests; typically, these are 'one-off' evaluations, each used to validate a single new proposal. (Subsequently, such proposals might appear as comparison data points in the evaluation of later proposals—albeit using the later authors' own self-devised tests.) Some authors have used mathematical analysis to substantiate certain desirable properties of their systems (e.g., [36, 29]). More commonly, researchers have conducted simulations (using a scenario of the authors' own devising) to show the value of their model (e.g., [73, 83, 76]). Typically, the new proposal is pitted against the authors' implementations of an existing model (or small set of models).

There is nothing fundamentally unreasonable about either of these approaches, in the absence of established testing tools. In practice, however, evaluation has proven to be challenging, in a number of ways:

1. It should come as little surprise that such tests, devised by authors themselves, often appear to favour the authors' own work. This is not to suggest any misdeeds on the part of these authors; rather, when designing a system, it is natural to have a particular scenario in mind, and for subsequent tests to reflect that scenario.

2. Because each evaluation is different, results presented by different authors are not comparable.

3. The evaluations presented are often quite brief, leading one to question whether the results thoroughly reveal the performance of the systems in question. Indeed, our investigations (which have been published [41], and are detailed in the next chapter) revealed numerous ways in which existing systems (including those cited above) can be defeated—issues that were not revealed in the authors' own analyses. These issues highlight the need for more thorough, objective testing of TRSes, ideally using tools that allow comparison and reproducibility of test results.

4. Perhaps most importantly, the evaluations typically used obscure critical problems that have received insufficient attention to date by trust and reputation researchers. Simulations typically make use of agents that are simple or naive in their dishonest activities. For example, many simulations (e.g., [73, 83]) are populated by random selections of agents that either always cheat or always behave honestly, or by agents

whose cheating is governed by simple probability distributions, where each time step is independent of previous ones. Further, the agents act alone, rather than coordinating their efforts in any way. Such simulations ignore the possibility that cheaters might behave in a more sophisticated manner—for example, trying to identify and exploit a specific weakness in the system—providing little comfort to those who might wish to consider these proposals for real-world use.

In earlier work [36], we identified a number of common vulnerabilities that might allow attackers to defeat the protection offered by TRSes, and argued the critical importance of security in TRSes. Subsequent work [41] (described in the next chapter) has demonstrated the practicality of such attacks by soundly defeating a number of noteworthy TRS proposals. These results demonstrate the need for more rigorous tests, and more objective tests, of TRSes.

3.1 The ART Testbed

A standardized, common evaluation tool can potentially address the issues noted above. Members of the trust and reputation community have invested significant effort in developing a standardized simulation: the Agent Reputation and Trust (ART) testbed [17]. A primary purpose of ART is to serve as a competition platform, and it has served this purpose well [74], having been used for competitions at a number of conferences. While ART is a valuable contribution, a number of design choices make it less appropriate for broad use in the experimental evaluation of TRSes.

In ART, agents are art experts, each with varying levels of expertise in different eras. Agents are periodically asked to appraise pieces of art by clients. The accuracy of the appraisals given to clients determines how much business each agent will receive in the future, according to a fixed formula used by the testbed. The agent can choose how much to invest in generating its appraisal—greater investment yields greater accuracy. If an agent is asked to evaluate a piece from an era about which he is not knowledgeable, he can seek appraisals from other agents. Agents can also share information with one another about the reliability of other agents' appraisals.

ART is very well-designed for its primary purpose: evaluating competitive agents (making use of a number of abilities) who interact directly with one another, in a small social trust scenario. ART has a number of desirable properties for a testbed:

- It offers a well-specified, standardized testing scenario and set of rules.

- It allows new agents to be easily implemented and plugged into the system; agents can then be used by others for future experimentation.
- It provides objective metrics for comparison between systems.

That said, ART has a number of features that make it less suited for general-purpose trust and reputation experimentation:

- Under ART, the distinction between buying and selling agents is unclear, making some forms of experimentation problematic. The ultimate purchasers of appraisals (the 'clients') are buyers, and as such, agents serve as sellers for these transactions. Note, however, that the clients' method of choosing appraisers (based on past performance) is fixed by the ART specification, precluding experimentation with buyer-side modelling of sellers for these transactions; similarly, it obviates investigation of sellers modelling potentially unreliable buyers. In contrast, as agents buy and sell appraisals with one another, each agent acts as both buyer and seller. Success under these circumstances requires a multitudes of diverse skills: determining when to make do with your own knowledge, and when to seek help; determining how much to invest in appraisals; determining whom to trust when seeking appraisals; and determining whether or not to be honest when another agent asks you for help. While this is a demanding test, and one appropriate for a competition testbed, it can also obscure the role of each individual skill in an agent's performance. This makes it difficult to isolate individual marketplace components for evaluation. For example, if a researcher wishes to evaluate the performance of a system intended to allow sellers to model untrustworthy buyers, it may not be useful to have the results clouded by the same agent's performance in the unrelated task of deciding whether or not to make honest sales to other agents.
- In its role as a competition testbed, ART requires a very well-defined scenario. Unfortunately, this requirement seems to limit the flexibility of the system for experimentation. For example, the ART architecture allows decentralized (where each agent maintains its own database of reputation values for others) and direct experience models, but precludes testing of centralized models (where one central store of reputation information is used), because the method of sharing information amongst agents is specified by the testbed. It also prevents experimentation with models that regulate an entire marketplace (e.g., mechanism-design based approaches). Moreover, features of the chosen scenario prevent investigation of important issues. For example, each appraisal has a fixed price under ART,

preventing exploration of vulnerabilities such as Value Imbalance (where a seller builds reputation by honestly executing small-value sales, then uses the reputation gained to cheat on larger ones [36]). The quality of an agent's appraisal is reflected in clients' decisions in the next timestep, preventing exploration of vulnerabilities such as Reputation Lag (where a seller can cheat a large number of sellers for a period of time before his reputation is updated to warn other potential victims [36]).

- ART provides a heterogeneous environment where agents share reputation information with agents using different trust and reputation models. To permit communication between agents with different internal models, the format of communication is determined by the ART specification. This imposes a specific trust representation for communication between agents (if not for agents' internal use); the imposed format may not map well to the TRS's native representation, potentially disadvantaging the TRS.

Some authors (e.g., [24, 25]) have noted the limitations of ART for evaluating their own work.

3.2 The TREET Testbed

In this section we describe TREET, the Trust and Reputation Experimentation and Evaluation Testbed. TREET was formulated to support diverse, flexible experimentation with TRSes, and more thorough, objective evaluation of such systems. (The results obtained using TREET, detailed in the next chapter and throughout much of the remainder of this document, demonstrate its value for these purposes.) In contrast to the competition focus of ART, TREET is designed specifically to support general-purpose experimentation and evaluation of trust and reputation technologies.

The TREET platform has a number of important advantages making it well suited for its intended purposes, including:

- It models a general marketplace scenario, allowing systems to be tested under realistic conditions. This includes reasonably large marketplace populations, turnover in the agent population, a large number of products/prices, etc.
- It is modular, allowing new TRSes, buying and selling agents, instrumentation, etc., to be added easily.

- It can support a wide range of trust/reputation approaches (for example, both centralized and decentralized models). It does not impose any particular view of trust on agents, nor does it impose a particular protocol or trust representation on agents.
- It allows collusion to be incorporated into agent behaviour.
- It allows individual marketplace 'components' to be tested in isolation. For example, it allows the protection a TRS provides buyers from cheating sellers to be evaluated, without being obscured by other potentially irrelevant issues (for example, whether or not sellers are dishonest with one another). In contrast, the success of an agent in ART requires competence in a number of abilities.
- Given the standardized platform, as new agents/TRSes are developed, they can be evaluated against all existing implementations; at the same time, new implementations constitute new tests for all of the existing systems. In this way, a continually improving battery of rigorous tests might be developed, which can be used by researchers to evaluate their work.
- Standardization also allows for objective benchmarking, permitting meaningful comparison between systems. Moreover, the availability of components will allow for results to be reproduced by other investigators.

The specifics of TREET are detailed below.

3.2.1 Conception and Goals

We sought to formulate a testbed that would support flexible experimentation and meaningful evaluation of trust and reputation technologies. Complete marketplaces may have many TRS components, from a range of possibilities: agents who have individual (and heterogeneous) internal models of other agents' trustworthiness, networks of agents that share reputation information, centralized repositories of reputation data, market-wide mechanisms that regulate trading between agents, etc. A potential adopter of a TRS may have to choose between multiple proposals, despite the fact that the proposals use very different methods internally. An adopter may have to assemble multiple TRS technologies to meet the needs of their complete working system, and may need to understand how well these components work together. For these (and other) reasons, a testbed will

ideally support experimentation with a wide variety of such components. Thus, we set out to design an architecture that was quite flexible.

At the same time, too general a testbed formulation might also be difficult to apply in practical terms. At best, it may be of little benefit to the researcher, leaving much work to be done simply in preparing the testing platform. Worse, a formulation that is too general can make evaluation of TRSes and comparison of results problematic: different researchers are likely to use very different instantiations of the testbed scenario, raising many of the same issues as the author-devised testing that has occurred to date. For this reason, we have specified a well-defined scenario that we believe is useful for a wide range of experimentation. We believe that this is an appropriate and useful balance between flexibility and standardization.

Nature of Tests

For a competition testbed, it is sufficient to supply the testing platform itself; competitors supply the agents, which seek to defeat one another. In contrast, a testbed intended for evaluation and benchmarking requires meaningful tests for candidates to perform. In some fields (for example, performance benchmarking of computer components), a typical approach would be to develop a set of standardized tasks to perform, with well-defined metrics (e.g., execution time) used for comparison. Ideally, the tasks would be representative of real-world demands. For TRSes, however, it is difficult to envision representative 'tasks' that do not involve actual interaction with other agents. The most illuminating tests are likely to be those conducted in a realistic scenario, interacting with other agents. Thus, in our formulation, tests consist of (at least) two components: a well-defined marketplace scenario, and a population of agents. The TRS technology in use might be a third component (e.g., in the case of centralized systems), or might be incorporated into the behaviour of the agents themselves.

This formulation provides a great deal of flexibility, as well as the ability to test specific components under controlled circumstances. For example, to test TRSes that attempt to allow buyers to cope with cheating sellers, a test would consist of a set of market parameters, and a population of sellers with specific cheating behaviours. These components are experimental controls; each TRS would then be tested against the same scenario, allowing comparison of the results. (This approach is used in the next chapter.) In comparison, to test TRSes that allow sellers to cope with untrustworthy buyers, each test would include a different set of buying agents.

Beyond the benefits noted above, this approach has a number of advantages. First, as

agents are developed (both TRS technologies, and 'tests'), they can be made available to other researchers. This allows the test suite to grow, increasing in thoroughness and rigor, as understanding of TRSes increases. (The cheating agents described in this document constitute an initial set of tests.) Second, the standardization of the platform and the availability of agents allows results to be reproduced by other researchers.

3.2.2 Scenario

We sought to develop a reasonably general scenario, one in which a variety of roles and strategies can be evaluated. For tests to be meaningful, the platform should model as realistic a scenario as practically possible. In the following, the parenthesized values are default settings representative of a reasonable scenario. (These values were used in published experiments [41].) A test specification would include a set of parameter values; the values can be adjusted as desired for experimentation.

We model an 'advertised-price' marketplace: sellers offer goods for sale, and buyers choose whether or not to make purchases, and from whom. A fixed set of products (1000) is available for sale. Because we wish to study trust primarily, and not other price-/cost-based forms of competition, there is an established market price for each good: every seller charges that same price for a given good. A typical marketplace will have more inexpensive items for sale than expensive ones. To reflect this, the price of each good is randomly determined using the right half of a Gaussian distribution (i.e., the median occurs at $0, and probability decreases as price increases).

A seller incurs cost in producing or acquiring each good that he sells. This is a primary motivation for cheating—to avoid incurring this cost, and thereby increase profit. Again, to keep focus on issues of trust and reputation, rather than profit margins, all sellers incur the same cost to produce each product (75% of selling price).[3] Similarly, a buyer who wishes to avoid being cheated can simply refuse to make any purchases. Doing so, however, means that she is doing without products that she needs, incurring a loss of sorts. Thus, each product has a utility value (110% of selling price) that a buyer realizes if a needed product is acquired successfully, which provides motivation to make purchases.[4] It is assumed that all participants have access to the cost and utility values

[3]TREET also allows a commission to be charged for each sale (0% by default). The commission is charged based on the sale price, regardless of whether the seller chooses to fulfill the sale honestly or not. This feature allows investigation into issues such as impeding ballot-stuffing by making it costly to engage in fabricated transactions, because each such transaction will incur a commission.

[4]While these parameters are configurable, cost should be strictly less than selling price, and utility should be strictly greater than selling price, or the motivation to engage in sales is not present.

for each good. These factors support investigation of a variety of aspects of trust and reputation; for example, both buyers' modelling of sellers, and sellers' modelling of buyers can be examined.

Each seller is assigned a random number of products that she is able to produce, selected from a uniform distribution (maximum of 10). To reflect the greater availability of less expensive products, the products are again randomly assigned using the right half of a Gaussian distribution (i.e., the median occurs at the least expensive product, with declining probability as price increases).

A simulation run can be populated by an assortment of agents, as desired by the researcher, or as defined in a test specification.

Marketplaces are often dynamic—traders join and leave regularly. This is important for TRSes, because new agents are unknown (and have no knowledge of other agents), and departing agents result in obsolete knowledge. For efficiency, agents join/exit the market at specific intervals (100 days). After each such interval, each agent departs the marketplace with a fixed probability (0.05). That said, it may be undesirable for the performance of TRSes to be clouded by changes in market size (e.g., profits increasing because the number of buyers increases.) Thus, for every departing agent, one agent of the same type joins, keeping the participant count constant.

3.2.3 Architecture

TREET is designed to be quite versatile for experimentation, within the constraints of the defined scenario. The architecture is depicted in Fig. 3.1. In this diagram, BA and SA refer to Buying Account and Selling Account respectively. BE and SE refer to Buying Entity and Selling Entity respectively. All components labelled in italic text are components that are intended to be provided/modified by investigators making use of the testbed. The grey box denotes those components that are observable by marketplace participants, although this does not imply complete visibility. For example, seller accounts may be visible to buyer accounts, but this does not imply that all seller account data is visible. (E.g., the user ID of the account would be visible, but the amount of money held by an account might be private.) Such limitations are described in more detail below.

Each complete run of the testbed is represented by a *SimulationRun*, into which the necessary arguments and objects are passed. A SimulationRun is responsible for setup and configuration of a run—creation of the product set, initialization of components, etc.—and initiating the Simulation Controller. A set of numerous tests can be executed by creating multiple instances of SimulationRun.

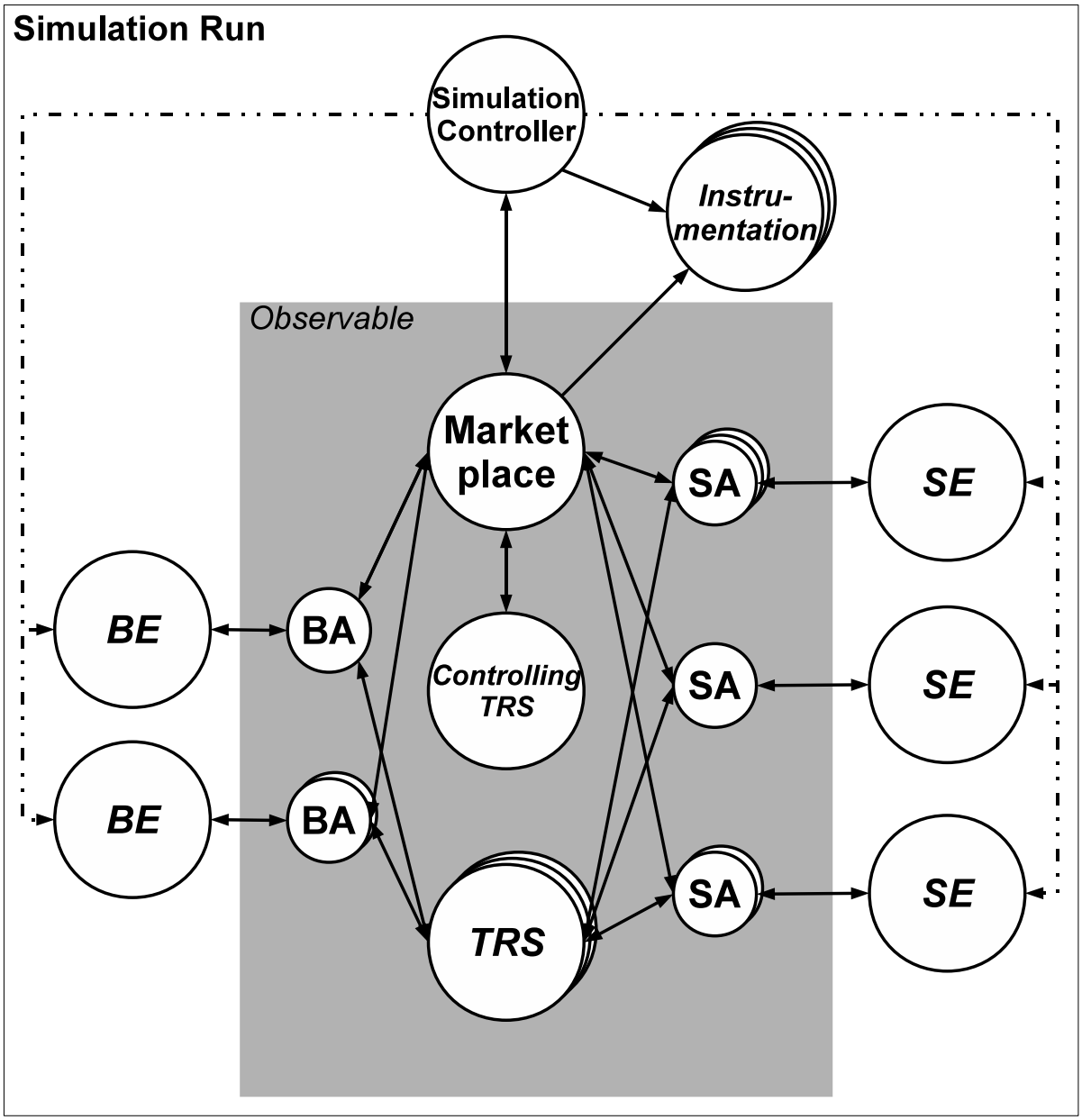

Figure 3.1: The TREET Architecture.

A *Simulation Controller* is responsible for actual execution of a simulation run. The controller triggers each of the day's events in turn, signalling the appropriate parties when they are required to take action. For example, the controller cues sellers to make product offers at the appropriate times, cues buyers to select products/sellers when offers have been posted, etc.

The scenario makes use of a single centralized marketplace, represented by a *Marketplace* object; this is consistent with the centralized model employed for marketplaces such as eBay. All offers, acceptances, and payments are made through the Marketplace. All accounts reside in the Marketplace, and requests to open accounts are processed through it.

One important aspect of TREET is the role of TRSes and agents. Some TRSes are implemented entirely centrally, some entirely within the agents; many fall between these two extremes. The option in TREET to use both agents (represented by accounts and entities, as described below), and TRS objects, facilitates a wide range of approaches. A *TRS* object implements those components of a TRS that are shared by multiple agents. For example, in a model that makes use of a centralized repository of reputation information, the TRS object would provide that service. TRS objects are useful even in fully decentralized systems, if only to coordinate trust-related actions. For example, when a buyer requests reviews from other buyers, this request could be processed through the TRS object, whose role is simply to co-ordinate such communication.

Some important points should be made regarding TRSes. First, as depicted in the diagram, multiple TRSes may be in use simultaneously, for example, by a heterogeneous population of agents. Second, in order to implement a system for experimentation, matching TRS objects and entities typically must be developed—a researcher must create both the TRS, and entities that 'know how to use' the TRS. The role of each type of component, and the nature of interaction between TRS and entity, is at the discretion of the researcher, allowing great flexibility in implementing TRSes and agent behaviours. For example, in a completely decentralized model, entities may do all reputation tracking and computation; in this case, the TRS might simply serve as the communication channel between agents. At the other extreme, with a completely centralized model, the TRS may perform all reputation-related functions, while entities simply make use of the services provided. TREET supports this diversity, which seems to provide a great deal of flexibility, without undue complication. Third, in some cases (e.g., a market-wide mechanism), a TRS is tightly integrated into the operation of the marketplace itself. For example, Basic Trunits [36] controls what offers may be made by sellers. TREET allows for a single *ControllingTRS* to be installed in the marketplace. When a ControllingTRS is present, it can control which offers are permitted to be posted by sellers, modify those

offers, control whether a buyer will be permitted to make a certain purchase, etc. In the absence of a ControllingTRS, the marketplace executes all offers/purchases/etc. without interference. Finally, note that both buyers and sellers can make use of TRSes. This allows for experimentation and evaluation of, for example, TRSes that help sellers to select buyers that will provide honest or favourable reviews.

Note that no particular trust representation, or communication protocol is enforced between agents. In fact, it is up to the designer of the TRS component, along with the associated agents, to determine exactly how (or whether) communication takes place between agents. This provides support for a wide range of approaches to trust. Note, too, that communication between heterogeneous agents can be supported. Certainly, mapping from one agent's trust representation to another's may be necessary, but the method for doing so is up to the designer of the TRS component used for communication. (This, in turn, also allows experimentation with different means of allowing heterogeneous agents to interact.)

Support for Investigating Coalitions and Collusion

Another important feature of TREET is the separation of the agent roles into two components: accounts and entities. *Accounts* represent actual user accounts within the marketplace; these are the identities that are observable by other parties in the market. *Entities*, however, represent the actual agents performing actions by using the accounts. Entities are not observable by marketplace participants, reflecting the fact that identity is difficult to establish, particularly in large electronic marketplaces. This distinction is important for a number of reasons. It allows entities to interact with the marketplace (via accounts) without revealing their true identities to other marketplace participants, important in modelling many real-world scenarios. It allows the re-entry phenomenon to be incorporated into experiments (where agents can simply open new user accounts to shed a disreputable identity). It allows for a single agent to control multiple user accounts (as they may in real-world scenarios), as used in attacks demonstrated in the next chapter. It also allows for investigation of collusion. In the case of perfectly loyal and coordinated collusion, a single entity can represent the entire coalition; in the case of less perfect coalitions, entities can be implemented that communicate with one another outside the observable marketplace.

Buying and selling accounts are shown as distinct in our architecture diagram, as are buying and selling entities. TREET supports this strict separation, which is useful for some forms of experimentation. (For example, one might wish to charge different

amounts for opening buying and selling accounts.) In the interest of flexibility, however, TREET also allows a single account to be used for both buying and selling, and a single entity to act as both buying and selling entity. Whether or not these dual roles are permitted may be configured by parameter. This flexibility allows, for example, the study of forms of collusion such as ballot stuffing, by permitting entities to play the role of both buyer and seller.

For different components of the testbed to communicate with one another, certain aspects of communication must be standardized. In TREET, communication for actual market transactions (i.e., actual purchases) is defined by the specification: the syntax and semantics of product offers, offer acceptances, etc. These are items that are likely to be standardized in a real marketplace situation. Note, however, that the characteristics of communications between TRS components (e.g., exchange of reputation information between agents) are not imposed by the specification, instead left to be determined by those implementing TRS/agents for the system. This ensures maximum flexibility and fair treatment of different TRS approaches. Illustrating this flexibility, in the next chapter this architecture is used to evaluate five noteworthy TRSes [73, 83, 36, 29, 76]. This set comprises a wide range of (very different) approaches to trust/reputation, including direct experience, witness information, and centralized mechanisms; for each model, agents were able to represent and communicate trust in the native form as described by the authors, without conforming to a specification imposed by the testbed. This demonstrates the versatility of the platform.

To support flexibility of experimentation and evaluation, TREET must also support flexibility of measurement, as researchers investigate a wide range of issues. This support is provided by *Instrumentation* modules. Each instrumentation object is provided with the identity and group-membership of every entity, the ownership of every account, and the details of every sale completed. With this data, Instrumentation objects can provide arbitrary output and metrics, from detailed turn-by-turn or sale-by-sale information, to heavily summarized aggregate data. For example, a default Instrumentation class that accumulates sales/profit/cheating statistics over time, by agent group, was used in the experiments in the next chapter. Researchers are also free to develop their own. Multiple Instrumentation objects may be in use during a simulation. Instrumentation modules also serve an important role in test suites, determining whether a test has been passed or failed.

Several other details are worth noting:

- Buyers do not know of selling accounts until that seller makes an offer. Sellers do not know of the existence of a buying account until it makes itself known by

accepting an offer.

- At the time of making an offer, sellers do not know or control whether an offer will be accepted, or by whom.
- A seller can only *physically provide* products that she is able to produce. A seller is able to *advertise* and *sell* (dishonestly) any product, however.
- It is possible for an agent to connect to multiple TRS objects. This allows experimentation with situations where, for example, an agent might make use of shared reputation information with trusted neighbours, as well as accessing data in a centralized repository (i.e., a different TRS).
- Entities can create new accounts at will. TREET provides the ability to charge a fee for each account opened; the fee is determined by the ControllingTRS, if one is in use. Entities can query to determine the fee, before deciding whether to create an account.
- A ControllingTRS has the ability to close accounts, rendering them unusable by their owners.
- Entities are not resource-constrained—they have unlimited money available. Instead, we track each entity's net financial position, the sum of all monetary gains and losses, so we can determine how effective the entity's strategy has been.

3.2.4 Simulation Execution

Each round represents one day. Each round consists of the following steps (co-ordinated by the System Controller):

1. All participants are notified that a new round is beginning, so that they may do any needed processing/initialization. Such initialization might include reseting internal data structures, opening new accounts, etc.
2. After entering into a sale, a buyer will not know whether or not he has been cheated until after some number of days has passed, reflecting processing, shipping, etc; we refer to the rendering of feedback after this *lag* (default of 14 days) as the *completion* of the sale. At the beginning of each day, each buyer is notified whether each completing sale was executed honestly or not, representing 'delivery' of the

item.[5] The buyer's net utility is updated to reflect the result: the utility is equal to the full utility for the product times the degree of fulfillment. (Note that a buyer only earns utility for a product that meets an assigned need (see Step 6); if it buys an unneeded product, it receives zero utility.)

3. After learning about the outcome of each completing transaction, the buyer determines its satisfaction with the transaction (in the manner determined by the agent designer, and expressed in the representation appropriate for the TRS in use). Each buyer has the opportunity to submit feedback to all TRSes to which it belongs (applicable to those TRSes that require agents to report results, rather than those in which agents respond to queries). It is also prompted to submit feedback to the ControllingTRS (if one is in use.)

4. Each seller is prompted to submit feedback to all TRSes to which it belongs, and to the ControllingTRS, as above.

5. Each market participant, including each TRS in use, is prompted to process all feedback that it has received in this round.

6. Each buyer's needs are determined for the day. Each buyer is randomly assigned a set of products (default, of up to 5) that it needs to purchase that day; again, these are selected using the right half of a Gaussian distribution, so there is a greater likelihood of needing lower-priced items.

7. Each seller is prompted to make offers, submitting them to the marketplace. No limits are placed on sellers' capacity or inventory; only one offer per product is allowed for a given account. If a ControllingTRS in use, it may reject offers in the set submitted by a seller. After submitting its offers, the seller is returned a list of the offers that were accepted for posting by the marketplace/ControllingTRS. If it wishes to, it can then modify its offers and post a new set, replacing the previous set. This cycle can be repeated as many times as needed, until the seller is satisfied.

8. Buyers select the products they wish to purchase, from which sellers, by consulting the posted offers. Buyers are free to consult their TRSes in so doing. For each purchase they decide to make, an offer acceptance is communicated to the corresponding seller account, via the marketplace. The acceptance of an offer can be

[5] The degree of fulfillment is represented by a value in the range [0, 1], with 1 representing complete fulfillment, and 0 representing complete lack of fulfillment, e.g., not shipping the good at all. In our experiments, sellers selected between these two extremes, but researchers are free to implement other behaviours.

rejected by the ControllingTRS. A seller can also refuse to make the sale to the buyer (e.g., if the buyer has a bad reputation for giving poor reviews); it can consult its TRSes in making this decision. The buyer is notified if the purchase was successful; if not, it can try again. This cycle can be repeated as many times as needed, until the buyer is satisfied.

9. For each sale that the seller agrees to make, it decides whether or not to fulfill it honestly or dishonestly. The cost to the seller for providing the good is equal to the full cost of the product times the degree of fulfillment (as described above). Acceptances are communicated to the marketplace, which forwards each to the corresponding buyer account.

10. Payment is transferred from the buyer account to the seller account, for each sale.

11. Each sale's status (honest or dishonest) is communicated by the seller to the marketplace for storage. This value is not observable by any other marketplace participant, until the buyer is notified during Step 1 of a later round.

12. All participants are notified that the round is ending, so that they may do any needed processing/cleanup.

3.2.5 Initial Test Set

Much of the value to be gained from an evaluation testbed such as this is the value of the tests: their difficulty, their breadth, and their representativeness of the sorts of issues TRSes might face in a real environment. As an initial set of tests, those agents described in the next chapter will be provided (including those used in our published investigations [41]). These agents employ a number of different tactics, and were designed to test the robustness of TRSes that attempt to cope with dishonest sellers. This test set is far more extensive and difficult than any we have seen used for evaluation of TRSes to date; as demonstrated (in published work [41], and in the next chapter), this set of attacks was quite devastating to the set of TRSes evaluated—all of the TRSes were defeated by numerous attacks.

3.2.6 Use and License

TREET is meant to provide an experimentation platform for researchers. Thus, TREET will be released as open source software, so that researchers can freely use and modify

it for their own purposes. A key goal for TREET, however, is to provide objective, reproducible, and increasingly thorough evaluation of TRSes. To that end, there will be some requirements for publishing of results, enforced by the software license, to support the vision for TREET and its value to the trust and reputation community. In particular:

- For authors to state that their test results were obtained using TREET, the platform and test software must be unmodified, and runtime parameters must be fully specified;
- The authors' code (in particular, the implementation of their agents) must be made publicly available, so that results can be independently reproduced and evaluation, and so that the test set can expand in the future.

While we are reluctant to impose any restrictions on use, we believe that the value of TREET to the community would be undermined without them.

3.3 Discussion

As noted earlier, TREET models a centralized marketplace scenario. We are careful to note the distinction between decentralized *marketplaces*, and decentralized *TRSes*. TREET provides no impediments to the use of decentralized TRSes, which can be readily implemented. We note, as well, that while the Marketplace is centralized, there is little about the architecture that prevents experimentation with decentralized marketplaces (e.g., a peer-to-peer network where agents sell directly to one another). In this perspective, the 'marketplace' is an abstract notion, representing the means by which offers are accepted, payments are made, etc., within the decentralized system. In this scenario, no centralized TRS would be used. Opening an account, in this case, represents the act of an agent creating a new identity; the marketplace does not limit the creation of accounts, nor communicate their existence to other agents until they reveal themselves by offering/buying products. The one notable violation of the decentralized marketplace perspective is that of offer advertisement. Under TREET, each offer that a seller makes is posted centrally, for all buyers to see; this is analogous to all offers being broadcast throughout a decentralized system. If this limitation is acceptable, then TREET may be suitable for experimentation with decentralized marketplaces.

TREET was inspired by the desire to study the security of trust and reputation systems. It provides (at minimum) a platform for researchers to try to defeat existing trust and

reputation technologies. We must be careful to distinguish, however, between defeating a TRS, and defeating TREET itself. While TREET attempts to provide checks against some forms of 'inappropriate' use (for example, 'forging' an offer by another entity), it is easy to envision ways in which one might bypass these protections. For example, one might create a TRS that is rigged to steal identities and provide them to colluding agents to be used for crimes. Given the nature of the platform, and the fact that developers have access to the source code, it is trivial to 'defeat' the platform. Such activities are well outside the intended purpose of this platform, and provide no insight into trust and reputation. This is one reason that the TREET license insists upon public availability of researchers' code in order for them to be able to publish results: to allow peers to verify that the results reflect legitimate insight into trust and reputation.

The TREET testbed allows a breadth of experimentation and a thoroughness and objectivity of evaluation that have previously been unavailable from publicly-available, standardized testing tools. As demonstrated in the next chapter, the design of this testbed is a proven one. It has been used to shed light on important issues that had previously been unexplored experimentally—in particular, the degree to which existing TRS proposals can withstand cheating agents that actively attempt to circumvent the protections of the system. The platform has shown itself to be flexible, supporting experimentation with TRSes using a variety of approaches.

TREET is intended to allow more thorough testing than has typically been performed for TRSes in the past. As TRS researchers develop new agents, the test suite will grow, increasing the rigor of the evaluations, and the insights provided. TREET's terms-of-use have been designed to support this vision.

We believe TREET to be an important tool in itself, and a significant step towards improving the evaluation of TRSes. Future extension and refinement of the platform, and of the accompanying test suites, will only increase its value to the community, and its usefulness in furthering the cause of security in trust and reputation system.

Chapter 4

"Smart Cheaters": Substantiating Vulnerabilities in Trust and Reputation Systems[1]

In the field of multiagent systems, the success of an agent may depend on its ability to choose reliable partners; for this reason, trust and reputation systems have received significant attention from researchers. A particular focus has been on the electronic marketplace scenario, a well-established and important example of a multiagent system. In this setting, agents act as traders, buying and selling amongst one another. The ability to find trustworthy partners is critical to an agent's success, because an untrustworthy agent may deliver an inferior good (or fail to deliver at all), or may not pay for goods purchased. The nature of electronic marketplaces complicates the evaluation of trustworthiness: identity is difficult to establish (because new accounts can be created easily), agents might not engage in repeated transactions together (because of the size of the market and the diversity of products), and one agent might have an advantage over another during a transaction (for example, when a buyer must pay in full before a seller ships the good (or not)). A variety of approaches have been proposed to cope with these difficulties; we outline several such approaches in the next section.

The fundamental motivation for work on trust and reputation systems (TRSes) is the understanding that some individuals may be dishonest. Typical proposals seek to provide some measure of protection for market participants against such dishonest traders—most

[1]Much of the work detailed in this chapter appeared at AAMAS [41]; the results here have been expanded and updated.

frequently, the proposals attempt to predict to what degree an agent will execute transactions honestly in the future. Work in this area often adopts a limited perspective, however. While it is assumed that agents may attempt to exploit *each other*, little consideration is given to the possibility that the agents may attempt to exploit the *system itself*. In fact, existing systems commonly suffer from vulnerabilities—weaknesses that may allow an unscrupulous trader to undermine or bypass the protection offered by the system.[2] In earlier work [36], we presented a catalogue of such vulnerabilities—opportunities for agents to cheat other users without the system preventing it, and without the agent being penalized. In either case, such vulnerabilities represent fundamental breaches in the protection offered by the system.

We have contended [39] that security is a critical issue for designers of trust and reputation systems—if vulnerabilities exist that allow agents to achieve increased profit by cheating, we should expect profit-maximizing agents to take advantage of them. Nevertheless, this issue seems to have received little attention in the trust community. This is borne out, for example, in the simulations typically used by authors to evaluate their proposals. For example, many proposals are validated using simulations (e.g., [73, 76, 83]) populated by random selections of agents that behave consistently, or by agents whose cheating is governed by simple probability distributions, where each time step is independent of previous ones.

In this chapter, we examine the issue of vulnerabilities in TRSes. In particular, we experimentally demonstrate the practicality and gravity of attacks on TRSes, by developing agents that purposefully and successfully employ cheating tactics designed to thwart such systems. Importantly, we demonstrate that these agents can cheat successfully without knowledge of the specific TRS in use, or even the general nature of the system. This undermines the notion of 'security by obscurity' for TRSes: ignorance of the system does not prevent an agent from cheating successfully. While central in illuminating the issue of security for TRSes, we also expect this research to be useful in the evaluation of future systems.

[2] We wish to be clear about our notion of 'attacks'. In this work, we do not refer to conventional attacks on the system implementation itself (for example, breaching the computer running a TRS, and modifying the software.) Rather, we refer only to attacks composed of actions within the system itself (for example, carefully-chosen combinations of honest and dishonest transactions.)

4.1 Vulnerabilities in TRSes

In earlier work [36], we identified the theoretical possibility of a number of vulnerabilities in TRSes, although such vulnerabilities had not been demonstrated.

Reputation Lag: A common policy in many electronic marketplaces is that the buyer pays before the seller ships the good. In this scenario, a seller is likely to know that he intends to cheat from the moment he receives payment. The buyer, however, will not know for some time afterward, because of processing, shipping time, etc. Under some TRSes, this presents an opportunity for a seller: he can cheat a virtually unlimited number of times before his reputation is updated to warn buyers of the new cheating activity.

Value Imbalance: In some TRSes, all reviews are weighted equally, regardless of the value of the transactions. This presents an opportunity: a seller can honestly execute small sales, then use the reputation gained to cheat on very large ones.

Re-entry: It is broadly accepted that in electronic marketplaces, we cannot assume that the identities of traders can be established. Users can create new accounts freely; in large markets, it is infeasible to verify the identity of every trader. This presents the opportunity for a dishonest trader to shed his bad reputation, starting fresh by opening a new account. This is particularly dangerous in systems that treat unknown sellers as preferable to disreputable ones.

Initial Window: In some TRSes, buyers rely only on their own experience in evaluating sellers. Once a buyer has found trustworthy sellers, this policy works well. Unfortunately, the buyer is vulnerable until he finds those trustworthy sellers—he does not have enough information to avoid cheaters.

Exit: If a seller cheats, it may damage his reputation, and hinder his ability to engage in future sales. If the seller is planning to leave the market, however, he has no further need for his good reputation. Thus, he can cheat freely, to the maximum extent possible, without consequence. This is an extremely difficult problem to combat, and affects most TRSes.

In addition, there are known attacks that involve the coordinated effort of multiple parties (or usage of multiple accounts by a single party):

Ballot stuffing (e.g., [13]): A set of agents may give falsely give each other very positive reviews (or a number of positive reviews), in order to inflate their reputations. This artificially enhanced reputation can then be used to win sales, at the expense of competing agents.

Bad mouthing (e.g., [13]): As with ballot-stuffing, a set of agents coordinates their efforts, but in this case, falsely negative reviews are used to damage the reputation of competitors. This allows the conspiring agents to win sales over those competitors.

Sybil [15]: In the Sybil attack, a single entity creates numerous accounts, for purposes such as ballot-stuffing or bad-mouthing. When sybil attacks are used for such purposes, we would consider these simply to be special cases of ballot-stuffing or bad-mouthing. Beyond these, we have noted two other important effects of Sybil accounts in this domain, discussed in Section 4.6.1.

4.2 Trust and Reputation Systems Evaluated

We sought both to validate the practicality of our attacks against a range of systems, and to evaluate the security of noteworthy TRSes. In the interest of fairness, we selected models that self-identified as applicable to marketplaces. We briefly outline our choices here; a more detailed description of each TRS can be found in Chapter 2.

Tran and Cohen

The work of Tran and Cohen [76] is representative of a *direct experience* model: agents make use only of their own experience in evaluating the trustworthiness of others. Tran and Cohen employ reinforcement learning—over time, the buyer will learn which agents can (and cannot) be trusted, and which ones give the best value for any given product. Here, we consider only the evaluation of sellers by buyers under Tran and Cohen's system.

We have noted [36] that this model is likely vulnerable to the initial window problem; in a very large marketplace, this is likely to be especially problematic, because repeated transactions are rare between traders. It suffers from the re-entry problem (because unknown sellers are favored over disreputable ones), and the exit problem. This model is immune to ballot-stuffing and bad-mouthing (due to its reliance on direct experience); it is vulnerable to some effects of the Sybil attack, however, as noted in Section 4.6.1.

The Beta Reputation System

In contrast to Tran and Cohen, the Beta Reputation System (BRS) [29] represents a *witness information* model: agents employ not only their own experience in evaluating a

seller, but also reports made by other agents. BRS uses the well-known beta probability distribution to estimate the probability that the seller will be honest on a future sale.

In witness information models, lying is always a potential issue: how can one know whether to trust another buyer's report or not? In the original paper [29], the authors propose a system where the reports from each buyer are discounted, based on the recipient's faith in the sender, before they are incorporated into the final estimate. Later, in [79], another system is proposed, where agent's reports are discarded if they are statistical outliers (i.e., if they are so far outside the distribution of most agent's experiences, so as to be suspect).

This model appears to be vulnerable to reputation lag (because agents rely on the recommendations of others, which do not immediately reflect cheating) and re-entry (because the beta distribution favours unknown sellers over those with more failures than successes). It is expected to be vulnerable to value imbalance (since transactions are counted equally regardless of value). It also suffers from the exit problem. As is typically true of witness information models, BRS is likely to be vulnerable to bad-mouthing and ballot-stuffing, and the effects of Sybil attacks.

TRAVOS

TRAVOS [73] is a later proposal that is closely related to BRS, differing primarily in how it approaches handling the reports of others. Each review is discounted based on the accuracy of previous information provided by the reviewing agent, before being combined into the final prediction. In addition, agents using TRAVOS rely on their own experience, rather than reviews from others, if they have sufficient information.

Given the similarity to BRS, we would expect this system to have similar performance and a similar vulnerability profile. As will be shown below, however, the different handling of direct experience/witness information yields different performance. Moreover, we include the model because it reflects a common thread of recent proposals: offering new methods of coping with inaccurate reports, while using established methods to compute trustworthiness of partners. This trend has important implications, which we discuss later in the chapter.

Yu and Singh

The proposal of Yu and Singh [83] is also a predictive model. It makes use of a different probability model than other proposals, however: the Dempster-Shafer theory of

evidence.

Under this proposal, like TRAVOS, an agent relies on its own experience if it believes it is sufficient. When necessary, an agent solicits information from its neighbors. If the neighbor cannot provide information, it may refer the agent to one of its own neighbors.

We have noted [36] that this model is appears to be subject to the re-entry problem (because unknown agents are intentionally treated differently from dishonest ones), reputation lag (to the degree it relies on witness information), value imbalance (because updates are not weighted to reflect transaction value), and the exit problem. As is typical for witness information models, it would also appear to be vulnerable to ballot-stuffing, bad-mouthing, and the effects of Sybil attacks.

Basic Trunits

Basic Trunits [36] stands in contrast from these systems, in that it is a transactional system in which the market operator intervenes, controlling an agent's ability to engage in transactions. Trust is represented using numerical units (*trunits*), which are required to engage in a sale. If the seller executes the sale honestly, his trunit balance grows; if he is dishonest, he loses the trunits that secured the sale. Thus, honesty enables further sales (and profits) in the future, while dishonesty curtails future sales—an incentive for honesty.

We have noted [36] that Basic Trunits is resistant to a number of vulnerabilities, but does suffer from the exit problem. It also faces other issues: how does one acquire an initial quantity of trunits? The obvious solution is to give a new seller an initial quantity of trunits. Unfortunately, this opens the system to re-entry.[3]

Basic Trunits is vulnerable to ballot-stuffing (which allow additional truants to be created) and bad mouthing (which can hinder competitors' ability to sell), as well as other Sybil effects. Basic Trunits is also vulnerable to another problem not noted above, called *surplus trust*. Each time a seller executes an honest sale, he gains additional trunits. If his sales are constant, however, he may not need these extra trunits to conduct this honest business. Thus, he can cheat with these extra trunits, without consequence.

[3] A solution to the startup/re-entry problems is provided by Commodity Trunits [43], an extension to Basic Trunits that allows trunits to be purchased and sold. Commodity Trunits requires the operation of a separate trunits marketplace, however, which introduces complicating factors that make it less appropriate for this investigation. Beyond re-entry, it also shares a very similar vulnerability profile to Basic Trunits; for this reason, we consider only Basic Trunits here.

4.3 Experimental Method

Our experiments are performed by marketplace simulation, using the TREET testbed described in Chapter 3. Except where noted, the default parameters specified in Chapter 3 are used.

A single TRS is in use in each simulation run. There are four sets of agents in the market during each run: buyers(100), honest sellers (250), randomly cheating sellers (250, each assigned a different probability of cheating, drawn from a uniform distribution over [0,1)), and agents implementing the cheating strategy we wish to evaluate (250). This mixture ensures a variety of agents for buyers to encounter, provides sellers with competition from sellers using different approaches, and provides comparison groups to evaluate performance.[4]

Each round consists of one day. After entering into a sale, a buyer will not know whether or not he has been cheated until after some number of days (14) has passed, reflecting processing, shipping, etc; we refer to the rendering of feedback after this *lag* (14 days) as the *completion* of the sale. At the beginning of each day, buyers discover whether each completing sale was executed honestly or not. Because we seek to validate attacks on TRSes (and we concern ourselves here only with attacks mounted by sellers), we wish to evaluate their effectiveness in the worst-case scenario (i.e, the best case for the TRS). Thus, in those systems in which buyers report their experiences with sellers, they always do so honestly.

Sellers decide what products to offer, and publicly post those offers. No limits are placed on sellers' capacity or inventory. For each product that it needs to buy, a buyer can evaluate each of the offers (i.e., evaluate the trustworthiness of each seller, using the system in place), before making a selection.

Sellers are informed of accepted offers, and paid. Each seller at this point decides whether or not to be honest. If she is honest, then she incurs the cost of furnishing the product (i.e., it is 'shipped out' that day). If she is dishonest, we assume maximal cheating: no good is shipped, and no cost is incurred. The buyer will learn the results after the lag has lapsed.

Marketplaces are often dynamic—traders join and leave regularly. This is important for TRSes, because new sellers are unknown, and departing sellers result in obsolete

[4]It might be argued that cheaters comprise too great a fraction of our marketplace. We note, however, that: a) if cheating is successful, then cheating will be encouraged, resulting in high rates of dishonesty; b) the protection offered by existing proposals should not be fragile in the face of the very cheating against which they are meant to defend.

knowledge. For efficiency of simulation, agents join/exit the market at specific intervals (100 days). On each day, each agent departs the marketplace with a fixed probability (0.05). That said, we do not want the effectiveness of our systems to be clouded by changes in market size (e.g., profits increasing because the number of buyers increases.) Thus, for every departing agent, one agent joins, keeping the participant count constant. Note that only buyers and honest sellers join and depart by this mechanism; dishonest agents stay to try to continue cheating, opening and closing accounts as their strategies dictate.

Attacker model

Here, we specify additional key points regarding the capabilities of sellers. At the time of making an offer, sellers do not know or control whether an offer will be accepted, or by whom. A seller can only provide products that she is able to produce. She is able to *advertise and offer* (dishonestly) any product, however. Sellers can freely create new accounts at will. An unavoidable consequence is that the same seller can control and operate multiple accounts at the same time. (Note that even if a seller opens multiple accounts, the set of products she can produce remains unchanged.)

TRS Implementation

All of the systems were implemented essentially as described by the authors, except as noted here. First, while the authors of these systems describe the calculation of reputation scores (for example, the expected probability of honesty), some do not describe the actual usage of this value in selecting a seller. In these cases, we have made the reasonable assumption that buyers choose the sellers with the highest scores (e.g., the one with the highest probability of being honest). Second, while these proposals specify how to combine ratings from multiple reviewers, which reviewers to solicit (e.g., the neighbors of an agent) may not be specified. To ensure the toughest tests for our attacks, and to investigate the soundness of each system's underlying evaluation of potential trustees, we assume the ideal case of complete connectivity: every agent receives reviews from every other agent (and can discount/disregard them as desired).

Reviewers may lie. Some of these systems make more of an effort to deal with the reliability of ratings than others—indeed, this has been a specific focus of research effort. As above, we assume the ideal case: buyers are perfectly honest in their reports to one

another. If a system is to be resistant to manipulation, it should certainly be so when noise/deception is eliminated from reviews.

Beyond what is specified here, where models require parameters we have used numbers provided by the authors in their own works wherever possible. Where no such numbers are provided, we have used reasonable values. Our TRAVOS implementation makes use of its integrated system for evaluating reviews. By comparison, in our BRS implementation all reviewers are considered to be reputable (i.e., reviews are not discounted), for several reasons: a) buyers and sellers are separate, and it was not clear how reputation scores for buyers might be established; b) all reviewers are honest, in our tests; c) this provides contrast with the TRAVOS system, allowing us to investigate the impact of attempting to cope with dishonest reviews.

4.4 TRS performance in the 'normal' case

First, we consider the performance of the TRSes in the 'normal' case, where simple attackers make no effort to exploit vulnerabilities in the TRS itself. Figure 4.1 depicts the operation of TRAVOS over time in the situation typically used for evaluation: where simple sellers cheat randomly. (The chart depicts one simulation run. Lines represent the total sales (in dollars) and profits for each group of agents for each day, smoothed for presentation by taking totals at 30 day intervals; profits = sales − costs incurred by the seller. Profit denotes the amount of money earned by sellers—indicative of the motivation to engage in the particular behaviour. Sales figures represent the amount spent by buyers with sellers in the particular behaviour group.) In this situation, the model operates 'as it should'—cheating quickly drops to very low levels, relative to honest sales. It is important to note the lines for profit. Recall that there are equal numbers of agents in each group of sellers. If the total profit for cheaters were higher than that for honest sellers, on average an agent would make more money by cheating than by being honest—dishonesty would be encouraged. Here, however, honesty is more profitable than cheating.

The performance for the other TRSes is similar in this situation. Throughout the chapter, we omit separate charts for each TRS, for the sake of brevity. Instead, we provide key data in numerical, tabular form, using charts only where illustration is informative. All tables report results over the second half (days 501 - 1000) of each simulation—after convergence to reasonably stable levels of sales for honest/dishonest sellers, reflecting long-term behavior.

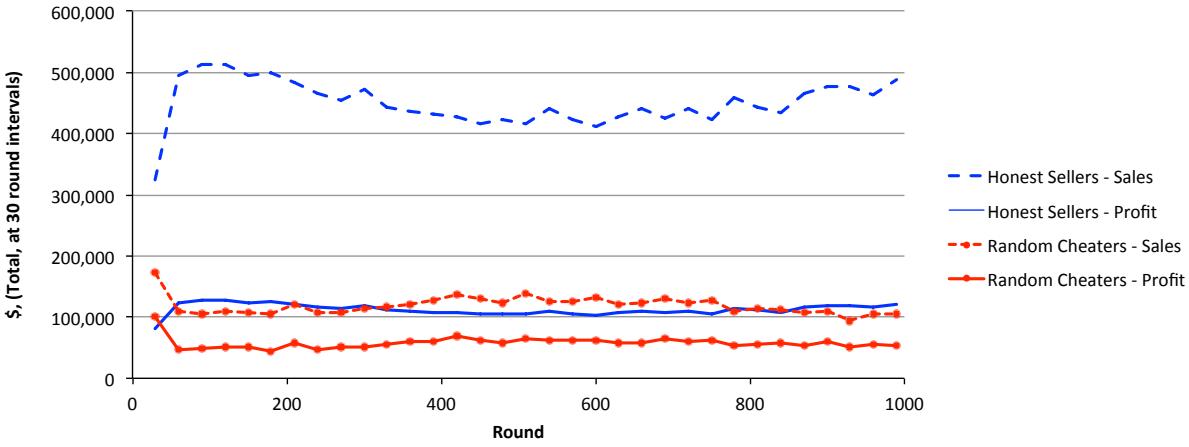

Figure 4.1: TRAVOS against randomly cheating sellers

Table 4.1 reflects the operation (over ten trials) of all models when faced with simple randomly-cheating sellers. In this table (and similar tables throughout the chapter), the middle column ('Cheater profit') represents the average profit per cheating agent, relative to those of an honest seller. Positive percentages mean that on average, a cheating seller earns more than an honest seller; for example, +25% would indicate that on average, cheaters make 25% more profit than honest sellers. Similarly, negative numbers indicate that on average, a cheater earns less than an honest seller.

On any trial, it would be troubling if cheating agents earned more than honest ones—this would suggest that a profit-maximizing agent should choose to cheat rather than be honest. The last column of the table ('Trials failed by TRS') indicates the percentage of trials where the TRS 'failed': where cheaters earned more, on average, than honest agents.

As with Figure 4.1, Table 4.1 shows the systems largely working as intended: honesty is more profitable than (random) cheating. Several points should be noted, however. First, cheating is not completely eliminated by any of the models. A key reason for this is the departure of known agents, and the entrance of new agents, which hinders the ability of systems to learn who (not) to trust. In addition, it is sometimes the case that for a given product, there may be no honest seller available. Moreover, some cheating agents will have very low (randomly assigned) cheating probabilities, and may not cheat frequently enough to be removed from consideration.

Basic Trunits fares extremely well here, reducing cheating to very low levels. This

Table 4.1: Sales/profit (per capita) for randomly cheating sellers, compared to honest sellers.

TRS	**Cheater profit (relative to honest)**	**Trials failed by TRS (% of 10 trials)**
Tran & Cohen	+58.39%	100%
Beta	-47.38%	0%
TRAVOS	-15.76%	10%
Yu & Singh	-8.80%	0%
Basic Trunits	-85.65%	0%

is because cheating agents quickly remove themselves from the market by losing their trunits. In contrast, the Tran and Cohen model fares badly here. This is a particularly difficult test for this model; relying only on direct experience, the model has trouble coping with both the turnover of agents, and the large proportion of cheaters in the market. (It also does not cope well with a large variety of products.)

These results validate both the operation of the models as expected, and the simulation scenario.

4.5 Attack Implementation

While a number of vulnerabilities have been theoretically identified, they do not constitute attacks in themselves. Rather, an attack is an actual method that exploits these vulnerabilities. As noted below, we may think of attacks as plays: sequences of events, with a desired outcome. An attack may take advantage of multiple vulnerabilities. In this section, we outline attacks that we constructed, evaluate their effectiveness, and discuss several issues raised.

It must be noted that our system can provide existential evidence only. Success in employing a tactic implies that a vulnerability exists in the system. Failure, in contrast, does not mean that no vulnerability exists, only that our agents as implemented did not successfully exploit one.

4.5.1 Playbooks

Our work employs a technique presented previously (e.g., [6, 60]), that of a *playbook*. Our agents seek to employ profitable strategies (cheating or otherwise), strategies consisting of sequences of actions. Unfortunately, there is an enormous number of possible sequences of actions that an agent might execute; it is difficult to learn strategies from amongst the set of all possible arbitrary sequences. As a solution, two proposals suggest the use of *plays*: pre-defined sequences of actions. Agents would have a 'book' full of known plays; selecting which play to employ at any given moment becomes the problem. This, too, presents an issue: the difficulty in directly specifying policies for which play to employ at which time, due to the enormous state space of the scenario. Each proposal takes a different approach to this issue.

The work of Ros et al. [60] employs Case-Based Reasoning. For the given scenario, a number of important features are defined that express aspects of any given state. Then, a number of example situations are created, consisting of a set of feature values and the correct choice of play for that situation. To select the appropriate play for a real situation, the agent chooses the play whose situation has the highest similarity score to the current state.

In contrast, Bowling et al. [6] suggest a technique which attempts to choose good plays for achieving a desired outcome. For each play, we track the number of times it has been executed, and the reward that has been earned each time it has been used. To select a play, a probability distribution is calculated over each applicable play: the probability of choosing a play is proportional to the reward earned using it in the past.

While our work makes use of plays, we were able to cheat very effectively without needing to use techniques such as these to choose between plays; our approach is detailed in Section 4.6.5.

Many of the attacks described below take input parameters. For example, when executing the reputation lag attack, for how long should one be honest, and then for how long should one cheat? One might envision using a learning algorithm to optimize these parameter values during execution. This is a tricky optimization problem, however, for a number of reasons. First, we have sparse data. We can gain very few samples of profitability at various parameter settings while the market is running. Second, the data is noisy. As shown in the charts above, sales and profits move in apparently random manner. Third, the function for which we are trying to optimize is constantly changing, as buyers are constantly updating the reputation values. We note, however, that we our goal was not to develop damage-maximizing behaviours, but simply to establish the exis-

tence of practical attacks. For such existential proof, simpler approaches were sufficient, as demonstrated in the reminder of this chapter.

For these reasons, and given that this was the first attempt at implementing such attacks, we instead arbitrarily chose seemingly-reasonable parameter values. It is, perhaps, telling that our attacks were successful with simple behaviors, arbitrary parameters set, and no attempt (even by hand) to optimize parameter values.

4.6 Single-agent Attacks

In this section, we focus on attacks that can be launched by a single entity (although possibly making use of multiple accounts.) Such attacks are simpler, in the sense that coordination and cooperation between entities is not required.

4.6.1 Sybil attacks

As explained in Chapter 2, Sybil attacks [15] consist of the creation of many false accounts by a single actor. In general, it is not possible to prohibit the creation of multiple accounts, so the impacts of Sybil accounts must be carefully considered. Sybil accounts are typically used to alter reputation scores (using a tactic such as ballot-stuffing or bad-mouthing). We begin our consideration of the larger issues of ballot-stuffing and bad-mouthing (whether executed by coalitions, or by single agents using Sybil accounts) later in this chapter. Here, we outline two other important ways in which Sybil accounts can be used, which we identified during experimentation.

The Proliferation Attack

When faced with multiple sellers, each with the same rating (specifically, the maximum across sellers), TRSes will typically choose randomly/arbitrarily between the sellers. To gain an advantage, a seller simply opens a multitude of accounts, and attempts to sell the same products through each of them. Consider the case where a product has only two sellers, both unknown to the buyer. If each makes one offer, then each has a 0.5 probability of winning the sale. However, if a dishonest seller offers the product through nine separate accounts, his probability of winning the sale increases to $9/10 = 0.9$. This might not consistute 'cheating', in the sense of a seller cheating a buyer—in fact, the

attacker gains an advantage even if she honestly provides the purchased good to the buyer. Rather, the seller cheats other sellers out of potential sales, which most would also consider to be problematic. (This form of dishonesty is the basis for many attacks on TRSes, including collusive attacks such as ballot-stuffing and bad-mouthing.) We call this attack *proliferation*, after the marketing notion of 'product proliferation': having more products on the shelves results in more sales.

This attack is extremely simple to launch, and extremely effective. Figure 4.2 depicts the results of this attack against BRS. (The sales revenue for random cheaters has also been included, for comparison, but random cheaters' profit has been omitted to avoid clutter.) The attack is devastating, with attackers dominating the marketplace.(Remember that the number of honest and attacking agents is the same, with the same product distribution.) As shown in Table 4.2, this attack is extremely successful against all systems tested.

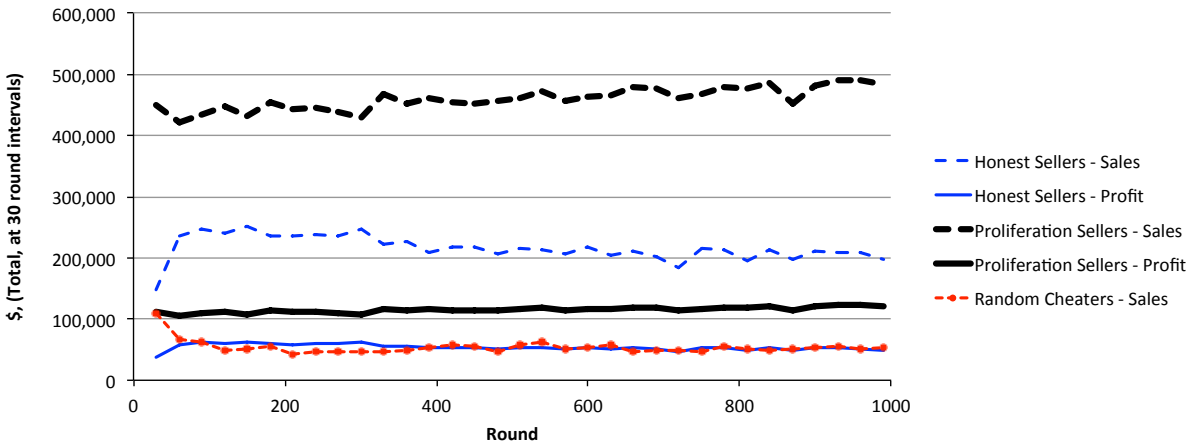

Figure 4.2: BRS, vs. Proliferation

The Countermeasures Effect

This attack sheds light on another use for Sybil accounts, which we have not seen noted, one we call *Countermeasures*. Many TRSes rely on gaining experience with honest sellers, to identify safe buyers. Even if cheaters behave naively, this can be a difficult process. However, an agent can employ this proliferation idea to further complicate the process. Consider again the situation where there is one honest seller of a product, and one

Table 4.2: Sales/profit (per capita) for sellers using Proliferation.

TRS	Cheater profit (relative to honest)	Trials failed by TRS (% of 10 trials)
Tran & Cohen	+400.70%	100%
Beta	+245.98%	100%
TRAVOS	+226.09%	100%
Yu & Singh	+435.69%	100%
Basic Trunits	+411.20%	100%

dishonest one, but this time, the cheating seller plans not to deliver the product. If each offers the product once, then the buyer has a 0.5 chance of picking the honest seller. If this happens, not only does the buyer benefit from the honest sale, but he gains information—he now has made progress in identifying a trustworthy seller. This will undermine the cheater's efforts in the future. But now, consider the case where the seller offers the product through nine separate accounts. Now, not only does the buyer have a 0.9 chance of being cheated, but he also has only a 0.1 chance of identifying the honest seller! This makes it much more likely that the seller will be able to continue cheating in the future. Worse still, the seller doesn't even gain any useful information about which sellers to avoid, because the seller will simply open another account, and abandon the old one.

The Countermeasures effect is incorporated into some of our other attacks, noted below.

4.6.2 The Reputation Lag attack

In this attack, the seller behaves honestly for a period (45 days), and then cheats for a period (15 days—the 'lag' before an act of cheating impacts reputation). After the cheating period, the seller abandons the accounts, and opens new ones. This attack takes advantage of reputation lag and re-entry, in particular.

Figure 4.3 depicts the use of this attack against the model of Tran and Cohen [76]. In this figure, each series represents the moving average over two intervals, rather than the raw total for each interval; this smooths out oscillations due the periodic honesty, then cheating, of the sellers. Two features are evident in this chart. First, the performance

of the system improves over time, as trustworthy sellers are found; in fact, the sales of honest sellers quickly surpass those of the Reputation Lag sellers. But second, and more importantly, profits of the Reputation Lag sellers remain higher than those of the honest sellers. This reflects the fact that the cheaters have higher profit margins than honest sellers, because they incur no cost when they fail to deliver a product.

As shown in Table 4.3, BRS and TRAVOS fared better than expected against this attack. On early iterations, cheaters were successful. It appears that on subsequent iterations, both systems had already identified trustworthy sellers, which should occur quickly in a (perfect) witness information model. These known good sellers were preferred over unknown (re-entering) ones. TRAVOS does fare worse than BRS, however, a pattern we will notice throughout our results, despite their similarities. TRAVOS agents only trust each individual reviewer's reports to the degree that the reviewer has shown itself to be reliable. Where reviewers may lie, this would likely prove beneficial. Here, where all agents report honestly, it slows the acceptance of useful information.

Basic Trunits performed poorly against this attack. This might be puzzling, as normally Trunits is resistant to reputation lag—trunits for each transaction are placed in escrow pending completion of the sale. Re-entry is a big part of the attack, however, and this implementation of Basic Trunits is quite vulnerable due to the initial sum of trunits provided to new sellers.

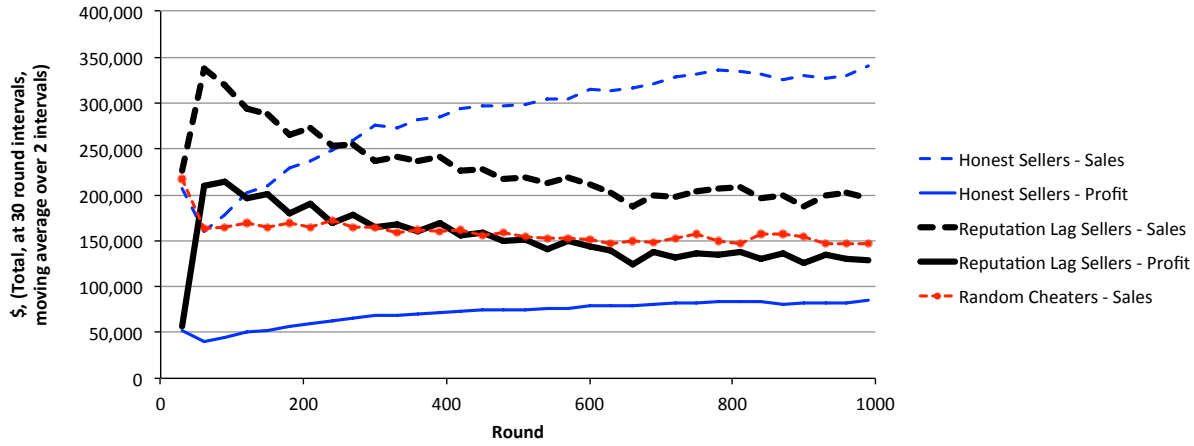

Figure 4.3: Tran and Cohen, vs. Reputation Lag

Table 4.3: Sales/profit (per capita) for sellers using Reputation Lag.

TRS	Cheater profit (relative to honest	Trials failed by TRS (% of 10 trials)
Tran & Cohen	+138.40%	100%
Beta	-3.87%	20%
TRAVOS	-8.25%	20%
Yu & Singh	+315.27%	100%
Basic Trunits	+85.88%	100%

4.6.3 The Re-entry attack

This attack is similar to the one above, except the agent never attempts to be honest. He simply opens an account, uses it to cheat for a period, then abandons it to open another. This attack is intended to exploit those systems that allow unknown sellers to trade effectively.

The execution of this attack, against the Yu and Singh model [83], is depicted in Figure 4.4. (Note that the profit and revenue lines for the Re-Entry sellers are superimposed in this chart—the cheaters never execute an honest transaction, so they incur no cost.) The results against all systems are shown in Table 4.4. This attack is very successful against every system, even those that defended against reputation lag. A key reason for this is the countermeasures phenomenon cited above. Since each cheating seller has no intention of delivering the product, he offers every product for sale (even those he cannot produce). This prevents the buyers from identifying honest sellers that they can rely on in the future. This is particularly hard on Tran and Cohen, where each buyer must find honest sellers by individual experience.

4.6.4 The Value Imbalance attack

In this attack, the seller attempts to be honest on small transactions to gain reputation, then cheat on large ones to gain extra profit. Unlike the previous attack, this is not periodic. Instead, the seller attempts to maintain a minimum threshold ratio of honest sales (75%) to dishonest sales, with the idea of maintaining a reasonably high level of reputability throughout.

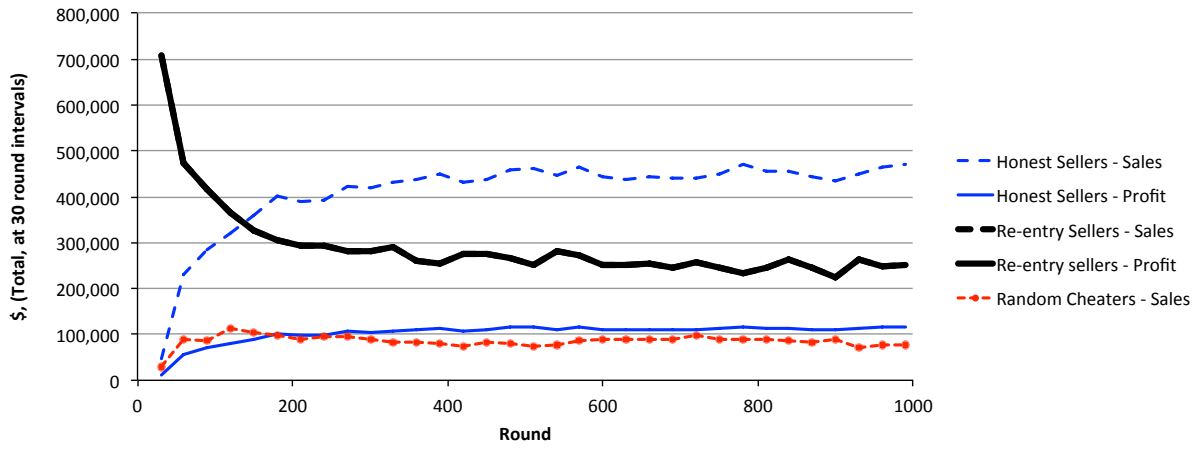

Figure 4.4: Yu and Singh, vs. Re-entry

Table 4.4: Sales/profit (per capita) for sellers using Re-entry.

TRS	**Cheater profit (relative to honest)**	**Trials failed by TRS (% of 10 trials)**
Tran & Cohen	+5,658.44%	100%
Beta	+242.93%	100%
TRAVOS	+235.25%	100%
Yu & Singh	+253.13%	100%
Basic Trunits	+347.01%	100%

This attack is successful against BRS (Figure 4.5): cheating is somewhat more profitable than honesty (and with much lower investment/sales volume). Further, as noted above, this attack was based on arbitrary parameter settings. It may fare even better with tuning of the parameters. Table 4.5 shows this attack to be effective against most systems. Basic Trunits fares well because each update is proportional to the value of the sale.

Here, the difference between BRS and TRAVOS is even more pronounced. Beyond the effect noted above, it appears that another factor is at play here. Once a TRAVOS agent has enough experience with that seller, it relies only on that direct experience. This appears to slow their response when agents' behavior changes—an agent does not learn from others' warnings when a seller has begun to cheat, and so must learn it directly.

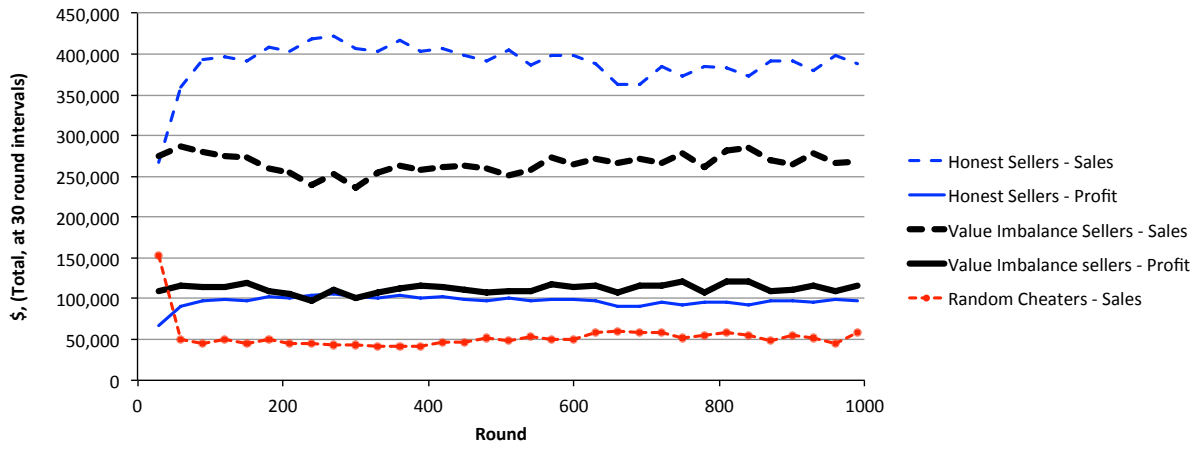

Figure 4.5: BRS, vs. Value Imbalance

Table 4.5: Sales/profit (per capita) for sellers using Value Imbalance.

TRS	**Cheater profit (relative to honest)**	**Trials failed by TRS (% of 10 trials)**
Tran & Cohen	+231.18%	100%
Beta	+51.72%	100%
TRAVOS	+182.03%	100%
Yu & Singh	+194.38%	100%
Basic Trunits	-51.37%	0%

We believe this is compelling evidence that these techniques are practical: every system tested was vulnerable to multiple attacks that made cheating more profitable than honesty.

4.6.5 Security by Obscurity?: The Multi-tactic Agent

While any given attack described above can be launched successfully against certain TRSes, it may be less effective against others. Researchers might be tempted to suggest that, because a seller might not know what system is in use, she might not be able successfully select an effective tactic. Here, we hope to dispel this notion. The question

is, can an agent successfully manage a portfolio (or playbook) full of attacks without knowledge of which TRS is in use?

We had initially intended to use learning techniques to choose between attacks (as noted in Section 4.5.1). We found a much simpler approach to be effective, however. We note two important points. First, as noted above, accounts cannot be tied to real identity, so an agent is free to open multiple accounts. There is no reason why an agent cannot open several accounts simultaneously to launch several attacks. Second, there is little to lose in launching an unsuccessful attack. Sellers offer goods for sale, which the buyers may or may not select. If a seller is seen as disreputable, he does not suffer any direct financial penalty—being bypassed for sales is the indirect penalty. Thus, there is no reason the seller cannot keep multiple accounts open, using each one for a different attack in parallel. The successful attacks generate profit, while the unsuccessful ones essentially result in dormant accounts. Hence, we do not need to choose between attacks—the more successful attacks will generate more activity on their own.

In implementing this method, we used all of the single-agent attacks (reputation lag, re-entry, value imbalance) except proliferation—it was so successful on its own, it would have rendered the results meaningless. The execution of the suite of attacks together against Basic Trunits is depicted in Figure 4.6; the results against all systems are shown in Table 4.6. In every case, the profitability from cheating is dramatically higher than honesty. This is an extremely important result: every system considered could be soundly defeated, without knowledge of the TRS in use, by employing straightforward tactics with no special optimization. This is a clear indication that security requires more attention from researchers.

Table 4.6: Sales/profit (per capita) for Multi-tactic sellers.

TRS	**Cheater profit (relative to honest)**	**Trials failed by TRS (% of 10 trials)**
Tran & Cohen	+7568.62%	100%
Beta	+367.06%	100%
TRAVOS	+587.42%	100%
Yu & Singh	+782.37%	100%
Basic Trunits	+874.68%	100%

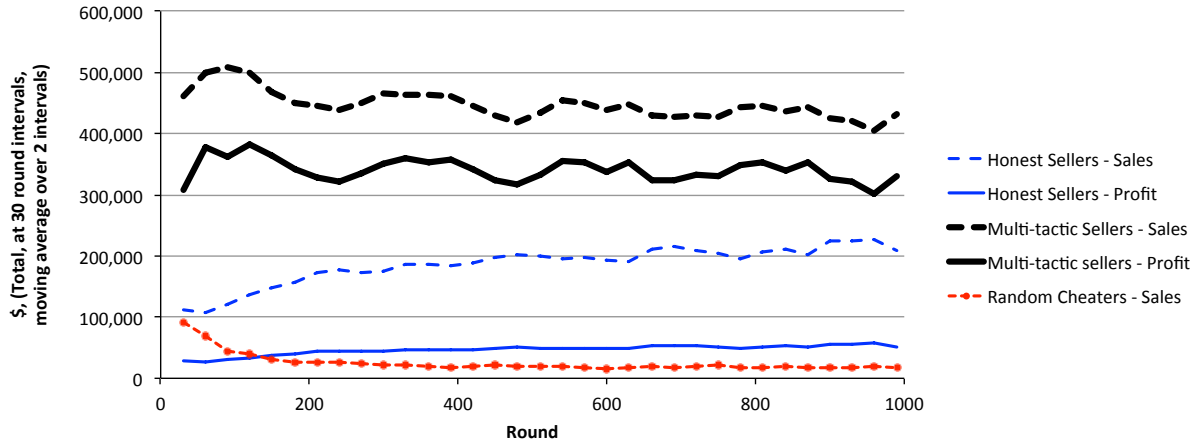

Figure 4.6: Trunits, vs. Multi-tactic

4.7 Coalition Attacks

In Section 4.6, we considered attacks that could be launched by a single entity. Now, we consider another class of attack: those that may be launched by multiple entities working in cooperation. Cooperation, in itself, is not necessarily bad; indeed, there are many scenarios in the real world in which cooperation is welcome and encouraged. In contrast, we focus on cooperation which is not welcome or well-intentioned: coordinated activity intended to thwart the operation of TRSes. We consider the two well-known forms of collusion used to target TRSes: ballot-stuffing and bad-mouthing.

As noted above, ballot-stuffing and bad-mouthing can be executed by a single entity making use of Sybil accounts. Here, we are concerned with the *potential* of an attack to be successfully launched; whether the cooperating accounts are owned by one individual, or many, has little bearing on the potential of the attack. (One might consider a set of Sybil accounts to simply be a very well-coordinated coalition.) For this reason, we consider the use of Sybils to be simply a special case of coalition activity.

Coalition attacks such as these require the involvement of both buyers and sellers—buying accounts are used to attempt to manipulate the reputation of selling accounts, while the selling accounts are used to earn money by executing sales. For this reason, the test configuration in this section differs slightly from that above: coalition members control accounts that are used for both buying and selling. The buying behaviour of coalition members is explained in the attack descriptions, below.

4.7.1 Ballot-stuffing

The Ballot-stuffing agents attempt to inflate the reputation of other coalition members by engaging in fake transactions with teammates, and giving positive reviews. As with other buying agents, the coalition members are assigned a set of products that they legitimately need to buy each turn. Some of the products needed will be available from coalition members, but many will not. To make these purchases, the members use the current TRS in the same way as any other buyer: determine who is the most trustworthy, according to the TRS, and buy from that agent. In addition to these needed purchases, however, coalition members engage in an additional number of fake, ballot-stuffing purchases with teammates. (In this test, ballot-stuffing purchases constituted an additional 50% of purchases, beyond legitimate needs.)

Apart from the ballot-stuffing activity, coalition member are completely 'honest'. When making legitimate purchases, they give honest reviews. When selling, they faithfully deliver the good to the purchaser. Their only 'dishonest' activity is the attempt to win additional sales by using false reviews.

Figure 4.7 depicts the performance of the TRAVOS system against ballot-stuffing. One point should be noted. The sales revenue for Ballot-stuffers is extremely high, dwarfing all of the other series. This is a result of the fake, ballot-stuffing transactions, which are included in the sales figures (in addition to the 'real' sales). Ballot-stuffing transactions do not directly result in profit, however, so the profit series accurately reflects the gains earned by the ballot-stuffers—which are substantially higher than those of honest sellers.

Table 4.7 illustrates the performance of all systems when faced with ballot-stuffing. (Note that the Tran and Cohen model is omitted from this table, and that of the following section. Because this model relies only on direct experience, there is no 'system' which can be ballot-stuffed, making this attack inapplicable.) All of the systems fair poorly—despite the fact that every sale is honestly fulfilled by the cheaters. Beta, which continues to rely on witness information throughout its operation, suffers worst from this attack. TRAVOS and Yu and Singh, by comparison, perform somewhat better because buyers rely more on their own experience as it is gained. Trunits also fairs somewhat better because the additional trunits earned by ballot-stuffing only allow sellers to engage in more simultaneous sales; this advantage does not directly allow them to win sales from honest sellers who also possess sufficient trunits.

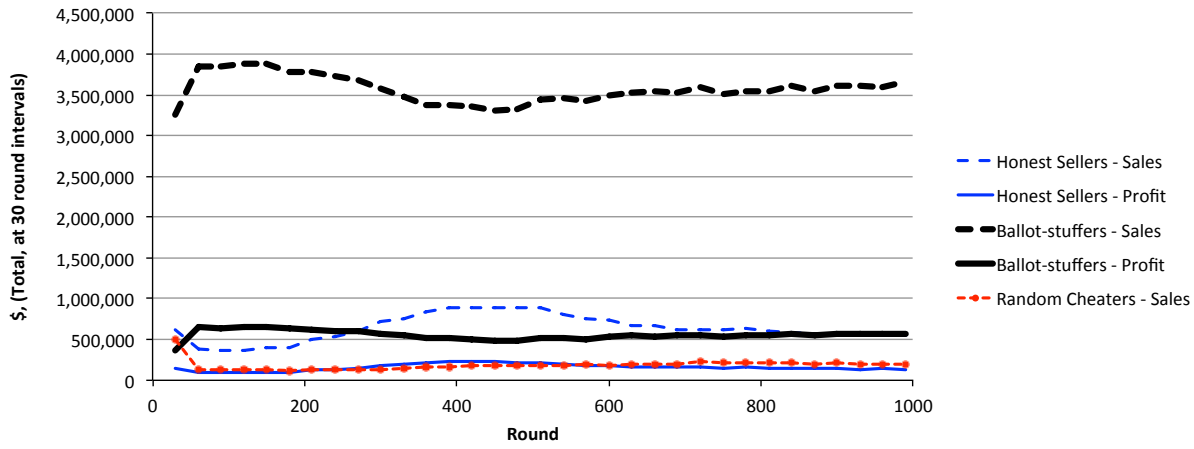

Figure 4.7: TRAVOS, vs. Ballot-stuffing

Table 4.7: Sales/profit (per capita) for Ballot-stuffers.

TRS	**Cheater profit (relative to honest)**	**Trials failed by TRS (% of 10 trials)**
Beta	+905.30%	100%
TRAVOS	+382.83%	100%
Yu & Singh	+150.01%	100%
Basic Trunits	+164.38%	100%

4.7.2 Bad-mouthing

The bad-mouthing agents behaved similarly to those described above. However, no additional, 'fake' purchases were made for bad-mouthing purposes. Instead, the bad-mouthing agents simply purchased the products they actually needed, but gave (falsely) negative reviews to non-members. This approach is the most conservative for bad-mouthing—the agents do not incur the cost of making 'fake' bad-mouthing purchases of unneeded products—but also limits the number, and hence the damage, of bad-mouthing reviews. (Additional, fake purchases are certainly an option, however.) It may be surprising then, that even this conservative approach constitutes such a devastating attack.

Figure 4.8 illustrates the performance of Beta against the bad-mouthing agents. This attack is even more profitable than the previous one; the impact of the negative reviews

seems to be greater than (additional) positive reviews. As can be seen in Table 4.8, all systems were catastrophically impacted. Trunits suffers particularly badly; in this Basic Trunits implementation, a seller who has lost his trunits cannot acquire more, so bad-mouthing can effectively remove competitors from the marketplace entirely.

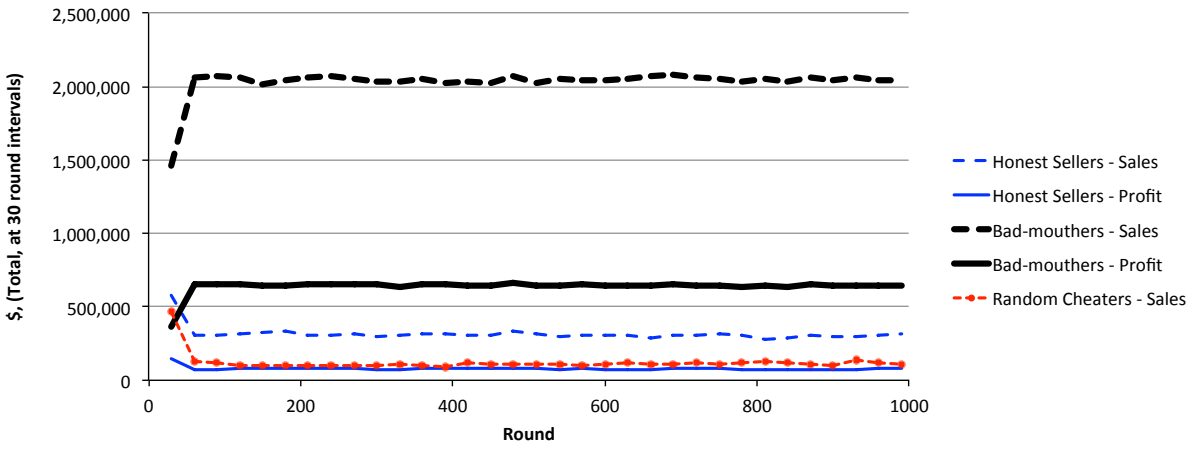

Figure 4.8: BRS, vs. Bad-mouthing

Table 4.8: Sales/profit (per capita) for Bad-mouthers.

TRS	Cheater profit (relative to honest)	Trials failed by TRS (% of 10 trials)
Beta	+1,007.22%	100%
TRAVOS	+433.00%	100%
Yu & Singh	+835.97%	100%
Basic Trunits	+129,119.18%	100%

4.8 Conclusions

We implemented a number of cheating attacks that prey upon the vulnerabilities in trust and reputation systems, vulnerabilities that for the most part, had previously been identified only in theory. These attacks are composed solely of conventional transactions that

are permissible within a marketplace, and that could be executed by virtually any trader. To the best of our knowledge, this is the first demonstration that multiple vulnerabilities do exist, and can be exploited in practice. Moreover, while we have selected a small number of TRSes for this study, we have no reason to believe that other systems will prove impervious to these attacks. Indeed, the survey presented in our earlier work [36] suggests that these vulnerabilities are ubiquitous. One might find surprising the ease of execution, and the effectiveness of the attacks. If such attacks can be launched so easily, we must expect that traders can and will take advantage of them. If attackers can thwart the protections offered by TRSes, then the real-world value of such systems is undermined.

Certain TRSes are resistant to certain attacks. The question must be asked, can an agent manage a playbook of attacks in order to be successful against a system, without advance knowledge of the system in place? For the first time, we have demonstrated that the use of a simple technique allows effective cheating without any knowledge of the system in use; this technique was extremely successful in breaching the defenses of every TRS tested.

4.8.1 Moving Forward: The Issues of Coalitions and Collusion

Our results strongly suggest that the security of trust and reputation systems (and certainly these specific vulnerabilities) require much more attention from researchers. Developing a perfectly secure system, one that addresses all such vulnerabilities, is an open problem, and beyond the scope of this document. Instead, throughout the remainder of this work, we focus our attention on the specific issue of coalitions, and on the types of attacks that coalitions may execute.

As demonstrated in this chapter, individual agents are certainly capable of launching successful attacks. Despite this, we have elected to focus on coalition behaviour, for a number of reasons:

- First, for each of the single-agent attacks, reasonable approaches have been proposed (e.g., [16]) to deal with the vulnerability. For example:
  - To limit the attractiveness of Sybil attacks, one might charge fees for the creation of accounts [45].
  - To prevent value imbalance attacks, one can tie the impact or weight of each review to the value of the transaction (e.g., [37]).

  - To combat reputation lag, one can ensure that some 'portion' of the seller's reputation must be 'allocated' for the duration of each sale, so that the same reputation cannot be used to simultaneously entice many buyers into making purchases [37].
  - To hinder re-entry, one can ensure that new accounts are treated similarly to maximally-disreputable accounts (e.g., [87, 43]).

  Certainly, it is true that no system has yet successfully incorporated protection against all such vulnerabilities, but tools such as these may help yield a solution.

  In contrast, there has been very little progress towards solutions to coalition attacks, motivating us to focus on such issues.

- Most of the single-agent attacks are issues very specific both to the trust and reputation domain, and to specifics of TRS formulations. In contrast, as outlined in Chapter 1, cooperation and coalitions are widespread issues impacting numerous, quite varied domains. Techniques to address the issues of coalitions and collusion might have applicability far beyond trust and reputation, greatly increasing their value.

For these reasons, the remainder of this document focuses on coalitions and collusion. In the next chapter, we begin to develop techniques for detecting the presence of coalitions and identifying coalition members—a potentially-useful tool for many purposes, including coping with collusion in trust and reputation systems.

Chapter 5

Coalition Detection and Identification

As explained in Chapter 4, there are numerous vulnerabilities common to trust and reputation systems (TRSes). Some of those vulnerabilities, which can be exploited by individuals, tend to arise from specific properties of given TRSes, or features of the trust and reputation scenario itself. In contrast, those vulnerabilities that can be exploited by coalitions seem to be more broadly inherent to multiagent systems in general; as noted in Chapter 1, coalition activity can be an issue in a wide variety of domains. In an effort to initiate solutions to these important concerns, we turn our attention throughout the remainder of this document to coalitions and collusion.

In considering these issues, an instinctive response might be to attempt to prevent cooperation from taking place, or at least prevent it from succeeding. Throughout most of this document, however, we take a different approach. Given the many, varied scenarios impacted by coalitions, it seems unlikely that a single preventative solution would apply to all, or even several of them. Moreover, depending on the scenario, cooperation may be benign or even welcome—preventing such activity would be undesirable.

Instead, we focus here on developing techniques to *detect* coalitions and *identify* coalition members. This ability is more likely to have broad applicability and utility. Our technique is not specific to the trust and reputation domain. On the contrary, our methods are based on properties that are fundamental to the nature of cooperation itself, rather than characteristics of specific scenarios or systems. For this reason, we believe that our approach will be directly applicable to a variety of multiagent domains.

While the ability to detect and identify coalitions is likely to be valuable in its own right, it can also represent an important step towards taking further, domain-specific measures to deal with the issues raised by coalitions. For example, in Chapter 8 we

use the techniques developed over the next several chapters to implement a reputation model that is resistant to collusion.

In this chapter, we introduce the conceptual foundation of our approach, and develop the core algorithm used for coalition detection and identification. In subsequent chapters, we examine a number of enhancements and refinements to the algorithm, improving performance and addressing a number of special or difficult circumstances.

While we believe our techniques to be broadly applicable, a concrete scenario is required for explanation and demonstration. Accordingly, throughout this document we continue to make use of the marketplace trust and reputation scenario.

5.1 The Nature of Cooperation

As noted in Chapter 1, our goal is to develop techniques that are effective even when we have no advance knowledge of the strategies that coalitions might pursue: we cannot rely upon a known plan library. To accomplish this, we must make use only of observable actions, yet without matching them against known patterns—we must rely on more fundamental properties.

To gain insight, we first return to our discussion from Chapter 2, of the community finding problem in social network analysis. As noted, there are key differences between this problem and our own, and thus the measures typically used for community finding are not generally informative when trying to identify coalitions.

For example, connectivity is limited in its ability to reveal coalitions. Coalition agents are likely to be very well connected, having interactions both within and outside their coalitions—the graph of links between agents might even be complete. Similarly, one might look to frequency of interaction, under the assumption that the closer the relationship between agents, the greater the number of interactions between them. As a tactic such as ballot-stuffing illustrates, however, this may not hold: the very purpose of this tactic is to use a relatively small number of ballot-stuffing transactions to win many sales from outsiders.

Taking a step back from the specifics of trust and reputation, however, insight can be gained: *features such as these have limited benefit for coalition identification, because they have little connection to the fundamental nature of cooperation.* Cooperation is not simply a matter of having a relationship with someone, for example, or interacting with them.

We are much more likely to have success in detecting coalition activity, if we focus on characteristics central to cooperation itself:

1. **The nature of actions:** Cooperative actions tend to be *beneficial*, or *helpful*.

2. **The targets of actions:** Cooperative actions tend to *target teammates* with those benefits. The fundamental purpose of cooperation is the furthering of group goals, and/or, the individual goals of coalition members—*helping each other*.

Of course, in some circumstances these features may be subtle or complex. For example, cooperative actions may not directly benefit other members, but rather harm outsiders. Such actions, however, indirectly benefit coalition members—for example, weakening an opponent results in a stronger position for coalition members.

Because benefit seems to be essential to the existence of coalitions, we use it as the basis of our techniques.

5.2 Benefit

In the scenarios with which we are concerned, there are observable interactions between agents (or observable actions, which will have impacts on other agents). Having only these observable actions for information, we need to consider the benefit (or harm) implicit in them. For example, in the trust and reputation scenario, paying someone for a purchase, or giving a positive review might be an act of benefit. In an online game, giving an item to another player, or healing another player, might be acts of benefit. For any given domain, identifying and quantifying the degree of benefit implicit in each action is likely to require expertise in that domain; we discuss applicable domains, and potential measures of benefit, in Chapter 9.

5.2.1 Benefit Graph

With a set of agents, and interactions between agents, a digraph is a natural representation. An example of such a graph, for a simple hypothetical situation, is depicted in Figure 5.1. Each vertex is an agent, and each edge indicates benefit bestowed upon the destination agent by the origin agent, with the edge weight indicating the 'quantity' of benefit. (This might be, for example, the number of beneficial actions, or the value of the benefit. Again, determining appropriate measures of benefit likely requires domain expertise.) We refer to this as a *benefit graph*.

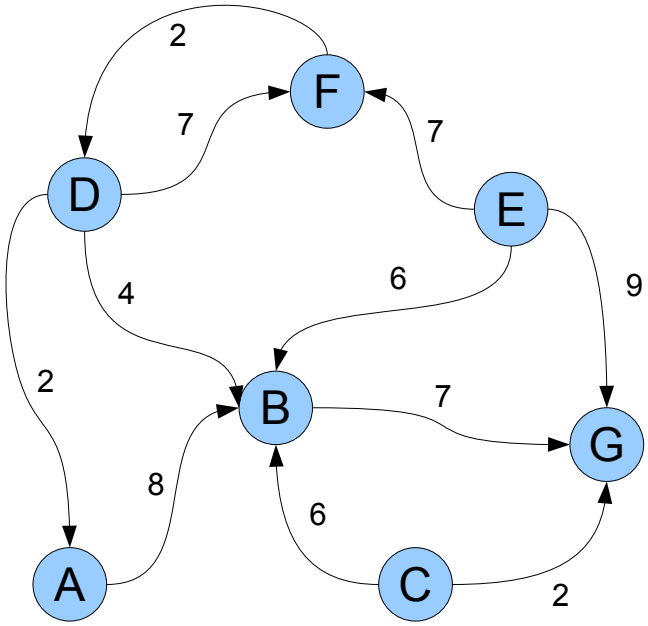

Figure 5.1: A benefit graph.

In this representation, graphical approaches immediately come to mind. Unfortunately, it appears that obtaining results directly from the graph may be challenging, for some of the same reasons noted in our discussion of community finding (Section 2.4).[1]

For example, in the situation depicted in Figure 5.1, although it is not directly observable (and hence, not depicted in the graph), agents A, B, and C are colluding: A and C are ballot stuffing to improve B's reputation, so that B can win additional sales.

It is not obvious that properties of the graph will reveal the presence of the coalition. The weights of edges between coalition members are not greater than between non-members. The level of activity between the coalition members is not greater than between other subsets of agents in the graph: for example, consider the set $\{B,E,G\}$, as compared to $\{A,B,C\}$. The presence of sources and sinks is not indicative of coalition membership. (The coalition contains no sinks; C is a source, but so is E). In fact,

[1] Although we consider a graph of benefit here, and community finding typically considers a graph of interaction, similar issues are encountered in attempting to work directly with the graph.

although A and B are teammates, and responsible for all of the collusive actions in this situation, they are not even connected in the graph. While we do not discount the possibility that useful insight can be gained by operating directly on the graph, we have found a different perspective to be effective.

5.2.2 Similarity of benefit

Typically, a self-interested agent will be part of a coalition because it believes it will receive benefit from doing so. (An agent might participate because, for example, it has been coerced to do so. It can still be argued, however, that there is benefit in such a case: the agent avoids the damage it might incur if it did not cooperate.) We presume that each agent is individually rational, and will only be part of a coalition if it expects a net benefit from doing so.

This does not imply that every coalition member will realize observable benefit. For example, due to circumstances, failures in coordination, etc., a plan may fail. More to the point, it may be the case that most or all coalition members benefit, but that not all of the benefit flows are observable—some benefit may be transferred privately between members. Considering our example, note that A and C do not receive any observable benefit from ballot-stuffing for B. Instead, it is likely that all three agents share in the proceeds of their activity, but the payments from B to A and C are not observable. (If B simply received benefit, and didn't make any contributions, then B would not be desired as a teammate by others in the group.) Note, however, that the agent making such payments is also rational—B must expect to accrue benefits at least as great as the payments it makes.

Similarly, we would not expect coalition members to do substantial harm to one another. Certainly, an agent might harm a teammate accidentally, or might do it to avoid the appearance that they are in the same coalition. Note, however, that the former case should not occur frequently, and in the latter case, the net positive benefits should outweigh the harm.

To be clear, we make two points:

- Members typically expect that the benefits realized by the coalition as a whole will be greater than the benefits received by the members if they had not coordinated their activity—if this were not the expectation, members may not wish to belong to the coalition.

  - If the activities of a coalition do *not* result in increased benefit to the coalition as a whole—i.e., the coalition is unsuccessful—then the coalition may not be of concern to those developing, operating, or participating in a multiagent system.

- While coalition members may be able to make unobservable transfers of benefit between members, it is assumed that the additional benefit accrued by collaboration is observable in some way.[2]

  In the situations with which we are concerned, this is a reasonable assumption. Coalitions seek to gain net benefit from or relative to outsiders (e.g., to earn additional profits from outsiders, or to improve competitive position relative to outsiders); improving the net position of the coalition requires taking observable actions (making sales, attacking enemies, etc.). For example, in the marketplace example, we cannot directly observe 'secret' payments between coalition members. We can, however, observe the actions that increase/decrease reputation, which is used to earn additional profits from outsiders.

Key insights

The key insights here are that, although coalition agents may interact often with outsiders:

- Coalition agents are likely to engage in actions that benefit other coalition members;
- Coalition agents are more likely to engage in actions that harm outsiders, than they are to harm coalition members;
- Because coalition members tend to help the same set of agents (coalition partners) and tend to harm the same set of agents (outsiders), *there is likely to be identifiable behavioural similarity in terms of benefit/harm.*

[2] This assumption does not imply that it will be *easy* to identify the actions that constitute benefit. For example, if attempting to identify coalitions in a battlefield scenario, identifying what constitutes beneficial, harmful, and incidental acts is likely to be difficult. Even so, the actions will still be observable.

5.2.3 *Benefit Space*

We define the *benefit space* as a high-dimensional space reflecting the degree of benefit (and/or harm) rendered to each agent in the system. Specifically, given N total entities in the system, the benefit space $\mathscr{B}$ is a space $\mathbb{R}^N$, where the value in each dimension β_i represents an amount of net benefit (i.e., total benefit minus total harm) to entity i. Positive values represent positive benefit, while negative values represent net harm.

Each entity maps to a point in the benefit space, according to the amount of net benefit it has rendered to each entity in the system. Thus, a given agent a can be reflected by the vector:

$$\mathbf{b}(a) \equiv (\beta_1(a), \beta_2(a), ...\beta_N(a)) \tag{5.1}$$

Members of a coalition are likely to be similar, in terms of the sets of agents that they benefit, and the sets that they harm. Thus, we would expect them to be close in this benefit space (even it they do not interact directly at all). Here, we have used a Euclidian distance as our dissimilarity measure, where the distance between a and b is:

$$d_{a,b} = \sqrt{(\beta_1(b) - \beta_1(a))^2 + \cdots + (\beta_N(b) - \beta_N(a))^2} \tag{5.2}$$

5.3 Algorithm

Overview

Our detection and identification algorithm, then, is a two step process.

1. First, we map agents into benefit space, which helps to reveal the structure of coalitions. We exploit similarity in behaviour to identify possible (or *candidate*) coalitions, using clustering.

2. Then, we use a statistical approach to characterize these candidate coalitions, to determine whether a cluster actually constitutes a coalition.

5.3.1 Clustering in benefit space

With this representation, it is natural to consider clustering in order to identify sets of agents that are close in the benefit space. Clustering is an unsupervised method of finding

sets of similar patterns; it has a long history in a variety of application areas [27]. There is likely to be much noise in a sample (for example, as agents do things that benefit those that are not part of its coalition, such as making honest sales to outsiders). Our results show, however, that collaborative activity does, in fact, produce detectable signal in many cases.

We illustrate this by returning to our earlier example, depicted in Figure 5.1. Recall that the weight of each edge represents the benefit rendered by one agent to another. First, we map each of the agents along one dimension, the benefit to agent B. This is portrayed in Figure 5.2. In this (and the following figures), the red crosses represent

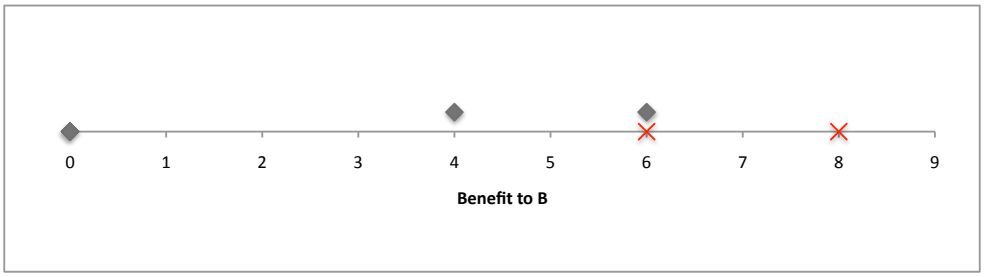

Figure 5.2: One dimension of the benefit space.

agents A and C, the ones making ballot-stuffing transactions; the remaining points represent the other agents. Note that A and C are close to one another in this dimension—they have both benefited B substantially. Other agents are also close to A and C in this dimension, however—D and E have also benefited B to a large degree. This is an example of the type of noise mentioned earlier. Without knowing the labels, we cannot separate the colluders from the outsiders in this dimension.

We can consider another dimension, this time benefit to agent F, depicted in Figure 5.3. In this dimension, there is even less separation between agents.

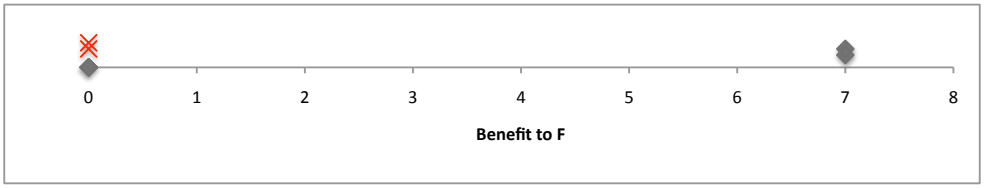

Figure 5.3: A second dimension of the benefit space.

If we consider multiple dimensions, however, the situation improves. Figure 5.4 depicts only two dimensions, both β_B and β_F, simultaneously. Here, clear separation be-

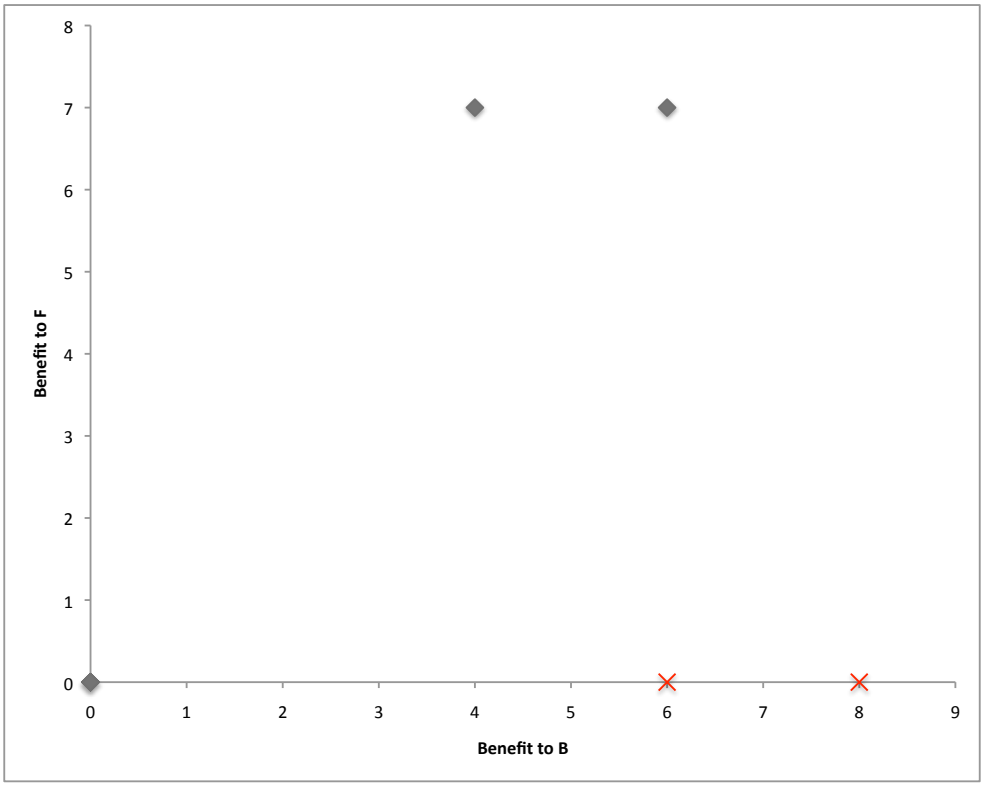

Figure 5.4: Two dimensions of the benefit space.

tween colluding agents and outsiders is evident. Agents A and C are similar enough to each other, and dissimilar enough to other agents, that a cluster stands out. Note that A and C can be identified as being potential teammates, despite the fact that there is no interaction between them, and no edge connecting them in the graph. Considering more dimensions can make the distinction between colluders and non-colluders even clearer. As long as there is enough separation in enough dimensions, a standard clustering mechanism can be used to identify possible coalitions.

Two points should be noted.

- First, although A and C are identified as being similar in Figure 5.4 (and thus, a possible coalition), B is not been included in this group: while it has been the

recipient of ballot-stuffing, it has not engaged in any transactions itself in these dimensions, so its behaviour is not seen as similar to A and C. Of course, in reality we would not limit our analysis to only two arbitrary dimensions, but rather consider all dimensions.

- Second, and related, is that a cluster may or may not be a true coalition. Considering Figure 5.4, we see another obvious cluster in the top of the chart, one that is not composed of colluders. This highlights that, once clusters have been found, further work is required to characterize the activity of each cluster: are cluster members cooperating, or do they, for example, simply have similar preferences?

5.3.2 The Clustering Step

A standard clustering algorithm is applied to find sets of agents that are close in benefit space, reflecting similarity in whom they favour. Here, we have used k-means clustering.[3] This results in a partitioning of the population P into a set of clusters $\{C_1, C_2, ..., C_n\}$, which we refer to as *candidate coalitions*. Each cluster is then characterized, as explained in the next section.

Determining the number of clusters

By the very nature of the problem, we are unlikely to know in advance how many (or even if) coalitions are present in the population. Certainly, this is valuable information for our algorithm to discover; moreover, it is a complicating factor because some clustering algorithms (including k-means) do not determine the number of clusters, but rather require the desired number of clusters to be set as an input parameter.

To determine the appropriate number of clusters[4], we iteratively run k-means with successively larger number of clusters k. (Here, we tried as many as ten clusters.) Then, for each resulting set of clusters, we compute the silhouette score [61] for the partitioning. Silhouette provides a measure of the quality of a clustering, without knowledge of

[3] We make use of some machine learning components, including the k-means implementation, from WEKA [23].

[4] Note that the 'appropriate number of clusters' does not necessarily equate to the number of coalitions present in the population. For example, there may be groups of participants that exhibit similar behavior, but which are not necessarily coalitions.

The 'appropriate number of clusters', then, is the number that best fits the natural structure present.

the hidden classes of the objects, by evaluating how well the clustering 'fits' the structure of the data. To calculate a silhouette score [61]:

1. For each object i, calculate $s(i)$ as follows:
   (a) Calculate $a(i)$, the average dissimilarity (distance) between i and all other members of i's cluster.
   (b) For every other cluster C, calculate $d(i,C)$, the average distance between i and members of cluster C; set $b(i)$ to the minimum such distance.
   (c) Then,
   $$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{5.3}$$
2. The silhouette score is the average $s(i)$ over all objects i.

Silhouette scores fall in the range [-1, 1]. For each object i, the closer i is to members of its cluster (i.e., the smaller $a(i)$ is), and the further it is from members of the next nearest cluster (i.e., the greater $b(i)$ is), the larger the resulting $s(i)$. Thus, larger silhouette scores represent a better 'fit' between a clustering and the natural structure of the data.

For all of the numbers of clusters k tried, the clustering that yields the highest silhouette score is then chosen as the best set of clusters.

5.3.3 Characterizing Clusters

While we would expect members of a coalition to be similar in benefit space, similarity does not necessarily imply that a set of agents is a coalition. Considering a marketplace scenario, for example, buyers who favor a particular set of sellers may be close in benefit space, but may not be colluding—instead, they may simply share similar tastes, or have found the same set of reliable sellers. Thus, we consider the clusters found to be *candidate coalitions*; in our second step, candidates are characterized to detect coalitions.

We might expect a true coalition T to be more 'self-serving' (i.e., benefiting each other more than outsiders) than a 'normal' group G (i.e., a set of agents which is *not* a coalition). In this case, we would expect the benefit flowing *from* members of T *to* members of T to be greater than the benefit flowing from members of G to members of G. Conversely, we might expect a coalition to damage outsiders more than a 'normal' group would. Here, given a population P (and recalling that harm can be represented

as negative benefit), we would expect the benefit flowing from T into $P \setminus T$ to be less than the benefit flowing from G into $P \setminus G$. The first of these behaviors describes helping allies, while the second describes harming enemies.

Testing for self-serving behaviour

We apply these principles to characterize candidate coalitions. First, we test for 'teammate supporting' (or 'self-serving') behavior. Consider any given set of agents S, where $m = |S|$. In our representation, benefit is directed (i.e., $\beta_a(b)$ might not equal $\beta_b(a)$), so there are $m(m-1)$ 'relationships' between agents in S. We can compute the average benefit (per relationship) flowing *from* agents in S, *to* agents in S by:

$$\bar{\beta}_S = \frac{\sum_{i \in S} \sum_{j \in S, j \neq i} \beta_j(i)}{m(m-1)} \tag{5.4}$$

We use Formula 5.4 to find $\bar{\beta}_C$, the average benefit within C.

To know whether the computed value is abnormally high, we need a benchmark to which to compare it. For this, we take random samples of m agents (drawn from the entire population P, including agents in C). For each sample G, we compute $\bar{\beta}_G$, by again using Formula 5.4. Performing this computation for a large number of samples (here, we use 100), we estimate the mean and standard deviation over $\bar{\beta}_G$.

With information about the distribution of $\bar{\beta}_G$, we can estimate the probability of obtaining a measure as high as $\bar{\beta}_C$ by chance, using the normal distribution. If this probability is too low, we conclude that members of C are benefitting each other far more a typical group of agents, and that C thus contains a coalition.[5] The threshold probability below which clusters are considered to contain coalitions (α) is a parameter: lower values reduce the risk of false positives, while increasing the risk of false negatives. In our tests, we used $\alpha = 0.001$.

[5]We only apply this technique to clusters no larger than half the size of the population. Clusters larger than this are ignored. There are two key reasons for this. First, we identify coalitions essentially by the fact that their behaviour deviates from the 'norm', i.e., non-colluding behavior. When coalition size exceeds half of the population, then non-colluding behaviour is no longer the norm; in this case, the tests can generate misleading results. Second, coalitions consisting of the majority of the members in a population are likely to be poorly-kept secrets, and need no special detection methods.

Testing for enemy-harming behaviour

Second, we test for 'enemy harming' behavior. Again, for any set of agents S (of size m), we can compute the average net benefit flowing from agents in S to outsiders:

$$\bar{\beta}_{\bar{S}} = \frac{\sum_{i \in S} \sum_{j \in P \setminus S} \beta_j(i)}{m|P \setminus S|} \tag{5.5}$$

We use Formula 5.5 to compute $\bar{\beta}_{\bar{C}}$. Then, as above, we repeatedly draw random selections G of m agents, and use Formula 5.5 to compute $\bar{\beta}_{\bar{G}}$. We compute the mean and standard deviation of $\bar{\beta}_{\bar{G}}$ over the samples, and use these statistics to determine the probability of encountering a value as low as $\bar{\beta}_{\bar{C}}$ (i.e., abnormally low net benefit), in the same manner described above. If the probability is less than α, we consider the cluster to contain a coalition.

Classification

If a cluster is recognized by either or both of these tests to contain a coalition, we label all agents in that cluster as coalition members.

We summarize the algorithm fully in Algorithm 5.1, for clarity, and for future reference.[6] (Informally, we henceforth refer to this algorithm simply as a 'coalition detection algorithm').

5.4 Experimental Results

5.4.1 Method

Ideally, we would have real-world data with which to evaluate our technique. Unfortunately, real-world colluders do not willingly reveal themselves as such, making it problematic to obtain labelled data that might be used for validation. Thus, simulation data was used to validate our technique for the electronic marketplace scenario; data was generated using the TREET testbed (described in Chapter 3).

[6] Although excluded from Algorithm 5.1 for clarity, note that the distributions generated in step 3a can be cached for the input data set, and re-used for any cluster C of the same size.

Algorithm 5.1 Coalition Detection and Identification

1. Map agents to points in benefit space.
2. **Clustering:** Apply k-means clustering, to identify groups of agents that are similar in terms of whom they benefit; use silhouette to choose the optimal number of clusters.
3. **Characterization:** For each cluster C:

   Characterize C as a coalition if its benefit to members is unusually high, or its benefit to outsiders is unusually low, as follows:

   (a) Repeatedly select random sets of agents G (of size $|\mathrm{C}|$) from the population, and for each G, calculate $\bar{\beta}_G$ (the average benefit from G to its own members) and $\bar{\beta}_{\bar{G}}$ (the average benefit from G to outsiders).

   (b) If $\bar{\beta}_C$ is unusually high (given the distribution from the random samples G), or if $\bar{\beta}_{\bar{C}}$ is unusually low (given the distribution of from the random samples G):

   - Classify C as a coalition, and all members as coalition members.

   Otherwise:

   - Classify C as a non-coalition, and all members as non-coalition-members.

In our simulations, the marketplace was populated with agents who make use of the Beta Reputation System (BRS) [29] in order to find trustworthy sellers and avoid unreliable ones. BRS was selected for several reasons:

- BRS is well known and understood, and frequently cited by researchers in the field.
- BRS represents an 'arms length' choice; although it is difficult to see how the choice of TRS might influence these results, selecting a system of our own design might cast doubt on the findings.
- Most importantly, the choice of reputation system is largely immaterial for this study. Our concern is not whether the TRS is effective in preventing the collusive attacks. (As shown in Chapter 4, all of the systems studied are ineffective in this regard.) Rather, we are considering acts of benefit/harm between agents. The principles used for detection are independent of the TRS used.

Except where noted, the simulations were configured as follows. Agents acted as both buyers and sellers. The majority of the population consisted of 'honest' agents who fulfill every sale they make honestly, and provided accurate reviews of other agents. Into this larger honest population, we inserted coalitions of various sizes making use of either bad-mouthing or ballot-stuffing to improve their competitive position. (The coalition agents were otherwise honest, fulfilling all sales diligently.) The total population size in each run was 1000 agents. In any given trial, from 0 to 4 coalitions are present. A variety of coalition sizes were tested, from 25 to 200 members, as reflected in the figures. For each combination of parameter values, 10 trials were run; the figures reported reflect the aggregate results across trials.

In our environment, a number of measures of benefit and harm can be identified, e.g., number of purchases, dollar value of purchase, number of positive reviews, average review score, etc. Here, we use only one of the available measures: the net sum of the review values given (counting a positive review as $+1$ and a negative review as -1), weighted by the dollar value of the transaction. This captures both benefit (positive reviews) and harm (negative reviews), as well as treating higher-value transactions as more important than lower-value ones.

The simulator provides us with labelled data: each agent is known to be either part of a coalition, or not. We remove these class labels before applying our technique. Afterward, we compute the accuracy of our technique by comparing our output to the actual classes of the agents.

5.4.2 Results

Single coalition

In the first set of tests, we evaluate the technique where exactly one coalition is present in the population. (Our technique does not make use of this fact, however—the same approach is applied, regardless of the number of coalitions actually present.)

First, we consider coalition members engaged in bad-mouthing. Agents did not make additional purchases in order to bad-mouth competitors; instead, coalition members bought the products that they actually needed, but if they purchased from a competitor, they gave a negative review. (This constitutes a conservative form of bad-mouthing, where no additional money is 'wasted' on fake purchases of unneeded products.) When the same product was available from both coalition members and non-members, members had no preference to buy from one or the other—in these tests we sought to detect bad-mouthing itself, and did not wish to introduce the possibility that agents were detected based on preferential buying from teammates. These results are shown in Figure 5.5, which contains three series. The first, 'Average overall accuracy', shows the percentage (across all trials) of agents that were accurately classified as either coalition members or non-members. By this metric, performance was excellent.Unfortunately, this measure can be misleadingly high, especially when the number of colluders is low. For example, where 25 colluders are present in a population of 1000, simply labelling all agents as non-colluders yields an overall accuracy of $(1000-25)/1000 = 0.975$.

The second series, 'Average coalition accuracy', depicts the fraction of coalition members that were accurately labelled as such. (Note that this is equivalent to *recall*.) Note that while overall accuracy is misleadingly high, coalition accuracy clearly shows that some colluders were missed for the smallest coalition size. This reflects a pattern throughout the experiments, in fact: the algorithm performs well in general, but finds very small coalitions challenging to detect.

The third series, 'Average false positives' shows the number of non-coalition members that were mistakenly identified as coalition members.[7] This is an extremely important

[7] One might use the well-known measure, *precision*, to measure the proportion of true positives to false positives, and we could have done so here. Our primary concern, however, is the risk that non-coalition agents might be incorrectly classified as coalition members. Precision uses the total number of positive classifications as the denominator—it reflects the probability that an agent accused of being part of a coalition is actually a member (i.e., $P(M|C)$), where M denotes actually being a coalition member, and C denotes being classified as a member. $1-$ precision would thus reflect the probability that an accused agent is not a coalition member (i.e., $P(\sim M|C)$). In contrast, our 'average false positives' measure uses

measure—wrongly accusing someone of collusion might be worse than missing a real colluder. Note that there were no false positives at all, except for a small number with the smallest coalition size.

Overall, the results on bad-mouthing are very strong.

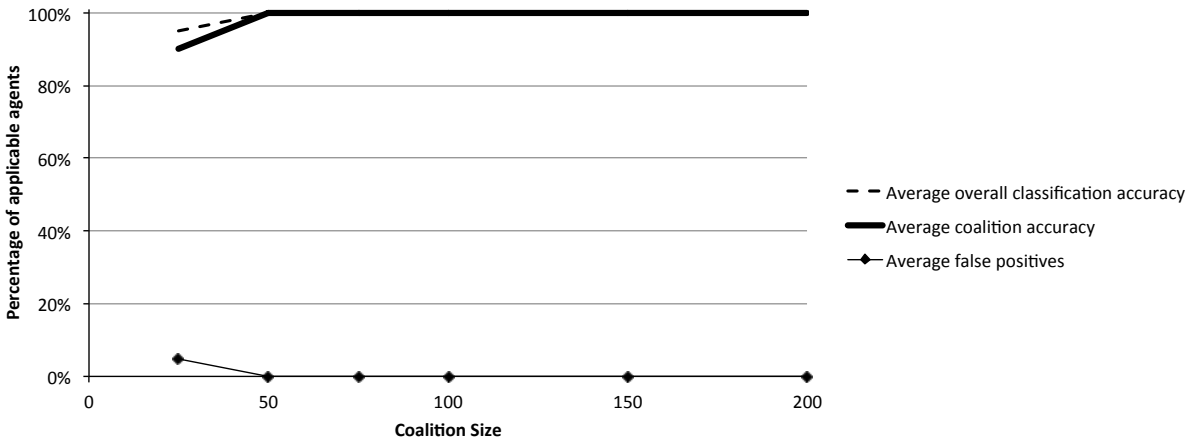

Figure 5.5: Bad-mouthing: Detection performance, with single coalition present.

One might wonder whether, because all sellers were acting honestly, if bad-mouthing agents stand out because of their negative reviews. Accordingly, we investigated the case where only ballot-stuffing is used. Here, the coalition members give accurate reviews of non-members. In addition to their normal purchases of needed products, coalition members engage in up to an extra 25% ballot-stuffing purchases from teammates. The results are depicted in Figure 5.6. Overall, we note excellent detection performance (nearly as strong as for bad-mouthing); in this case, there are no false positives at all.

Zero coalitions

While performance is strong with exactly one coalition, it may be the case that there is no coalition present in a given population. Such situations provide a good test of the

the total number of non-coalition agents as a denominator—it reflects the probability that a non-member be classified as a coalition member (i.e., $P(C| \sim M)$), which more accurately reflects our concern. This measure is also well defined when our algorithm (conservatively) refrains from classifying any agents as coalition members.

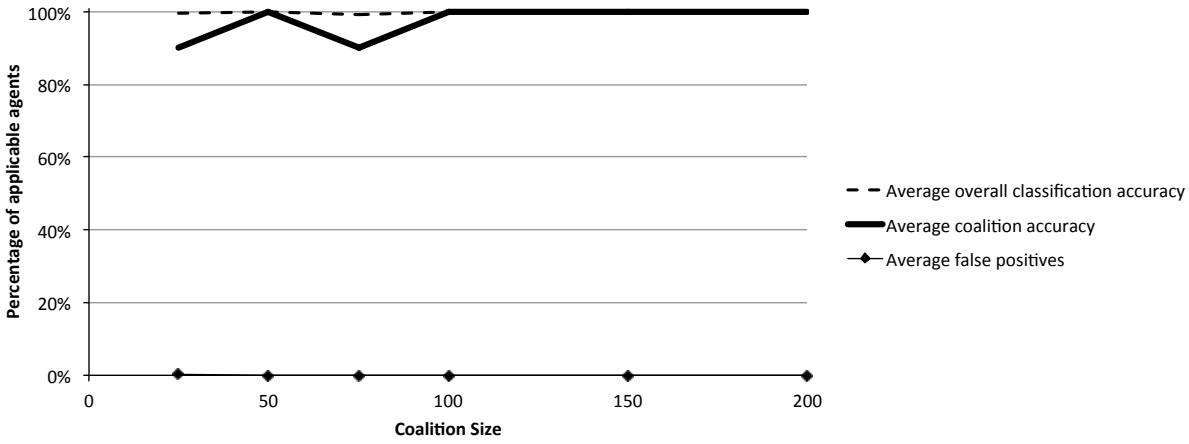

Figure 5.6: Ballot-stuffing: Detection performance, with single coalition present.

algorithm's resistance to false positives. The results are shown in Table 5.1; virtually no false accusations were made.

Table 5.1: Performance when no coalitions present.

Case	Trials	Pop.	False pos.	Rate
Bad-mouthing	60	1000	0	0
Ballot-stuffing	60	1000	12	0.0002

Multiple coalitions

Just as a population might contain no coalitions, it might also contain multiple coalitions. To validate the algorithm's ability to handle multiple groups, we ran trials with up to 4 coalitions.[8] The results for bad-mouthing are displayed in Figures 5.7 (average coalition detection accuracy) and 5.8 (average false positives); those for ballot-stuffing are shown in Figures 5.9 (coalition detection) and 5.10 (false positives).

[8]As explained in 'Characterizing Clusters', we do not apply our characterization technique where the number of coalition members exceeds half of the population. Thus, certain parameter combinations (e.g., 4 coalitions of size 200 each = 800) were not included in our experiments, and do not appear in the figures.

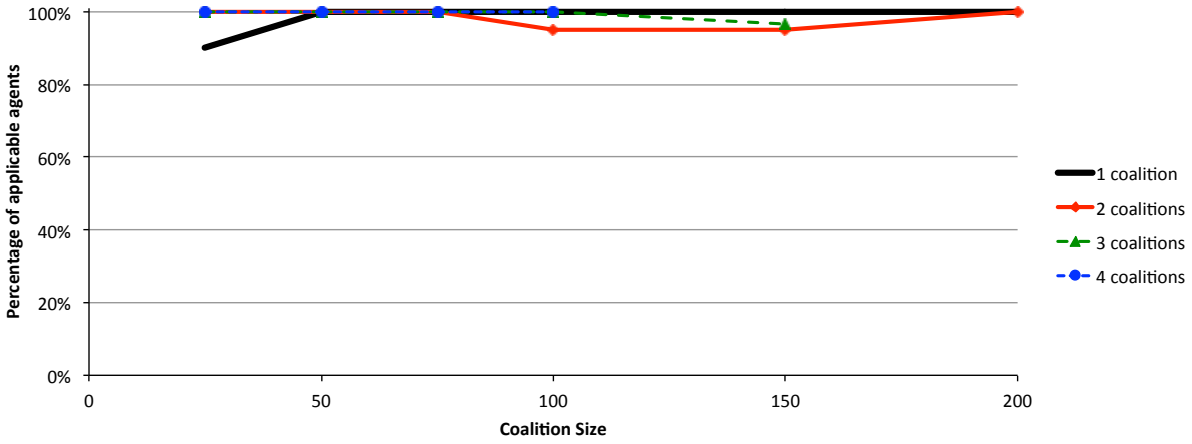

Figure 5.7: Bad-mouthing: Coalition detection accuracy, multiple coalitions.

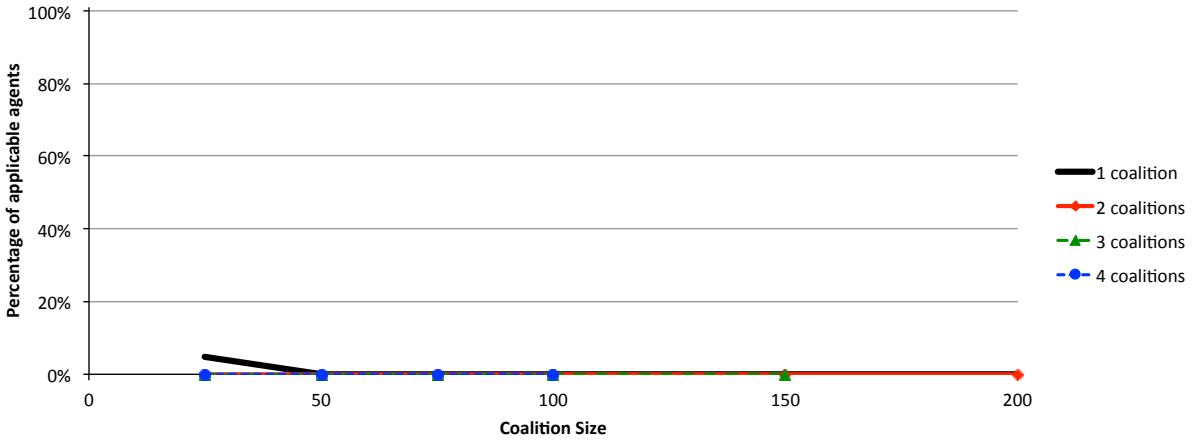

Figure 5.8: Bad-mouthing: False positives, multiple coalitions.

Several overall patterns can be noted in this set of charts.[9] First, overall performance is quite strong, in all cases. As revealed in the single-coalition cases, coalition detection performance is again better for bad-mouthing than for ballot-stuffing; similarly, the general pattern of weaker performance on smaller coalitions is again evident in the ballot-

[9]There is a high degree of intrinsic randomness in our scenario. For example, the random determination of which agents can produce which products has an impact on purchasing patterns. This randomness is evident in these and other charts throughout this document—there is fluctuation in series, despite the fact that each data point represents an average over 10 trials.

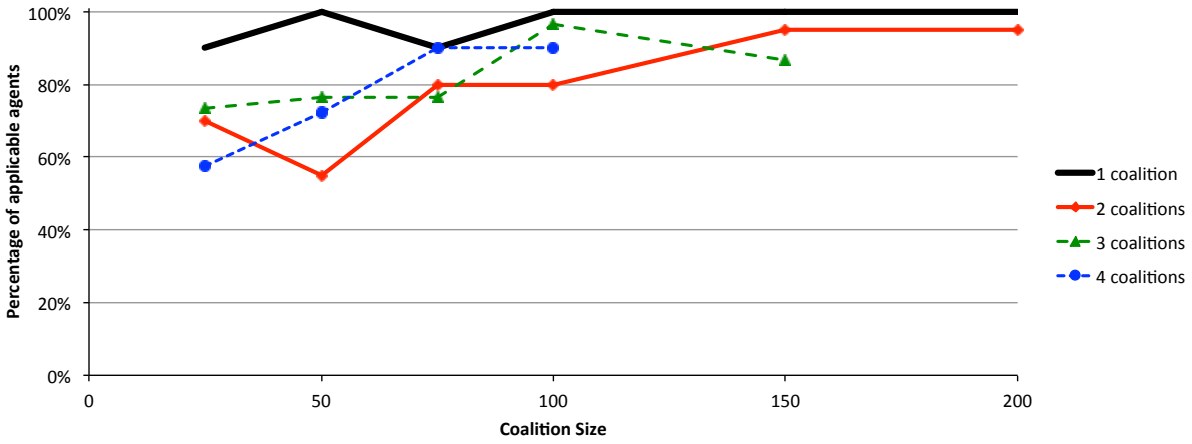

Figure 5.9: Ballot-stuffing: Coalition detection accuracy, multiple coalitions.

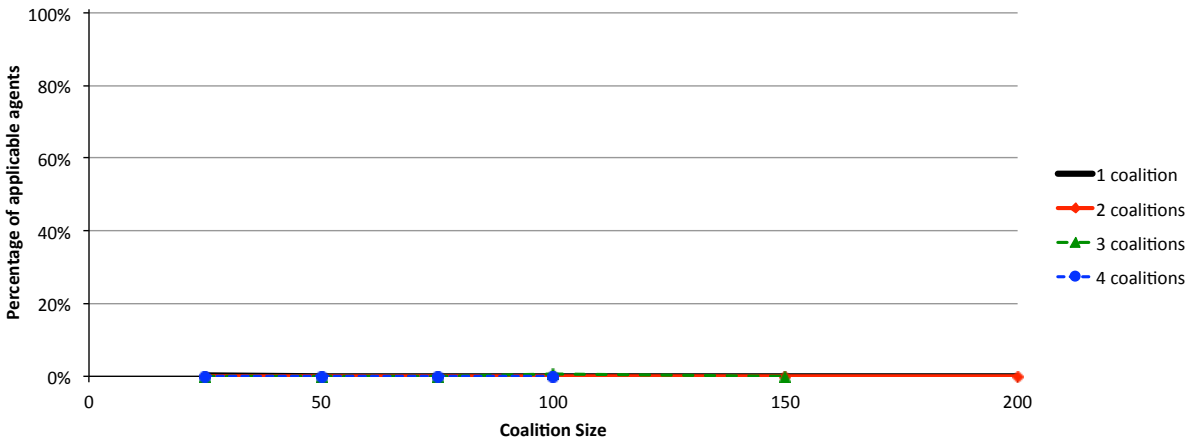

Figure 5.10: Ballot-stuffing: False positives, multiple coalitions.

stuffing data. There are virtually no false positives in the bad-mouthing results (with the exception of the single-coalition case already noted) or in the ballot-stuffing results.

Perhaps most importantly, note that there is no clear correlation between number of coalitions and performance: increasing the number of coalitions does not have the detrimental impact on performance that one might expect. (For ballot-stuffing, the multi-coalition cases have lower detection rates than the single coalition cases; note, however, that performance on the 2-coalition cases is worse than the 3- and 4-coalition cases.)

5.4.3 Exploring other key parameters

In a domain such as this, there are a multitude of different scenario parameters; while many are unlikely to play a role in detection accuracy, others may be potentially important. Here, we consider a number of key parameters, and explore their impact on the performance of our algorithm.

Sensitivity to cheating agents

In the previous section, we sought to focus on collusive activity itself, without introducing other factors that might obscure the issue. Thus, in previous tests all agents fulfilled sales honestly—when a good was sold, it was faithfully delivered. (This is distinct from collusive reviews, which involve dishonesty in reporting past experience, not in fulfilling sales.) One might wonder, however, if the absence of cheating may be a factor in our strong detection performance (particularly in the case of bad-mouthing, where reviewers falsely claim to have witnessed dishonesty). To examine this issue, we ran additional trials where every agent (coalition members and non-members) was assigned a probability with which they cheated on sales (i.e., agreed to sell a product, but then failed to deliver the promised good).

In the first set of tests, every agent had the same probability of cheating on any individual sale; trials were run for a variety of different probabilities. The results for bad-mouthing are depicted in Figure 5.11, and those for ballot-stuffing in Figure 5.12. (As can be seen in the graph, we stopped short of having agents cheat with 100% probability. A marketplace where every agent cheats on every sale is non-functional, and any results would be suspect.) Two coalitions were present in each trial, with coalitions sizes of 50 and 100 (shown as separate series) used for comparison.

Several features should be noted. First, overall performance is excellent; the performance on ballot-stuffing is slightly worse then for bad-mouthing, continuing a trend we have seen in earlier results. Second, there is little obvious correlation between cheating probability and performance. There appears to be a small deterioration in performance for bad-mouthing, as cheating rates approach their maximum level. Note, however, that such a deterioration is not obvious in the ballot-stuffing case. (Further, we note that accuracy against bad-mouthing is still strong, even at the highest cheating levels.)

It is unlikely, however, that every agent will cheat with the same probability. It is more interesting to consider the realistic case where different agents cheat with different

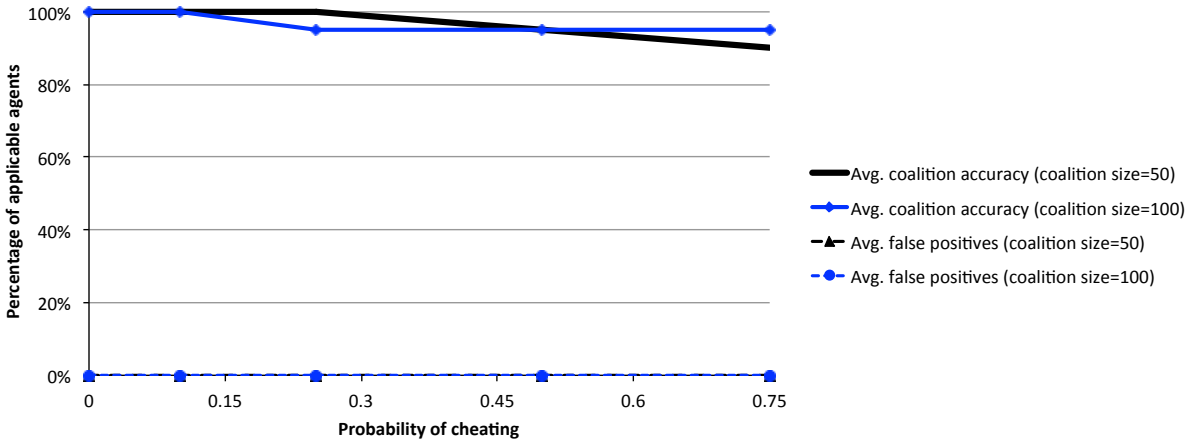

Figure 5.11: Bad-mouthing: Performance with uniform probability of cheating.

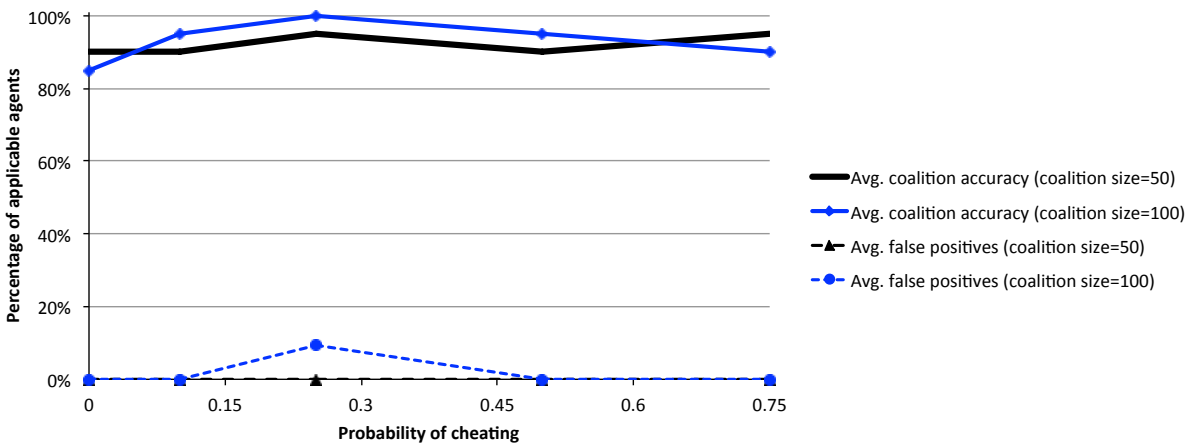

Figure 5.12: Ballot-stuffing: Performance with uniform probability of cheating.

probabilities. In particular, because our method is based on similarity, one might question whether coalitions can be effectively identified if agents are very *dissimilar* in their cheating behaviour. Thus, we ran further tests, where each agent's cheating probability was drawn from a Gaussian distribution. The mean cheating probability was 0.5, while a range of different standard deviations were used. If a value greater than 1 was drawn, it was treated as 1; similarly, negative values were treated as 0. Standard deviations of up to 0.3 were used—at 0.3, the entire range [0,1] is well covered. The results for bad-mouthing are shown in Figure 5.13, and those for ballot-stuffing in Figure 5.14.

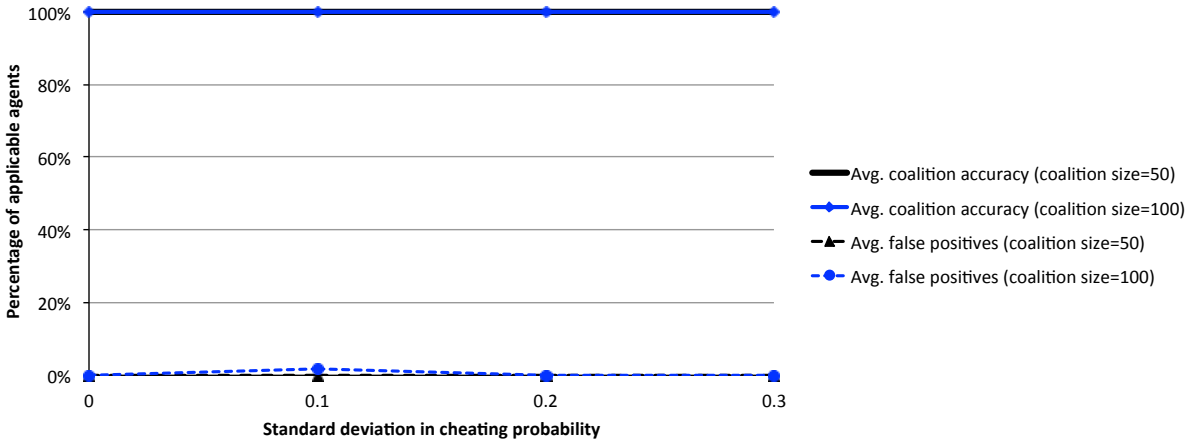

Figure 5.13: Bad-mouthing: Performance with random probability of cheating.

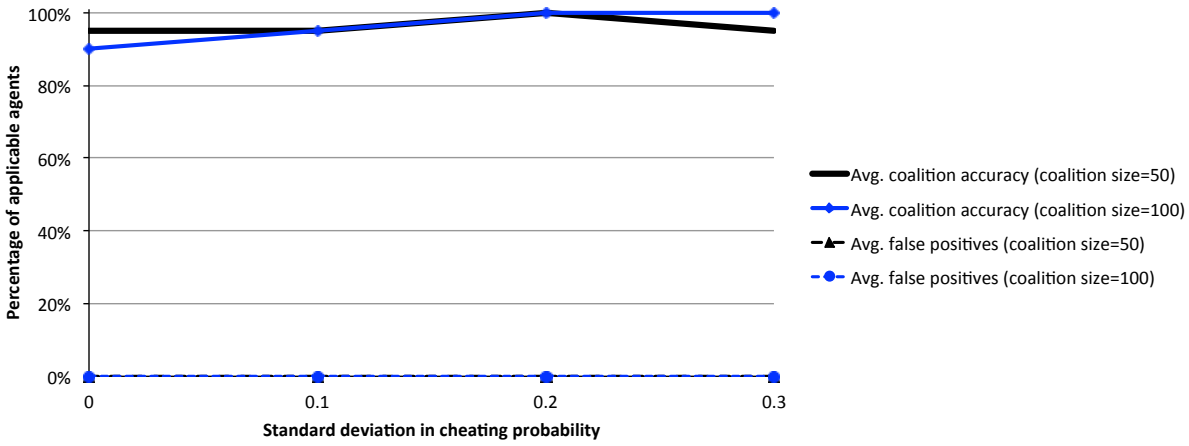

Figure 5.14: Ballot-stuffing: Performance with random probability of cheating.

Detection accuracy is excellent, in both cases, and no correlation between variability in cheating and performance is evident.

Sensitivity to the quantity of cooperative behaviour

While cheating appears to have little impact on performance, there is another issue which is likely to be of greater importance: the degree or quantity of cooperative behaviour. Our

method identifies coalitions using the similarity of beneficial activity; it is reasonable to suspect that its accuracy may be affected by the quantity of such activity.

We ran a series of tests with different rates of collusive behaviour; in this first set, the probability/proportion[10] of engaging in a collusive transaction was the same for every coalition member. For bad-mouthing, the probabilities tested covered the entire probability range [0,1]. For ballot-stuffing, agents engaged in up to an additional 30% ballot-stuffing transactions, above their legitimate needs. (We had obtained strong results in our earlier tests, at a rate of 25%.) The results for bad-mouthing are depicted in Figure 5.15, and those for ballot-stuffing in Figure 5.16.

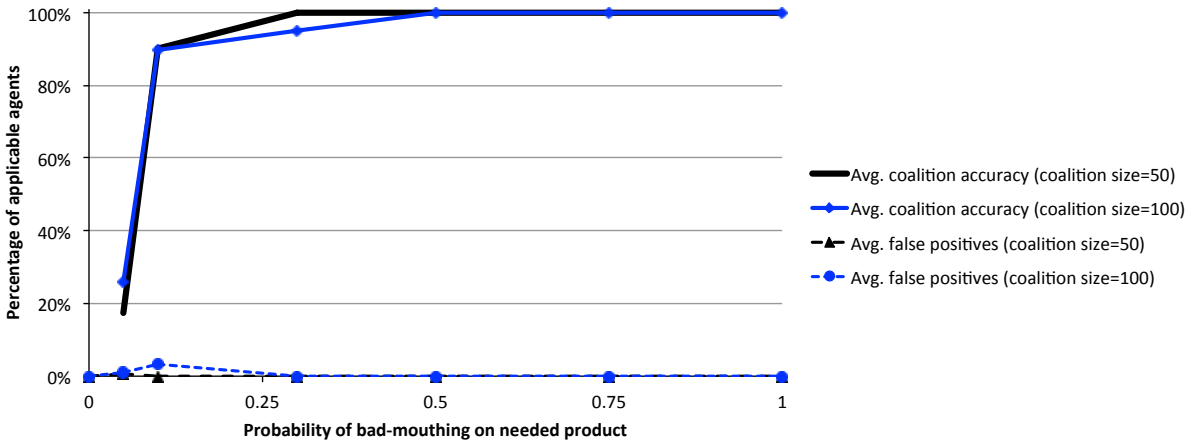

Figure 5.15: Bad-mouthing: Performance with uniform probability of collusive behaviour.

As we have seen previously, performance is generally better for bad-mouthing than for ballot-stuffing. More importantly, though, is that we see what we would intuitively expect: as the quantity of collusive behaviour decreases, the detection accuracy decreases. (Despite this, note that false-positives remain extremely low, an important result.) Performance does not decrease linearly, however. Instead, there appears to be something of a threshold relationship, which is particularly evident in the bad-mouthing data.

While it is expected that low levels of collusive behaviour will be difficult to detect, improving performance here is a goal of refinements presented in Chapter 6.

[10]For convenience, from this point we simply refer to 'rate', rather than 'probability/proportion'.

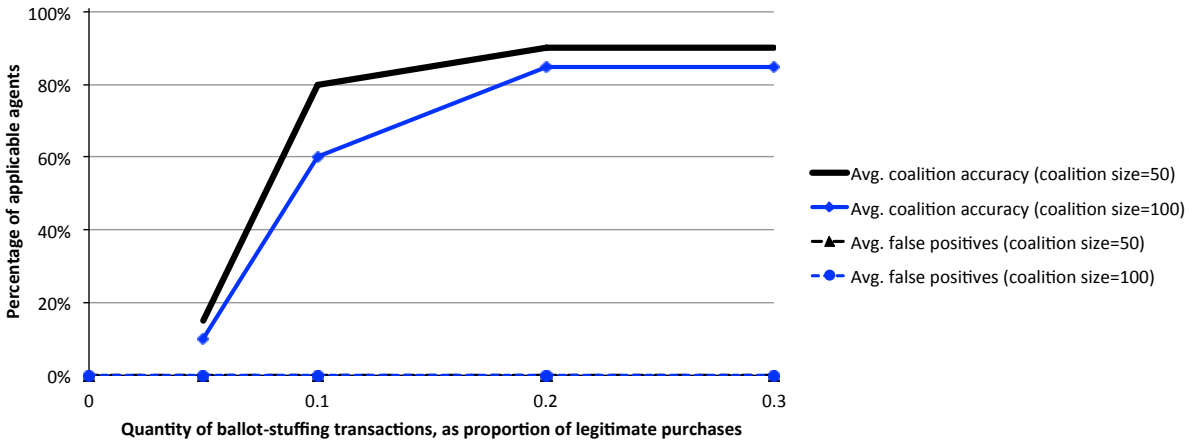

Figure 5.16: Ballot-stuffing: Performance with uniform proportion of collusive behaviour.

Sensitivity to randomness in the quantity of cooperative behaviour

Again, as with cheating, it is unlikely that all agents will collude at exactly the same rate. Thus, we conducted further tests, where the rate of collusive activity for each agent was randomly drawn from a Gaussian, as was done earlier for cheating rates. Trials were run with a range of different standard deviations. For the bad-mouthing case, a mean of 0.5 was used; results are depicted in Figure 5.17. For ballot-stuffing, a mean of 0.25 was used, with results shown in Figure 5.18.

As one might expect, performance degrades as the amount of variation in collusive activity increases. The obvious conclusion is that agents with different rates of collusive activity are less 'similar', and thus less likely to be detected by our method. This conclusion is premature, however. When the rate for each agent is drawn from a Gaussian, some agents will get a higher rate, and some a lower rate. Is it the *variability* that foils detection, or is it simply that those agents with lower rates are less likely to be detected?[11]

To explore this issue, we analyzed the results of the previous trials in a different way. Figures 5.19 and 5.20 depict the same results as Figures 5.17 and 5.18, except this time,

[11] In fact, in many of our test cases, agents will receive a collusion rate of 0. Consider the case where the mean of the Gaussian is 0.5, and the standard deviation is 0.3. We would expect approximately 4.8% of the agents to receive values of 0 or less (which are treated as zeros). Because these agents are, in fact, coalition members, failures to detect them count as a false negatives. There is an obvious argument, however, that an agent that engages in no collusive activity need not be labelled as a coalition member.

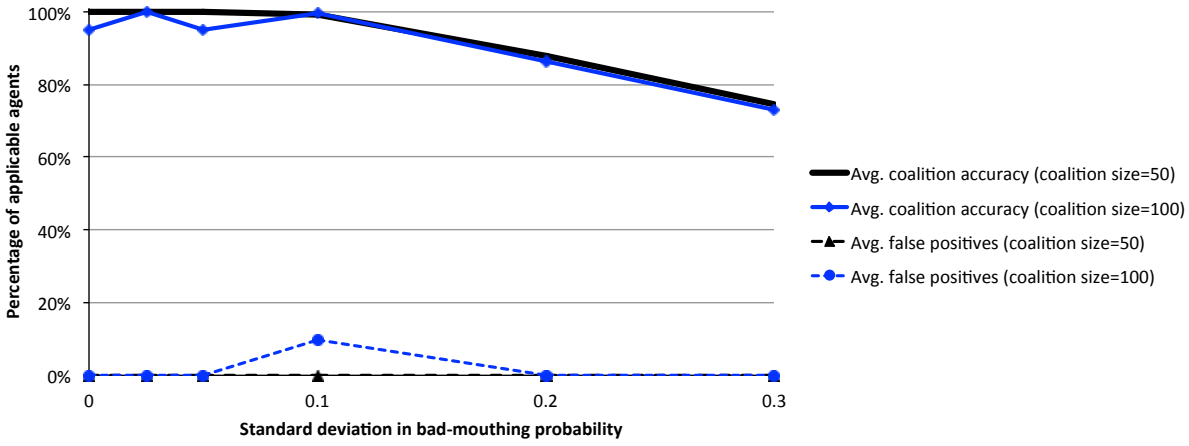

Figure 5.17: Bad-mouthing: Performance with random probability of collusive behaviour.

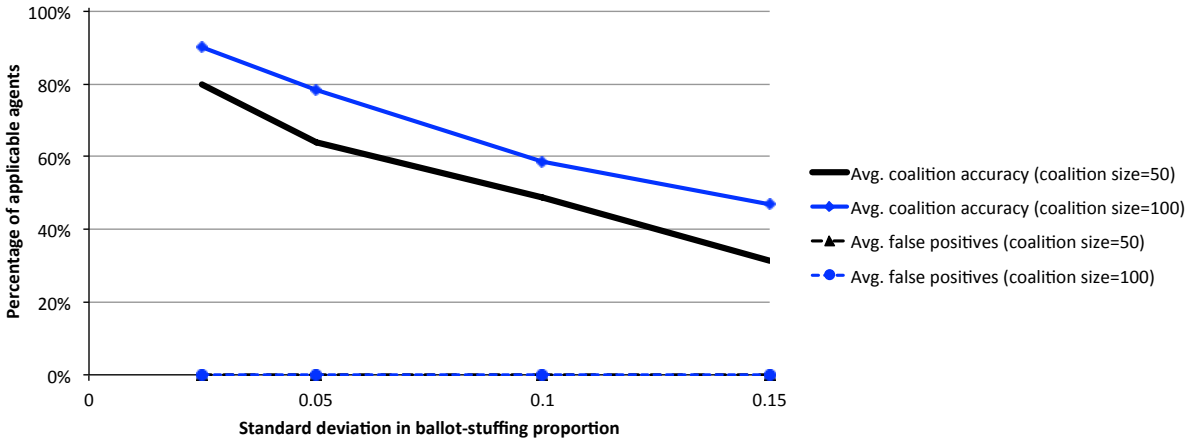

Figure 5.18: Ballot-stuffing: Performance with random proportion of collusive behaviour.

agents are mapped to the x-axis not by the standard deviation *parameter for the trial*, but instead by their *individual randomly-assigned collusion rate*. Note that different agents from the same trial will be mapped to different x-axis coordinates. (For display, collusion rates were discretized, with bins of 0.05 in width.)

It is interesting to note that Figures 5.19 and 5.20 reflect the same threshold character as Figures 5.17 and 5.18. This indicates that an agent's individual collusion rate

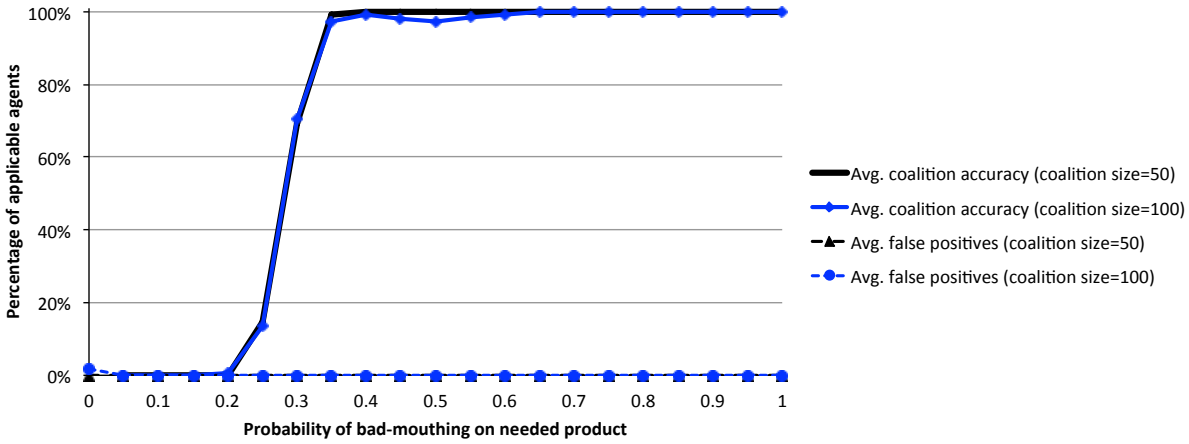

Figure 5.19: Bad-mouthing: Performance with random collusive probability, by agent's individual probability.

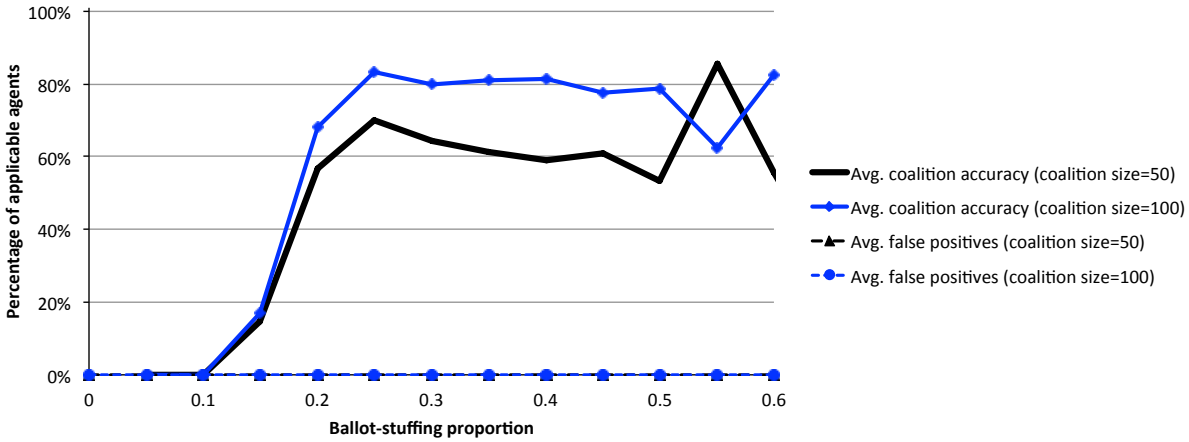

Figure 5.20: Ballot-stuffing: Performance with random collusive proportion, by agent's individual probability.

is key to determining whether it is detected. Agents that fall below the threshold are unlikely to be detected, and agents above the threshold likely to be detected, regardless of the standard deviation parameter for the trial, and despite the collusion rates of other coalition members.

This result is perhaps surprising. It is also informative. It suggests that, to improve

detection accuracy, sensitivity to low levels of collusive behaviour is more important than robustness in the face of variability.

One must note, however, that variability *does* play a role. Detection accuracy in the cases with variability is, in fact, worse than in the cases without variability. Comparing Figure 5.20 with Figure 5.16, one can see that detection levels are lower.[12] Comparing Figure 5.19 with Figure 5.15, one can see that the threshold has been pushed to the right—when variability is present, higher collusive rates are required for detection.

As noted above, improving performance when levels of collusion are low (and variable) is a goal of enhancements presented in Chapter 6.

Sensitivity to buying rate

Because our method depends on similarity, we have explored two key dimensions in which agents might be dissimilar—frequency of cheating, and collusion rate. There is another important such dimension: the overall frequency with which agents make purchases. The number of products needed to purchase is randomly determined each turn, for each agent. The expected value of the distribution used, however, has been the same for every agent: each agent averages 5 products per turn. If agents buy at different rates, this may hinder detection. Consider a case, for example, where each agent engages in an additional 25% ballot-stuffing purchases, beyond legitimate needs. If agent A makes 4 legitimate purchases per turn on average, and agent B makes 8, then agent B will also engage in twice as many ballot-stuffing transactions as A (2 vs. 1), despite A and B having the same 'collusion rate'.

We explored this issue by running a number of tests where agents have varying (legitimate) buying rates. The buying rate for each agent was drawn from a Gaussian distribution with a mean of 5. Trials were run with a range of different standard deviations. The results for bad-mouthing appear in Figure 5.21, and those for ballot-stuffing in Figure 5.22.

Interestingly, the bad-mouthing and ballot-stuffing cases differ dramatically. Detection of bad-mouthing appears little-affected by variability in buying rate. In contrast, for ballot-stuffing coalition detection drops significantly and false positives spike on trials with high variability.

[12] The results in Figure 5.20 also appear more erratic than those in Figure 5.16. This does not imply that detection accuracy is erratic, however. Recall that the collusion rates are drawn from a Gaussian, centred on 0.25. This means that there are relatively few agents present at the higher collusion rates, resulting in the jaggedness in the right-most reaches of the series.

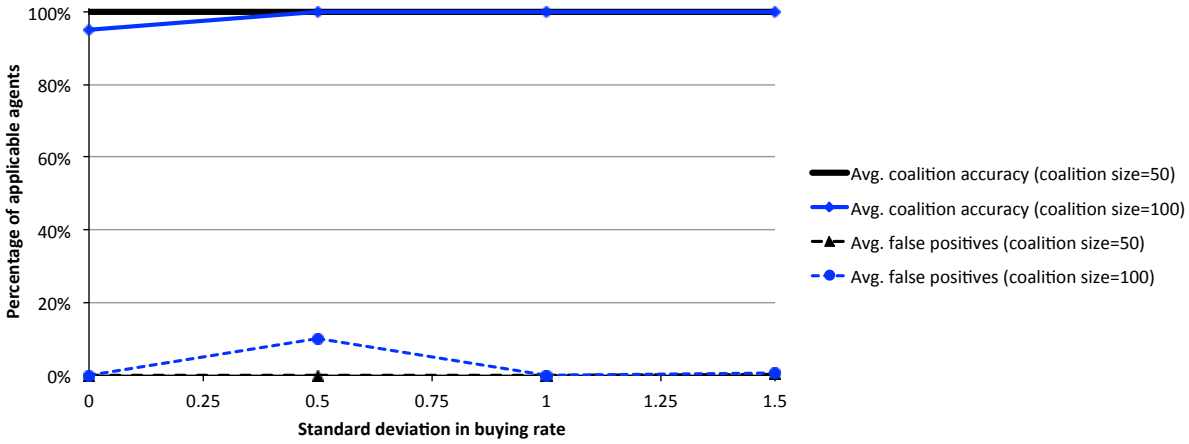

Figure 5.21: Bad-mouthing: Performance with variable buying rates.

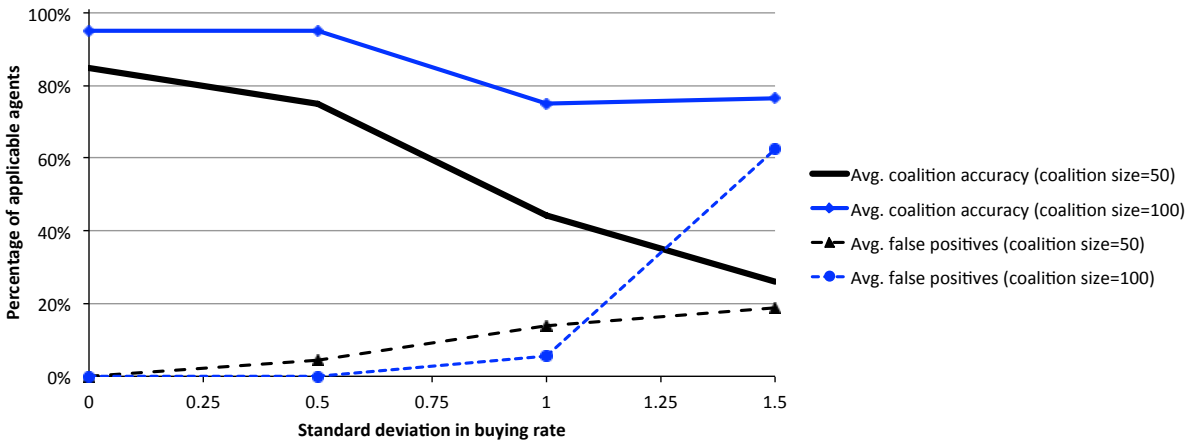

Figure 5.22: Ballot-stuffing: Performance with variable buying rates.

Given the real-world likelihood that buying rates will differ between traders, this is an important issue. Fortunately, there is a reasonably simple solution.

Normalization of activity In the analysis of an agent's activity, we are interested in whether it favours particular agents more than others; the overall level of activity is of much less concern. As such, to improve the quality of data before analysis, we can normalize it to minimize the impact of varying levels of overall activity (here, purchasing

rates). Until this point, our analysis has been based on absolute quantities of benefit: for any given agent a, the have used the (raw) amount of benefit it bestows upon agent i, $\beta_i(a)$. Now, we divide this raw quantity of benefit by the total benefit agent a generates:

$$\frac{\beta_i(a)}{\sum_{j \in P} \beta_j(a)} \tag{5.6}$$

where, again, P is the entire population of agents. Thus, the benefit values used in analysis represent the fraction of each agent's overall activity—each agent's characterization is normalized to the same level of activity.

Returning to the case depicted in Figure 5.22 (where ballot stuffing agents vary in buying rates), we employ normalization in our detection algorithm. The results are depicted in Figure 5.23.

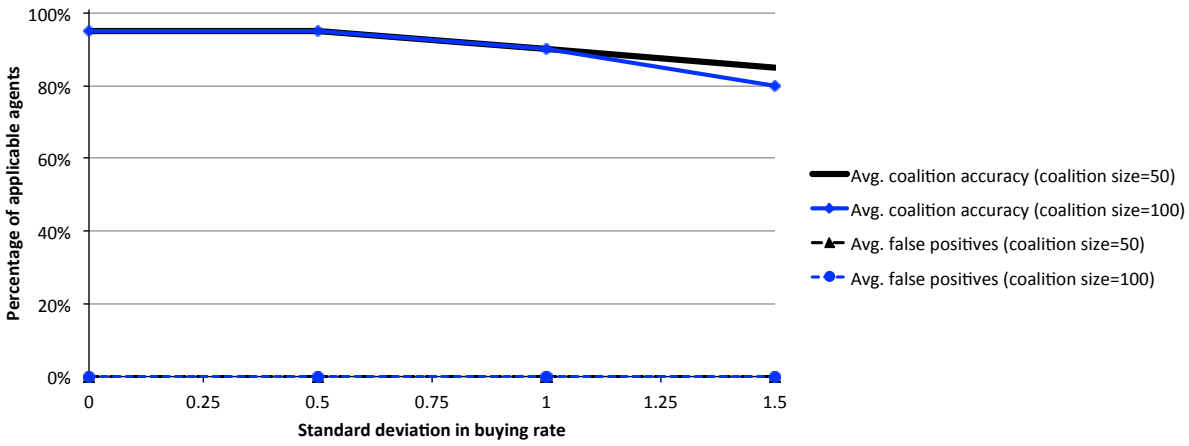

Figure 5.23: Ballot-stuffing: Performance with variable buying rates, using normalization.

Compared to performance without normalization, detection accuracy has dramatically improved; false positives have been eliminated entirely.

Normalization is simple, yet powerful. Moreover, we encountered no cases during our experimentation where it had detrimental effects. Thus, normalization is used throughout the remainder of this document (except where specifically noted.)

Scalability

One final question relates to scalability: how will performance change as the population size changes? To explore this issue, we ran trials with population sizes ranging from 500 to 5000, with a coalition of size 50.

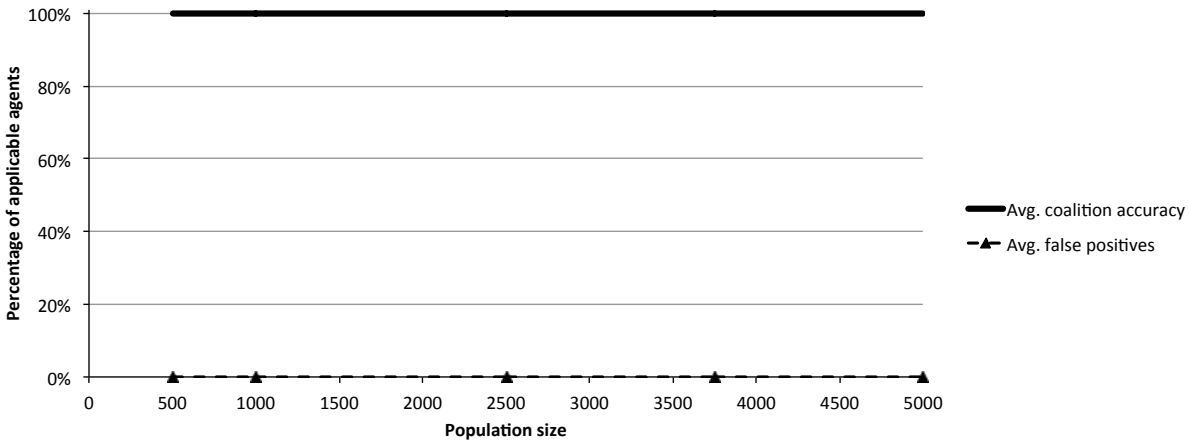

Figure 5.24: Bad-mouthing: Performance as population size changes.

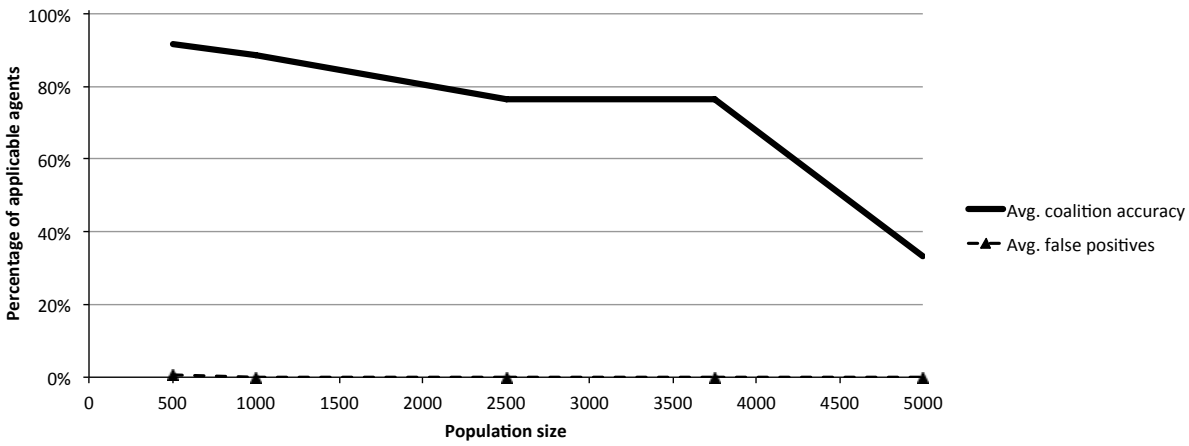

Figure 5.25: Ballot-stuffing: Performance as population size changes.

Results for bad-mouthing, shown in Figure 5.24, are excellent, and performance is unaffected even as population ranges over an order of magnitude.

While performance for ballot-stuffing has typically been slightly weaker than for bad-mouthing, Figure 5.25 may still be surprising. Ballot-stuffing detection accuracy deteriorates significantly as the overall population size increases. An examination of this problem, and refinements to our algorithm which improve on this performance, are presented in Chapter 6.

5.4.4 Pathological cases

Our algorithm detects coalitions by the similarity of their behaviour (and by the difference between coalition activity and 'normal' activity). It is possible to conceive of certain 'pathological' cases however, where coalition members are able to benefit one another, but not look 'similar' in doing so. Such cases may be of interest because coalitions seeking to avoid detection may attempt to make use of them.

Here, we identify two potential such cases, and examine the performance of our algorithm when faced with them.

Ring structure

A well-coordinated coalition might seek to develop a ring structure similar to that depicted in Figure 5.26. Agents A and B ballot-stuff only for C and D, C and D ballot-stuff only for E and F, and so on. In this way, every agent benefits members of the coalition, and every agent receives benefit from members. However, A and B attempt to avoid looking similar to other agents in the coalition, because they are the only two who are ballot-stuffing C and D. (We will refer the a set of agents who are ballot-stuffing for the same targets (e.g., A and B) as a *link*; the number of agents in each link is the *width* of the ring, while the number of links is the *circumference*).

Will this structure avoid detection? It is worth noting that, while the agents in one link (e.g., A and B) and not ballot-stuffing for the same agents as other links, they are very similar with *each other* in terms of who they are benefiting. (The coalition might try to minimize this by severely limiting the width of the ring. However, this has the simultaneous effect of limiting the amount of benefit each agent can receive.) Moreover, while agents in different links are not ballot-stuffing the same target agents, their behaviour may still be very dissimilar from 'normal' agents outside the coalition.

To explore this, we ran tests pitting the detection algorithm against coalitions using a ring structure. One coalition was present in each trial, with coalition sizes of 50 and

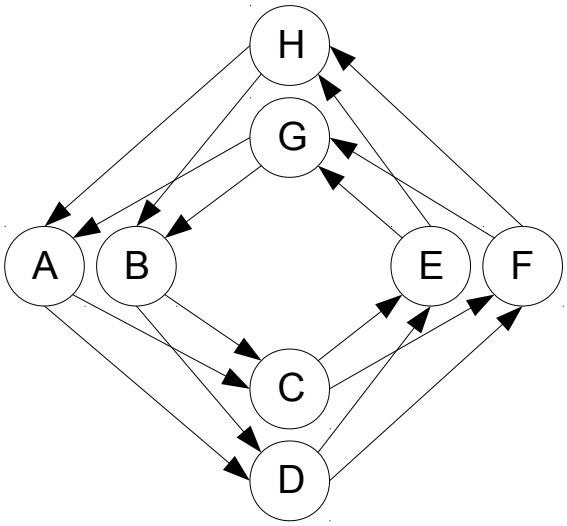

Figure 5.26: A ring structure for ballot-stuffing.

100 used. A range of different circumferences were used across trials. (Thus, the width of the rings varied as well, where width $\approx |C|/$ circumference.) The results are shown in Figure 5.27, where the x-axis is the ring circumference used.

Note that the range of tests included very wide (100 agents / 5 circumference = 20 width) and very narrow (50 agents / 20 circumference = 2.5 width) rings. Detection accuracy is very strong throughout, and no false positives are present.

Singletons

Another attempt to potentially avoid detection might be the use of what we refer to as *singleton* accounts: 'fake' accounts that are used only for a small number of bad-mouthing or ballot-stuffing transactions (here, one), and then abandoned. The singleton accounts do no other buying or selling. This behaviour is easily executed in marketplaces where identities can't be established, and new accounts can be created freely—conditions that exist in many real-world systems.

We conducted tests where coalitions consisted both of a number of 'real' members (50% of each coalition), and a number of singleton accounts (50%). Each singleton

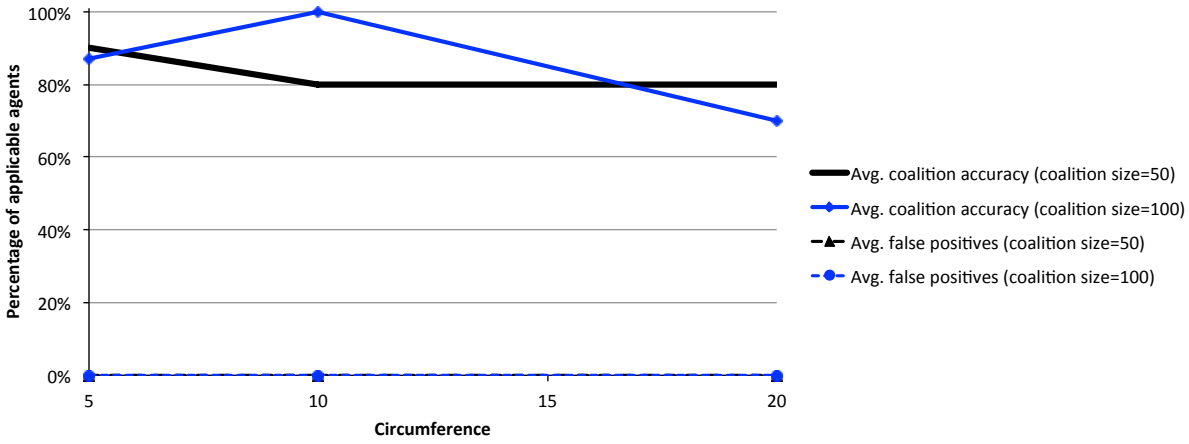

Figure 5.27: Ballot-stuffing: Performance against ring structure.

account was used for one collusive transaction, which occurred on a randomly selected round between the 10th and the 300th. 'Real' coalition members behaved normally, bad-mouthing or ballot-stuffing as in previous tests. The results are shown in Figures 5.28 (bad-mouthing) and 5.29 (ballot-stuffing).

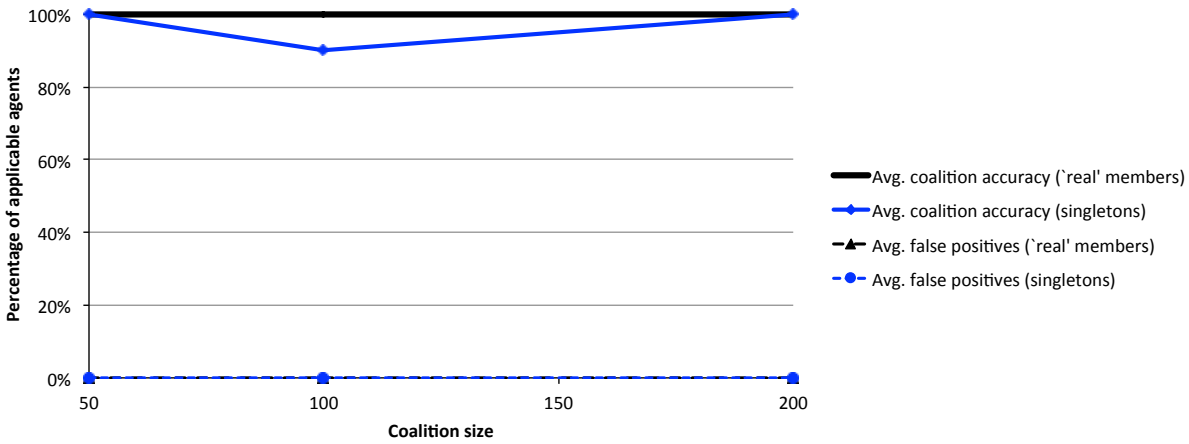

Figure 5.28: Bad-mouting: Performance against singletons.

In each figure, the heavy black line shows the detection accuracy for the 'real' members. Performance here is strong, and very similar to that for a 'normal' coalition without

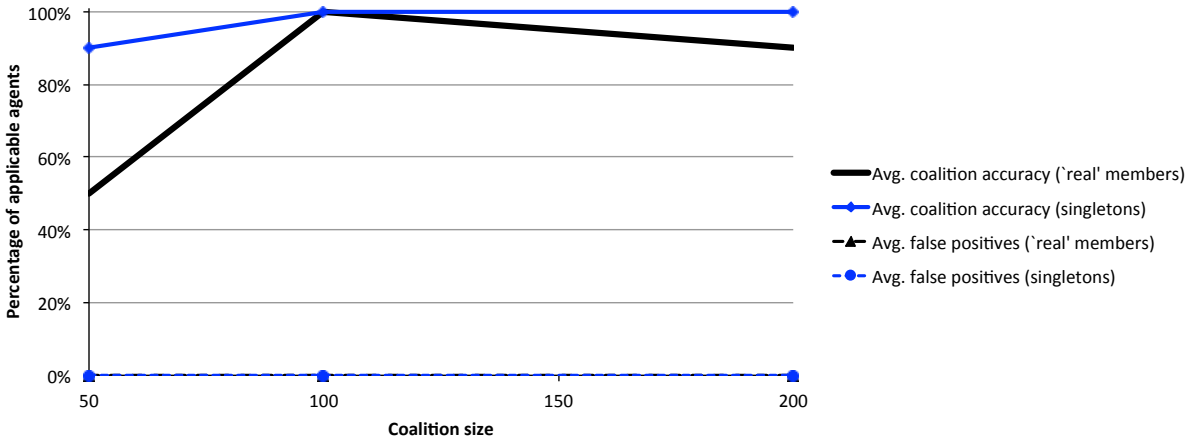

Figure 5.29: Ballot-stuffing: Performance against singletons.

singletons—compare to Figures 5.7 and 5.9.[13]

The second series in each figure shows the detection accuracy for the singleton accounts. Performance is very strong. False positives are absent, in every test.

As noted, these pathological cases might be seen as opportunities for coalitions to avoid detection; our results suggest that such tactics will have limited success.

5.5 Discussion

The performance of our algorithm is impressive, particularly considering that novelty of this problem. As revealed in the previous section, however, there are several circumstances that make coalition detection more challenging:

- Low levels of collusive activity (both due to low collusion rates, and in the case of ballot-stuffing, due to small coalitions);
- Significant variability in the rate of collusive activity;

[13]Note that here, 'real' members, constitute only 50% of each coalition; a coalition of size 100 here has the same number of 'real' members as a coalition of size 50 in Figures 5.7 and 5.9. Here, the only case where the algorithm struggles is for ballot-stuffing coalitions of size 50; this is equivalent to a coalition of size 25 in Figure 5.9, which revealed that ballot-stuffing detection is limited for such small coalitions.

- Increasing size of the overall population (in the case of ballot-stuffing).

In the next chapter, we develop enhancements and refinements to our method, with the goal of improving performance when faced with such circumstances.

It is worth noting that our algorithm detects groups of agents that are providing more net benefit to each other than other sets of agents. We might call this a *de facto* coalition. It may arise because the group is an actual coalition, intentionally acting in concert. Such a situation might possibly arise, however, due to other circumstances: for example, a group of agents may have closely aligned needs and capabilities, and favor each other for this reason. We make no attempt to distinguish between these cases (i.e., to determine intent), and the importance of this issue is likely to be scenario-specific. We do note, however, the close correspondence between actual coalitions and detected coalitions in our results.

Conversely, an actual coalition might be ineffective—perhaps members act too little to benefit one another (e.g., low collusion rate), or they coordinate poorly. Such a group might not be detected, despite technically being a coalition. Is this a problem? The answer may depend on the scenario; in our marketplace setting, perhaps not. It may be that the lesser degree of activity, which makes these harder to detect, also limits the benefit derived.

Chapter 6

Refinements and Enhancements

The method described in Chapter 5 (Algorithm 5.1) represents a first step toward the detection and identification of coalitions within a larger population. While the algorithm exhibits strong detection performance and resistance to false positives, opportunities exist for improvement. In this chapter, we explore a number of potential enhancements, to improve the accuracy and robustness of the method, and to further broaden its applicability.

In previous chapters, we sought to be as thorough as practically possible in the presentation of experimental results. From this point onward, for brevity (and the benefit of our reader) we present only those results that are noteworthy or illuminating.

6.1 Recursive refinement of coalitions

In Chapter 5, we focused on one key issue: were coalition members (and non-members) successfully classified as such? Here, we turn to a second concern: *are members of the same coalition grouped together?* In some scenarios, it may be sufficient to simply know whether an agent is part of a coalition, without knowing that agent's partners. There are likely to be scenarios, however, where it is desirable to identify the memberships of separate teams. For example, in a game or battlefield scenario, identifying the membership of a team might aid in subsequent analysis of the tactics the team is employing.

In this section, we examine the accuracy of our core algorithm not only for identifying whether individual agents are members of coalitions (as explored in Chapter 5), but also

for grouping members according to the coalitions to which the belong. We introduce a technique to improve on this accuracy, resulting in a recursive refinement algorithm. Before proceeding, we examine the performance of Algorithm 5.1 again, introducing metrics that reflect this issue.

6.1.1 Purity

To evaluate performance in grouping agents, a suitable metric is required. Measurements employed in the previous chapter are problematic here, because there is no natural ordering or naming of coalitions. For example, we cannot ask, "what percentage of the members of coalition 2 were identified as belonging to coalition 2?", because the identifier '2' is meaningless. Instead, we might attempt to come up with suitable labels for the coalitions identified; that is essentially the basis for the measure, *purity* [1].

Purity is a measure from the field of clustering, which seeks to measure the degree to which each cluster consists of a single, 'pure' type. A measure such as this can be used when one has access to the true class of each object for evaluation purposes (as we do here); without such knowledge, one may be limited to evaluations based solely on the structure of the unlabelled data.

The purity of a cluster is the proportion of objects within that cluster that have the same class as the *majority* class[1] within that cluster. For example, if a cluster of size 100 contained 40 elements of class A, 50 elements of class B, and 10 elements of class C, the majority class would be B, and the purity of the cluster would be $50/100 = 0.5$. A cluster consisting entirely of members of a single class has a purity of 1 (i.e., it is 'pure').

The overall purity of a clustering (across all clusters), then, is the proportion of agents that have the same class as the majority for the cluster to which they have been assigned. A purity of 1 would indicate that every cluster consists purely of a single type.

Purity is a useful measure here: it reflects the proportion of agents that were grouped with members of their team.

Here, we revisit the situations depicted in Figures 5.7, Figures 5.9, where there are multiple coalitions engaged in bad-mouthing (on needed transactions only) and ballot-stuffing (an additional 25% above legitimate purchases), respectively. To illustrate our

[1]Here, *class* refers to the true type of the object. In our case, the members of each coalition would constitute one class (as would the remaining non-members). For example, in a case with two coalitions, members of the first coalition would be one class, members of the second coalition would be a second class, and non-members would be a third.

concern here, consider a case where there are two coalitions present, A and B. According to the measures used in Chapter 5, the algorithm would be successful if agents from A and B were placed into clusters that were labelled as coalitions, even if, for example, agents from A and B were placed into the same cluster. We turn our attention now to success in identifying team membership.

Figure 6.1 depicts the same trial and analysis run as Figure 5.7 (bad-mouthing), but this time showing purity. Recall from Figure 5.7 that coalition detection was extremely accurate, approaching 100% in all trials. Figure 6.1 reveals that, while the algorithm was fairly successful at separating members of different teams, it does not approach the coalition detection performance. Purity is clearly higher for trials with fewer coalitions present; more coalitions provide more opportunities to group agents incorrectly.

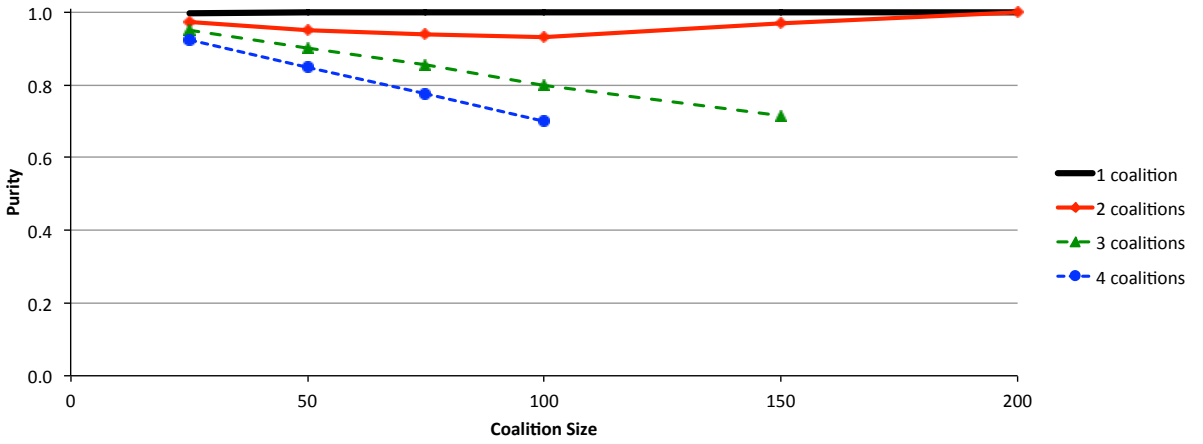

Figure 6.1: Bad-mouthing: Purity, with multiple coalitions.

Unfortunately, while these results are strong, they are also misleadingly high. This is because colluders make up a relatively small portion of the entire population. Consider, for example, the case where there are 50 colluders and 950 non-members in a population. Simply placing all of these agents into a single cluster would yield a purity of $950/1000 = 0.95$. The inclusion of non-members into our purity scores, then, obscures the true performance at separating coalitions from one another. Moreover, while we are interested in whether coalition members are grouped together, we are unconcerned about how the non-members (who are not labelled as coalition members) are grouped: only clusters that are identified as coalitions are returned as such in our results, and other clusters are discarded. Thus, it is more informative to disregard those not classified as members, and to calculate purity over only those groups classified as coalitions.

Figure 6.2 shows the same bad-mouthing case as Figure 6.1 above, but purity is calculated only over those agents classified as coalition members. (The series for single-coalition trials has been omitted, because it is trivial by this measure.) The series are similar in shape to those in Figure 6.1, but the scores are clearly lower—partitioning teams effectively is challenging.

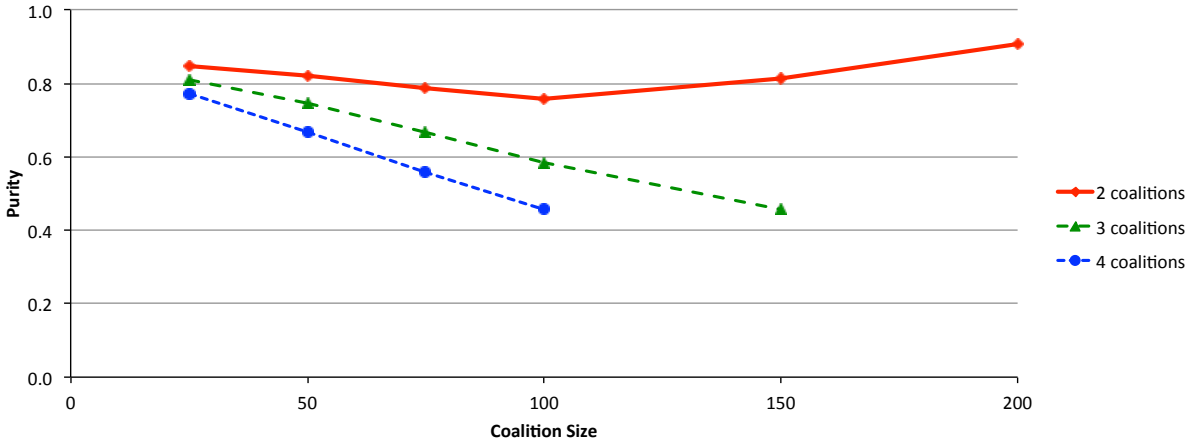

Figure 6.2: Bad-mouthing: Purity over those labelled as coalition members.

An examination of the detailed clustering data shows that separation failures typically arise due to multiple entire coalitions being grouped together as one (rather than agents from multiple coalitions being shuffled across several clusters).

Figure 6.3 depicts the same trial and analysis run as Figure 5.9 (ballot-stuffing), again, with purity calculated only over those classified as coalition members. Results here are quite strong.[2]

One may wonder why separation performance is better in the ballot-stuffing case than the bad-mouthing case. A key factor is that agents engaged in ballot-stuffing are targeting members of their own group, and thus target entirely different agents than other coalitions. In contrast, bad-mouthing coalitions are targeting outsiders; they may be targeting many of the same agents as other coalitions, looking similar in the process.

[2]The observant reader may notice that the results here are actually higher than the coalition detection accuracy shown in Figure 5.9, and wonder, 'how can we group a higher percentage successfully than we detected in the first place?' Note that these purity results only include those *classified* as coalition members—false negatives are not incorporated into the figures.

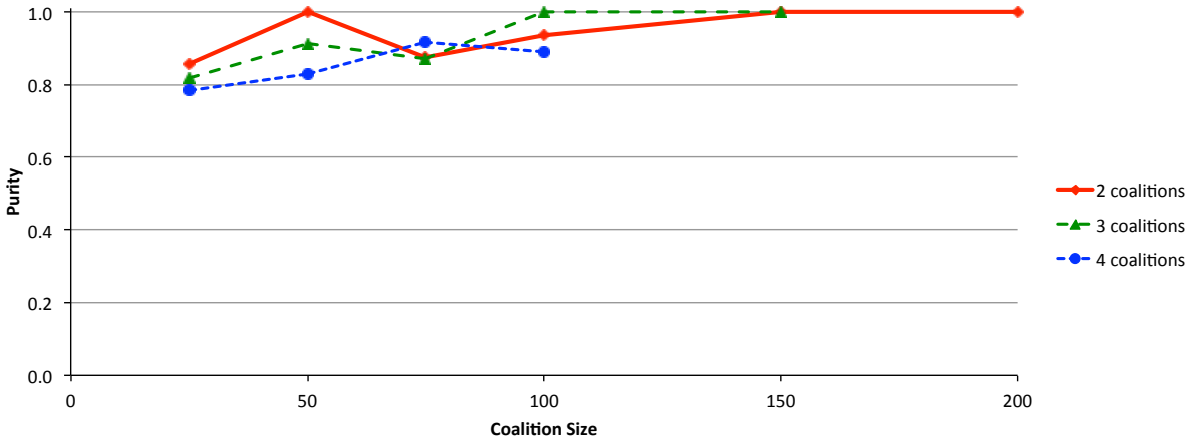

Figure 6.3: Ballot-stuffing: Purity over those labelled as coalition members.

6.1.2 Recursive refinement

As noted above, failures to effectively partition coalitions tend to occur because multiple coalitions have been grouped together into a single cluster. In seeking to effectively separate these groups, it is natural to consider applying the very same algorithm we have used to separate groups already. Thus, we developed an algorithm (6.1) which recursively applies our detection algorithm to detected clusters.

Algorithm 6.1 Recursive refinement

1. Run the coalition detection algorithm (e.g., Algorithm 5.1) on the input set of agents, yielding a set of coalitions $C_1, C_2, ...C_k$.
2. For each detected coalition C_i:
   - Run Recursive refinement using the set of agents in C_i as the input set.
     - If a set of coalitions is returned, add that set to the result set.
     - Otherwise, add the original coalition to the result set.
3. Return the result set.

Recursive refinement can only take already-detected coalitions, and further decom-

pose them. It cannot introduce new false positives, beyond those of the non-recursive algorithm: any agent passed into a recursive call was already labelled as a coalition member by the caller; similarly, it cannot correct for false negatives. (Note that it *is* possible for the recursive algorithm to remove false positives, by further decomposing a cluster that contains both coalition members and non-members.)

Stopping

The prospect of a recursive algorithm making use of clustering may seem problematic. When a clustering algorithm is used, it will return a set of clusters, regardless of the nature of the data. Won't the recursive algorithm cluster indefinitely (until clusters of size 1 are obtained)?

Here, our characterization method (developed in Section 5.3.3) provides an effective stopping rule. Consider first the case where a recursive call is made on a cluster containing two coalitions, A and B. If the clusterer effectively separates A and B, the characterization method is likely to show that agents in A favour themselves more than outsiders (i.e., those in B), and vice versa: two coalitions will be detected. In contrast consider the case where a cluster consisting entirely of one coalition, C, is passed into a recursive call. Regardless of the clustering obtained, the characterization method is unlikely to detect that agents in a cluster (i.e., some members of C) favour each other more than they favour 'outsiders' (i.e., the remaining members of C)—no coalitions will be detected. Thus, when no coalitions are detected, recursion stops.

6.1.3 Results

Recursive refinement was applied to the same data as the non-recursive Algorithm 5.1 (from Figures 6.2 and 6.3, above). The results are shown in Figures 6.4 and 6.5.

Comparing these results to the non-recursive algorithm, dramatic improvements can be seen. This is not the whole story, however. Purity tells us whether members of the different classes have been *separated*, not whether members of the same class have been *grouped together*. One might wonder whether the recursive algorithm achieves high purity values simply by separating the agents into an (unnecessarily) large number of clusters. (Note that a purity value of 1 can be trivially obtained simply by placing every agent into a separate group; each cluster will be entirely 'pure'.)

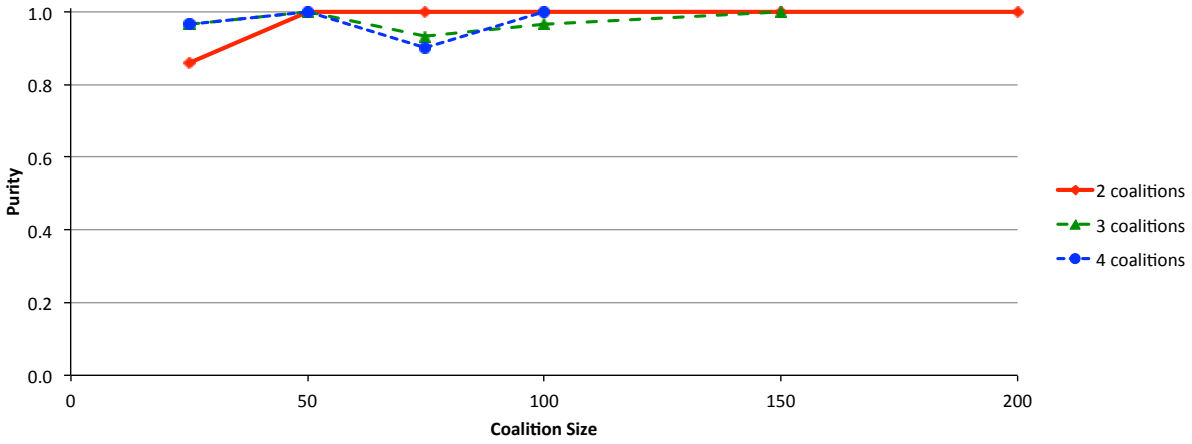

Figure 6.4: Bad-mouthing: Purity over detected members, recursive algorithm.

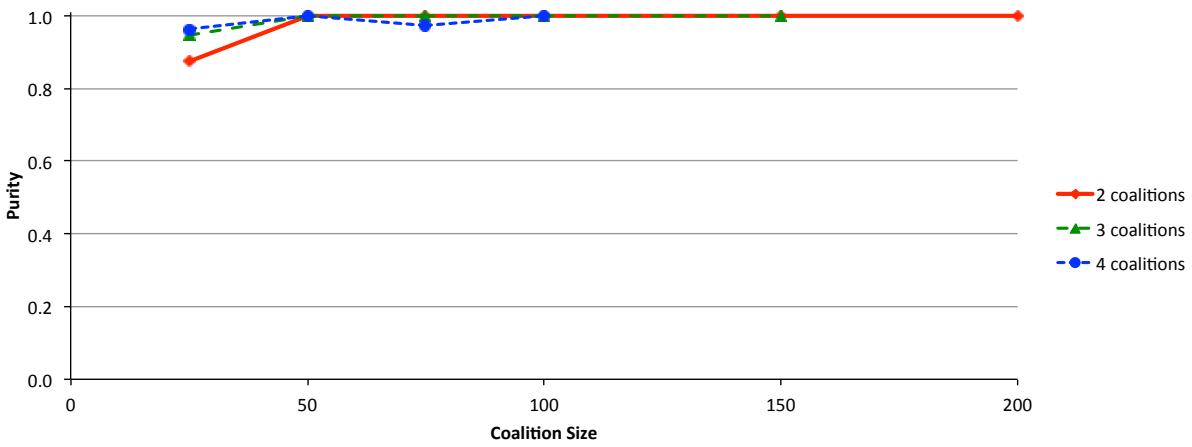

Figure 6.5: Ballot-stuffing: Purity over detected members, recursive algorithm.

Rand Index

In order to determine success at grouping members of the same coalition together, we consider a new metric. A measure that is conceptually similar to purity, but which penalizes for unnecessary clusters, is *Rand index* (RI) [56]. For every possible pairing of agents (a, b) in population P, RI considers whether the right separation 'choice' was made: if a and b are members of the same class, they should be together in the same cluster, while if they are of different classes, they should be in separate clusters. RI is the proportion of

such choices which are correct. Very simply, if we consider a clustering of P and denote the number of correct decisions d, then:

$$RI = \frac{d}{\binom{|P|}{2}} \tag{6.1}$$

An RI of 1 indicates that different groups have been entirely separated into different clusters, and the members of each group are entirely contained within a single cluster—i.e., that the clustering exactly corresponds to the true classes of the objects.

Figures 6.6 and 6.7 compare RI for the non-recursive and recursive algorithms respectively, for the bad-mouthing case. Figures 6.8 and 6.9 present the ballot-stuffing case. (As above, RI is calculated only over those agents classified as coalition members.)

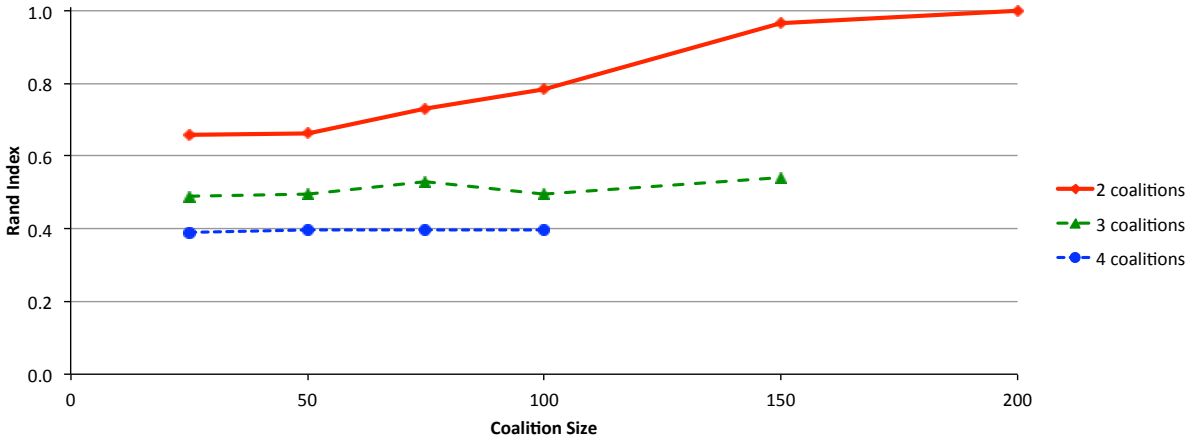

Figure 6.6: Bad-mouthing: Rand index over detected members, non-recursive algorithm.

Two points should be noted. First, the recursive algorithm results in far better RI scores, indicating that it is doing a substantially better job of identifying team membership. Second, the recursive algorithm is achieving improved purity scores without introducing large numbers of unnecessary clusters.

6.2 Iterative refinement

In this section, we explore the issue of moving into more appropriate clusters individual agents that have been mis-grouped (which may occur, albeit infrequently, as discussed in

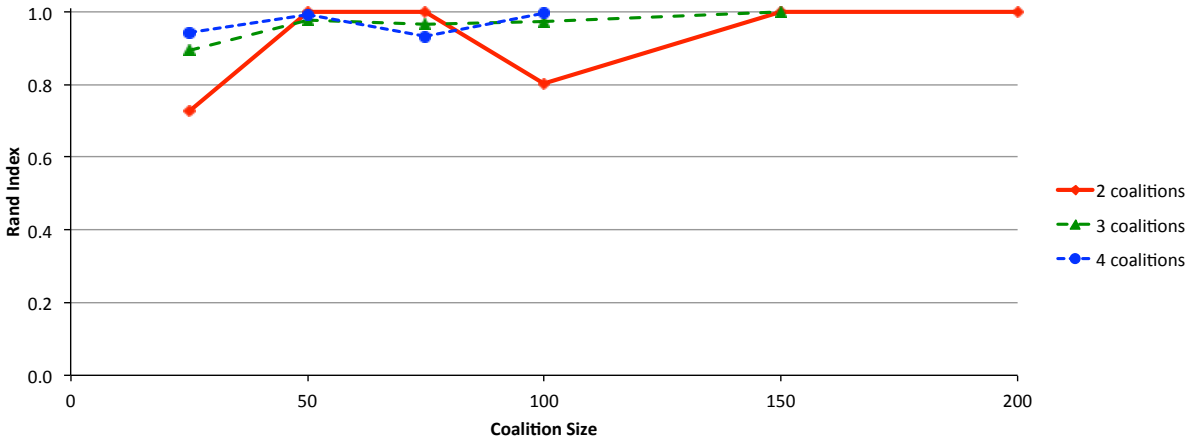

Figure 6.7: Bad-mouthing: Rand index over detected members, recursive algorithm.

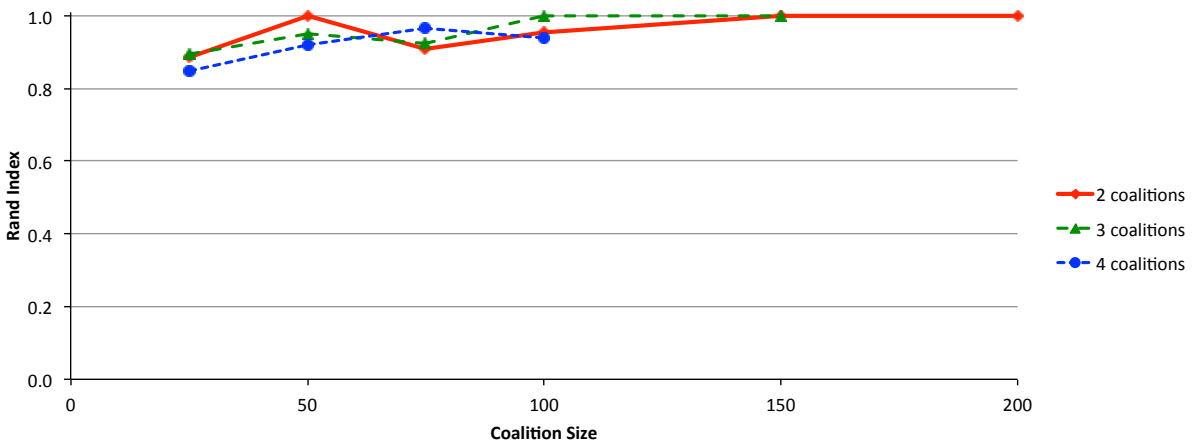

Figure 6.8: Ballot-stuffing: Rand index over detected members, non-recursive algorithm.

the previous section). This entire section should be viewed as a sidebar. In fact, we ultimately conclude that what might seem intuitively useful (an iterative form of refinement) will not be successful. We use this result to reinforce the importance of similarity-based detection, which we then return to in Section 6.3, for accurate detection. As such, that section can easily be read immediately following Section 6.1.

Recursive refinement can further break down clusters, in order to improve separation

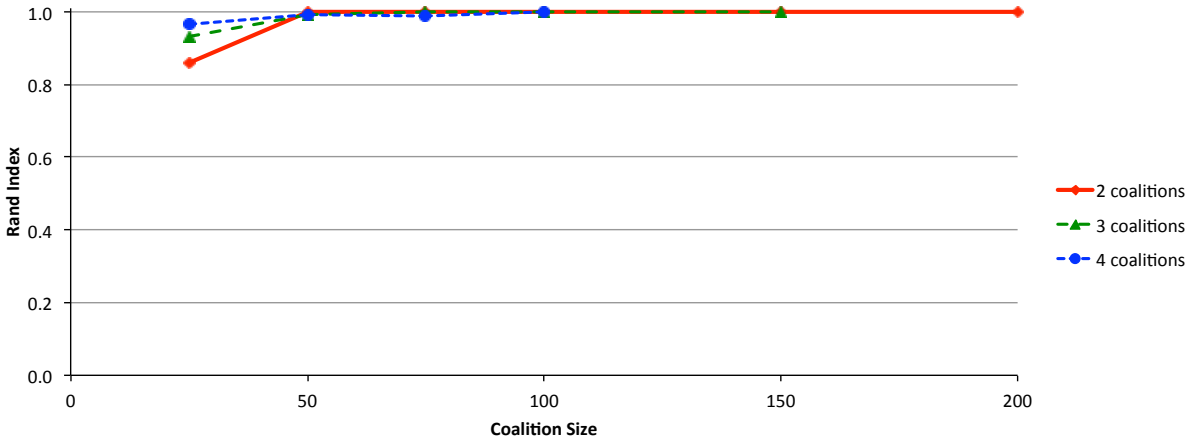

Figure 6.9: Ballot-stuffing: Rand index over detected members, recursive algorithm.

of different teams. Beyond this, however, it cannot make changes to the memberships of the clusters themselves. For example, consider the case where most of the members of coalition A are grouped into a single cluster, but a small number of members were mistakenly grouped with cluster B, or were mistakenly excluded from the detected coalitions entirely. Ideally, we would be able to correct for these errors, by moving individual agents into the appropriate clusters.

A natural approach for doing so would be to consider each agent individually, and, if that agent looks to have been placed in an inappropriate group, to move it to a more suitable cluster. We refer to this approach as *iterative refinement*.

If we plan to move agents individually, how do we know to which cluster a given agent should be assigned? Considering again our characterization method (developed in Section 5.3.3) can yield insight. In Figure 6.10, we portray the 'within' test (helping teammates); the 'exiting' test (benefiting outsiders less, or harming them) is symmetric. To characterize a cluster C of size n, we first repeatedly take random selections G of n agents from the population, and calculate the mean benefit members of G bestow upon one another. This yields a distribution of sample means, the normal curve shown in Figure 6.10. Then, we calculate the average benefit that members of the candidate cluster C give to one another. The p-value is the area under the normal curve to the right of C's mean benefit. Given the distribution of sample means, the p-value represents the probability, for a randomly selected group of agents, of getting a mean at least as large as the one for C. The farther to right, the lower the p-value, and hence, the stronger the evidence that the result was not obtained by chance (i.e., the stronger the evidence that C

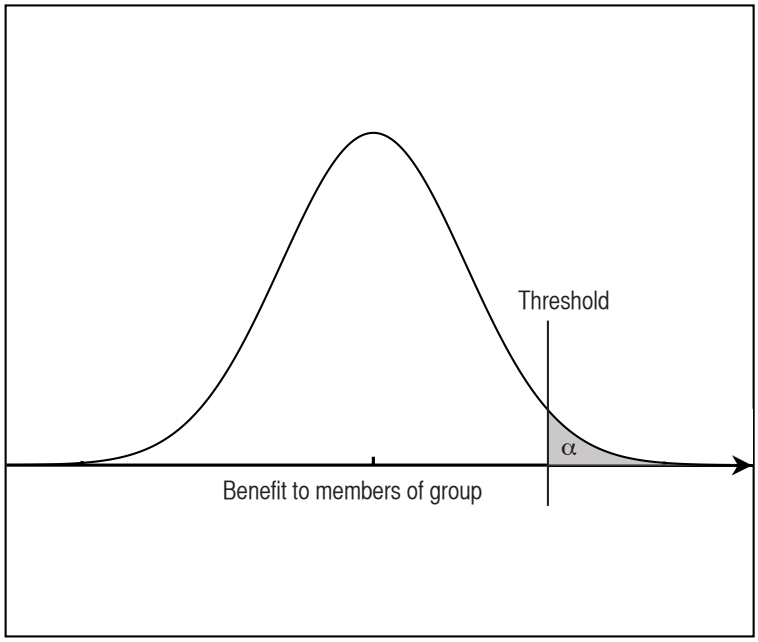

Figure 6.10: Distribution of sample means, benefit to other group members.

contains a coalition.) If the p-value is less than our chosen α (the threshold probability), we conclude that cluster contains a coalition. There is a threshold level (known as a critical value) for average benefit, then, where values falling below the threshold result in p-values greater than α, and values greater than the threshold result in p-values less than α; we have illustrated this threshold in the figure.

It is trivial, yet important to note that the average benefit for C is composed of the individual benefit values for members of C. If C's average benefit is to the right of the threshold, many or all of the members of C will individually have values falling to the right of the threshold as well.

Now we examine the case where agents have been mis-grouped, and we are trying to iteratively fix the assignments. First, consider an agent a that is a member of the coalition contained in C, but who was mistakenly placed in different cluster. As a true member of the coalition, a is likely to be benefitting members of C more than non-member agents—

a is likely to fall to the right side of the distribution. In fact, if the average amount of benefit a bestows on members of C falls to the right of the threshold, then it is reasonable to think that a, too, may actually be a member of the coalition contained in C. Thus, we might add a to C.

Now, consider a non-member agent b who was mistakenly grouped with the coalition in cluster C. b is unlikely to be benefiting members of the coalition in C much more than a 'typical' agent would, and is very unlikely to be benefitting members as much as the members themselves would be. If b benefits members of C little enough that he falls to the left of the threshold, then it is reasonable to suspect that b does not, in fact, truly belong to the coalition. It follows that we might remove b from C.

This is the general basis for our iterative refinement algorithm: If an agent inside a cluster provides benefit below the threshold, remove it from the cluster; if an agent outside a cluster provides benefit beyond the threshold level, then add it to the cluster. (In fact, if there are multiple candidate coalitions present, we move an agent to the cluster for which it is furthest past the threshold. We explore the algorithm in greater detail below.)

6.2.1 Termination?

While we have sketched out a general basis for iterative refinement, important issues have yet to be addressed: in particular, how do we know that the algorithm will terminate? In fact, this turns out to be a problem. (We have also postposed discussion of the issue of how to decide which agent to move next, a secondary concern.)

Consider a hypothetical case of four agents, depicted in Figure 6.11. The edge weights in the graph indicate the amount of benefit flowing from the source to the destination. Suppose the threshold quantity of benefit required is 10. (For simplicity, we assume the same threshold for all cluster sizes.) At the beginning, there are three agents contained in the cluster: $\{b,c,d\}$. b's average benefit (per edge) to cluster members is $(11+10)/2 = 10.5$ as is c's; d's average benefit to members is $(15+15)/2 = 15$. All three members have averages above the threshold.

Now, consider agent a, currently an outsider. His average benefit to coalition members is $(12+9+12)/3 = 11$. This value is above the threshold, so a is added to the cluster, which now consists of $\{a,b,c,d\}$.

With the addition of a, we now need to recompute the benefit for each of the existing members. b and c have averages of $(11+10+4)/3 \approx 8.33$; d's average is $(15+15+15)/3 = 15$. b and c now fall below the threshold, so they are removed, yielding a cluster of $\{a,d\}$.

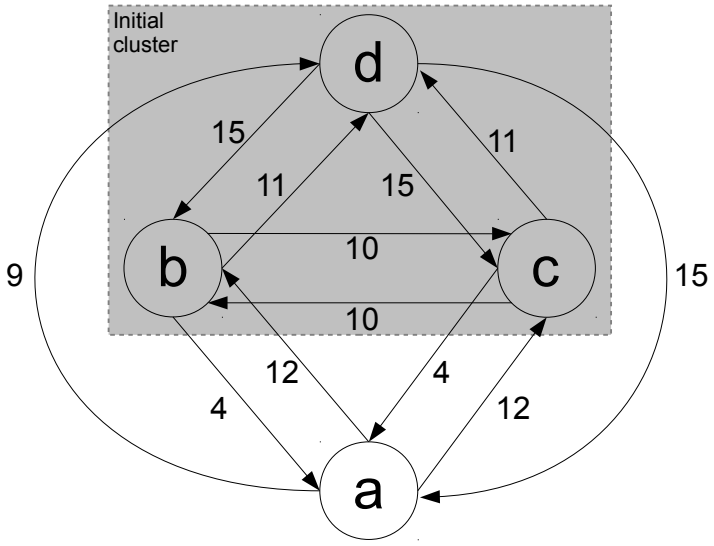

Figure 6.11: A hypothetical subset of agents.

Again, the averages for the cluster need to be recomputed. a's average is simply $9/1 = 9$, while d's is $15/1 = 15$. a's average falls below the threshold, so it is removed, yielding a cluster of only $\{d\}$.

With this removal, however, we need to reconsider b and c. The average benefit to members of the cluster from b is now $11/1 = 11$, while c's is the same. Both b and c are now above the threshold, so they are added, yielding a cluster of $\{b,c,d\}$.

Unfortunately, this cluster assignment is the same one we began with: we have encountered a cycle. Because of this, we cannot guarantee termination—in fact, experiments have shown that cycles frequently occur in practice.

6.2.2 Progress

Ideally (and to ensure termination), for an iterative optimization algorithm, we would have a well-defined objective function and a notion of progress. Unfortunately, this is problematic in this case.

Why minimizing p-value doesn't work

We have noted that lower p-values indicate stronger evidence for the presence of a coalition, and that the individual members' contributions combine to determine the p-value. This might lead one to think that minimizing p-value, meaning the strongest possible evidence, might equate to an optimal solution. This is not the case, however.

To illustrate, consider a simplified[3] case of a coalition C with 3 members, a, b, and c, which contribute average benefits to the coalition of 8, 10, and 12 respectively. Suppose that the threshold to be classified as a coalition is 6. The average benefit over a, b, and c is $(8+10+12)/3 = 8$; this value exceeds the threshold, so C would be classified as a coalition. But now, what if we remove a from the cluster? The resulting average for b and c, $(10+12)/2 = 11$, is now further past the threshold, resulting in a lower p-value. Removing a thus results in a cluster with stronger evidence of being a coalition—but *has excluded true coalition members (in this case, a) in the process.* Clearly, this is not an improved result.

Total Margin

The problem in Section 6.2.1 is that when considering an agent's addition or removal from a cluster, we examine benefit in the context of a single agent. As such, we focused first on *only what that agent has done*, not *what has been done to it*. Consider again the situation in Figure 6.11, and our first step, when we added a. When making this decision, we considered only the benefit flowing from a to members of the cluster, but this is only half the story. We did not incorporate benefit *from* cluster members *to* a, which carries equal weight in computing the cluster's average benefit and p-value (as calculated using the method from Section 5.3.3, and used in Algorithm 5.1).

Before the addition of a, the per-edge average for the cluster was $(11+11+15+15+10+10)/6 = 12$. After a's addition, the average was $(11+11+15+15+10+10+4+12+9+15+4+12)/12 \approx 10.67$. Although the benefit *from* a was above the threshold, its addition lowered the overall average.

As noted above, the fact that a's inclusion lowers the overall average is not necessarily a problem, and a should not be excluded on that basis. But a's inclusion did more than lower the average: it had the side effect of expelling two other agents, b and c. Such side effects cause problems such as cycles.

[3] Details such as a digraph with edge weights are not necessary to illustrate the point here.

To avoid such issues, we sought a measure that would consider both the number of agents that belong in the cluster, and the degree to which each agent surpasses the threshold. We refer to amount by which an agent surpasses the threshold (i.e., the agent's average benefit to others in the cluster, minus the threshold), as its *margin*. The higher the margin, the stronger the suspicion that the agent belongs in that particular cluster. As an objective function, we seek to maximize the total margin for a cluster. When we consider a possible additional/removal of an agent to/from a cluster, we consider the change in total margin (i.e., both the benefit flow from that agent, and that to the agent.)

Adding an agent with a positive margin (e.g., one that exceeds the threshold), increases total margin for the cluster, *even if it would lower the overall average for the cluster*. Similarly, removing an agent with a negative margin (e.g., one that falls below the threshold) increases it.

Total margin also copes well with situations like those in Figure 6.11. The total margin at the beginning, before the addition of a:

Member	**Average benefit to cluster**	**Margin**
b	10.5	0.5
c	10.5	0.5
d	15	5
Total	-	**6**

If a were to be added, the total margin would be:

Member	**Average benefit to cluster**	**Margin**
a	11	1
b	8.33	-1.67
c	8.33	-1.67
d	15	5
Total	-	**2.66**

Thus, a would not be added, because the total margin would decrease. Essentially, in situations such as this, using total margin allows us to make choices: is it better to include a, or $\{b,c\}$?

We can guarantee that with this approach, total margin is monotonically increasing (i.e., progress is made) with each step, by only making moves that increase total margin.

(We have taken the greedy choice, always choosing the move that results in the largest increase in total margin.) Thus, we can ensure termination.

We cannot guarantee that using total margin in this way arrives at a globally optimal solution. In fact, there are likely to be many local maxima because, as seen in our example, including certain agents often means excluding others; the inclusion of the other agent may have lead to an entirely different allocation. Unfortunately, there are other issues.

Limitations of total margin There are two key problems with using total margin as the basis of iterative refinement, the first conceptual, the second more practical:

1. To avoid cycles, we base our decisions on not only what the agent has done, but what is done to it by others in the cluster. While there may be algorithmic advantage to doing so, one could argue that it is problematic to characterize an entity's behaviour (i.e., accuse them of being a colluder) based substantially on the actions of others.

2. Empirically, the results using this method were extremely poor. As noted, we cannot guarantee that this algorithm arrives at a global maximum; more to the point, we cannot guarantee that a local (or even global) maximum results in a clustering that matches the true coalitions. This is likely related to the first point.

Other means of ensuring termination

We are left with a quandary: considering only the benefit given by an agent (and not that received) potentially results in cycles, while considering both benefit given and received results in poor allocations. Judging the latter problem more severe than the former, we returned to considering only the benefit given by agents.

Without the ability to rely on an objective function to ensure termination, another means was needed. To guarantee termination, we simply need to avoid cycles. This can be accomplished by simply restricting repeated moves by an agent.

While cycles can have periods of two moves, they can also be longer; longer cycles were encountered during experimentation. It seems reasonable to attempt to prevent cycles of two moves at first, and only turn to longer cycles if needed.

The approach we adopted was to render an agent inactive after a move for the next 2^m moves, where m is the number of times the agent had been moved during refinement.

Thus, after a first move, it would only be inactive for 2 turns, thwarting a two-move cycle. After a second move (meaning that a cycle longer than 2 moves might have occurred), it is frozen for 4 turns, and so on. This method allows for increasingly long cycles to be broken, and lets agents other than those in the cycles to have an opportunity to be moved.

The method described guarantees termination: even under continuing, long cycles, agents will eventually be frozen for more turns than there are agents in the system.

We detail the algorithm below. This algorithm is another form of refinement that might be applied to the results of, e.g., Algorithm 5.1.

6.2.3 Algorithm

In Algorithm 6.2, we use the term 'margin' for brevity; here we refer simply the average benefit given by the agent, minus the threshold. It does not include the benefit received by the agent. Note, too, that it is presented in simple form for clarity. In reality, the margin calculations from step 1 are cached for as long the membership of an agent's cluster remains unchanged.)

6.2.4 Results

To demonstrate the performance of this algorithm, we first consider its application to the same multi-coalition ballot-stuffing test depicted in Figures 5.9 and 5.10. In Figure 6.12 we show the performance of both Algorithm 5.1 without refinement, and with iterative refinement. For clarity, the trials for varying numbers of coalitions (0 through 4) have been collapsed into single series.

Iterative refinement does increase the detection rate, a desirable result. It also results in a not-insignificant increase in the number of false positives.

For contrast, we consider its application to the bad-mouthing test from the same section (Figures 5.7 and 5.8). The results for iterative refinement are depicted in Figure 6.13. Here, the detection rate has substantially deteriorated. Worse still, there are an enormous number of false positives.

Algorithm 6.2 Iterative Refinement

1. Run the coalition detection algorithm (e.g., Algorithm 5.1) on the input.
2. Set all agents to 'active' status.
3. Repeat until no more moves are made:
   (a) Set 'best move' to null.
   (b) For each agent a:
      i. If a is inactive and the required number of turns have passed, set it to active.
      ii. If a is active:
         - If a is in a cluster that has been characterized as a coalition, calculate its current margin. If not, set its current margin to 0.
         - For every cluster C:
            – Calculate the margin if a were to join C.
            – Calculate the change in margin: margin if C were joined, less the current margin.
            – If this change in margin is better than for the best move, store moving a to C as the new best move.
   (c) For the agent a having the 'best move':
      - Move a from its current cluster, to its new cluster.
      - Mark a as inactive for the next 2^m turns, where m is the number of times a has been moved during refinement.

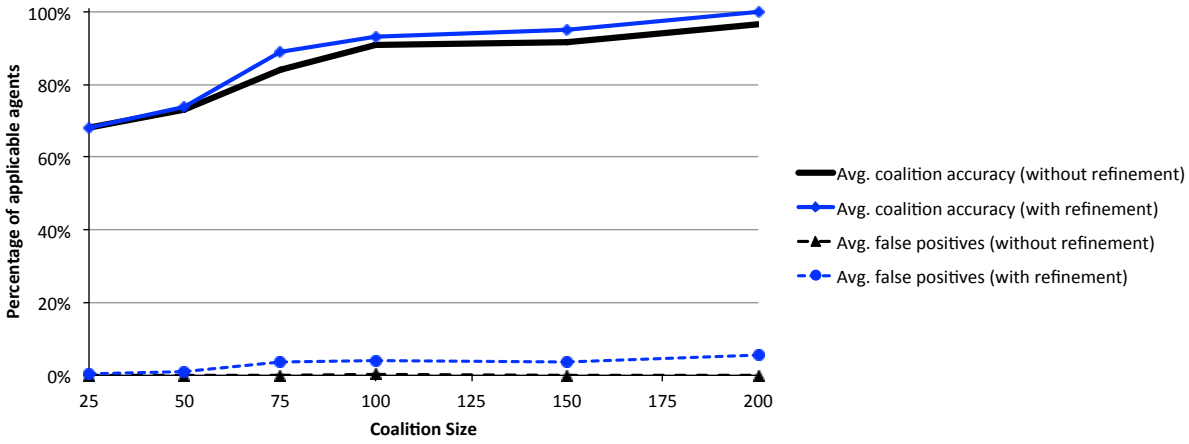

Figure 6.12: Ballot-stuffing: Iterative refinement performance, multi-coalition case.

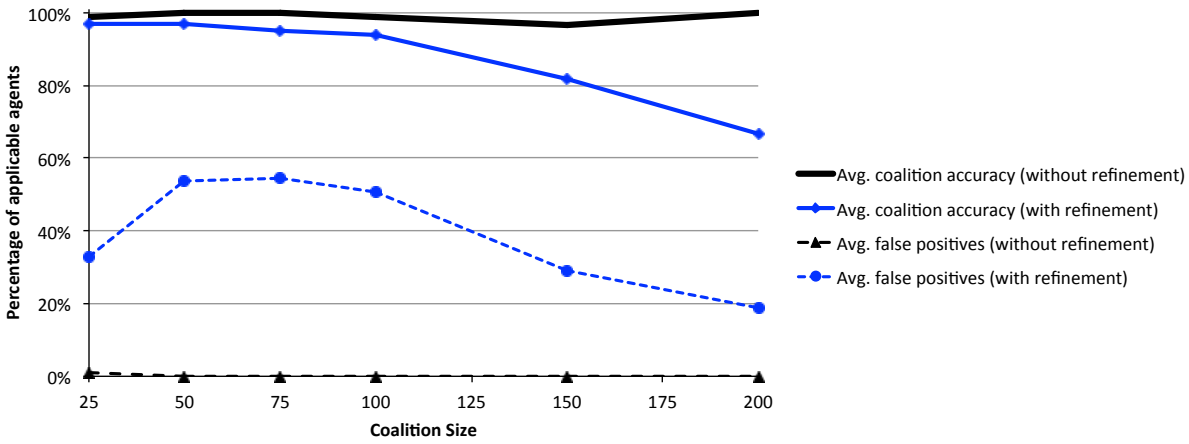

Figure 6.13: Bad-mouthing: Iterative refinement performance, multi-coalition case.

6.2.5 Implications: The importance of similarity

Unfortunately, the poor performance seen is Figure 6.13 is mirrored in a significant number of trials. Thus, we do not recommend use of the iterative refinement algorithm. While we do not discount the possibility that iterative refinement algorithms might yield breakthroughs in the future, we have turned our attention elsewhere at present.

Our iterative refinement algorithm is particularly prone to false positives: outside

agents are encountered who are greatly benefitting coalition members. This is likely due to the effects of the coalitions' collusive activity itself. Successfully ballot-stuffing or bad-mouthing wins additional purchases (and positive reviews) from outsiders, which registers as benefit. If enough sales are won, the benefit totals from outside purchasers may be large.

While this is a negative result, it holds an important positive implication. These results suggest that there is more to the nature of cooperation, and hence to the detection of coalitions, than simply benefit flows themselves. *Similarity* of behaviour appears to be fundamental, suggesting that the clustering step of our algorithm is essential. This finding should be noted by those, in future, who might consider approaches to coalition detection that do not incorporate clustering.

6.3 Improved clustering

Having outlined how recursive refinement can be employed to improve team membership identification, we now turn our focus to improving the ability of our algorithm to detect coalitions, particularly under difficult circumstances. In the remainder of this chapter, we propose further enhancements to Algorithm 5.1, to improve accuracy and robustness.

We return now to the ballot-stuffing scalability test depicted in Figure 5.25. Recall that, while performance against bad-mouthing was excellent throughout, performance against ballot-stuffing deteriorated significantly as the agent population increased.

While exploring this issue, an examination of the detailed analysis records was informative. In particular, it was observed that when classification errors were made, the 'misses' were usually the result of bad clustering:

- False negatives occurred when the clustering algorithm did not successfully separate the coalition members from the mass of non-members. (It was not generally the case that clusters consisting entirely of coalition members were incorrectly classified as non-coalitions.)
- False positives (although very few) typically occured when some non-members were mistakenly grouped into a cluster with coalition members. (It was generally not the case that an entire cluster consisting of purely non-members was mistakenly classified as a coalition.)

Table 6.1: Agent counts by class and cluster; silhouette score of 0.00904.

Cluster number	Non-colluders	Coalition 1	Coalition 2
0	1	0	0
1	28	0	0
2	4871	50	50

Given a clustering problem, one might immediately seek to use a different clustering method, or a different set of features. (We do discuss these options later in this document.) However, an exploration of *why* we were missing was revealing. Recall that Algorithm 5.1 clusters the data with a successively larger number of clusters; to choose the optimal number of clusters, we select the set with the highest silhouette score, as explained in Section 5.3.2. Unfortunately, silhouette decreases in usefulness as the agent population increases.[4] Data from one of the trials (where two coalitions of size 50 were present) will illustrate. First, in Table 6.1 we present the cluster composition and score when three clusters were used.[5] Here, the clusterer has not separated the coalition agents from the bulk of the larger population.

Compare these to the results when nine clusters are used, shown in Table 6.2. Here, the clusterer has successfully separated the coalitions from the other agents, into pure clusters. Note, however, that the silhouette score for the nine-cluster case is lower than that for the three-cluster case. Our current algorithm (5.1) would select the 3-cluster assignment, and would subsequently fail to detect the coalitions. Note that this failure is entirely the result of the choice of number of clusters: the clustering method was capable of delivering the desired separation.

The key problem

Based on this, one might conclude that silhouette is ineffective, and seek to replace it with another such metric to choose the correct number of clusters. While this may improve performance (an exhaustive exploration of other measures is beyond the scope

[4]We mean only to say that silhouette's usefulness decreases *for our usage*, not that it is ineffective in general for larger populations!

[5]Note that the cluster counts shown in the table are based on the agents' true classes. This data is used for evaluation purposes only—it is not available to the algorithm during analysis.

Table 6.2: Agent counts by class and cluster; silhouette score of 0.00758.

Cluster number	Non-colluders	Coalition 1	Coalition 2
0	1	0	0
1	4	0	0
2	2663	0	0
3	153	0	0
4	0	50	0
5	1	0	0
6	164	0	0
7	1914	0	0
8	0	0	50

of this work), there is reason to believe that looking beyond existing such metrics will be more fruitful for our purposes.

Silhouette seeks to measure how well a clustering 'fits' the structure of the data—*all* of the data. In contrast, we are unconcerned about how the cluster treats the non-member data; our overriding concern is simply that coalitions be separated from non-coalitions (and grouped together).

Recall that measures of cluster quality seek to balance conformity to the data against the number of clusters: fewer clusters are preferred to large numbers of clusters, unless a larger number conforms substantially better to the structure of the data. It is likely the case that the 3-cluster allocation above provides better fit with *all* of the data than the 9-cluster allocation. A (conceptual) diagram of a hypothetical case, presented in Figure 6.14, illustrates this point.

In this diagram, the colours indicate the type of the object (black represents non-colluders, and green and red represent two different coalitions.) Of course, the type is not visible to the clusterer. The clustering on the left, using fewer clusters, provides a good fit with the overall structure of the data, and likely matches what a human would judge to be the groups in the data. The clustering on the right is probably worse for the entire set of data: many more clusters are used, but without significantly better fit to the shape of the data. However, the clustering on the right is preferable for *our purposes*, because the coalitions have been separated from the rest of the population.

It is for this reason that other standard measures of clustering quality, while poten-

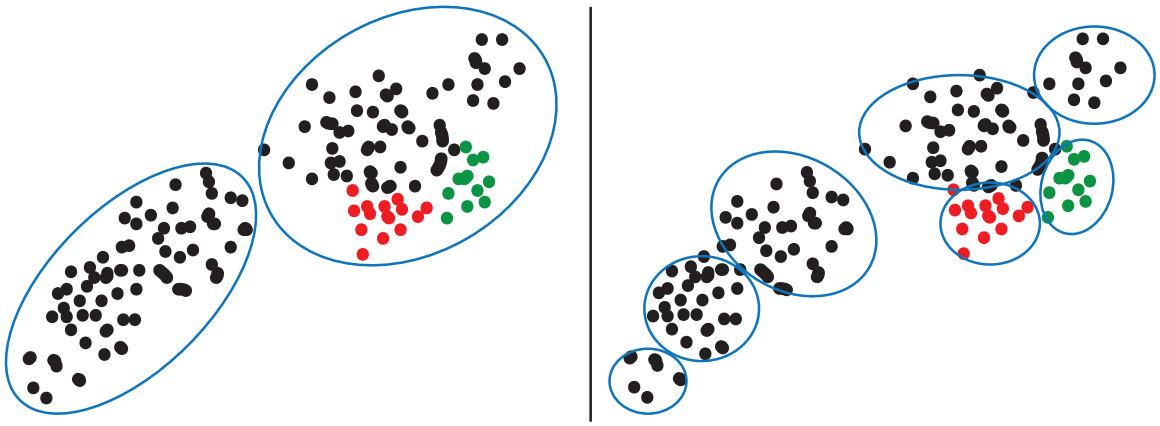

Figure 6.14: Two alternate clustering of the same data.

tially useful, are unlikely to be optimal for our purposes.

6.3.1 A new method for optimizing the number of clusters

To develop a superior method for optimizing the number of clusters, we turn to our own tools.

The characterization method we developed in Section 5.3.3 has shown itself to be highly resistant to false positives: typically, if a cluster is classified as a coalition, then it legitimately consists of coalition members. This means that when our characterization algorithm detects a coalition, *those coalition members were separated from the rest of the population into a cluster.* It follows that our characterization algorithm can be used to judge the quality of a clustering for our purposes: *Because the method is resistant to false positives, it is likely that the more coalition members we find, the better separation we achieved.*

Thus, we replace silhouette scoring in the clustering step of our algorithm (5.1) with the method described in Algorithm 6.3.

Making this change, we apply the new algorithm (i.e., Algorithm 5.1 with the use of silhouette replaced by this new method to find the number of clusters) again to the ballot-stuffing scalability test from Figure 5.25. The results are shown in Figure 6.15, with the performance of the Silhouette-based algorithm included for comparison. There

Algorithm 6.3 Characterization For Cluster Count

1. For each clustering result R:
   (a) Apply the characterization method to each cluster C in R; the score for C is the number of agents classified as coalition members.
   (b) The total score for clustering R is the sum for all clusters C.
2. Choose the clustering with the highest score.

is a substantial increase in coalition detection accuracy. There has also been an increase in false positives—selecting clusterings based on maximizing the number of positives does increase the possibility of false positives—but the levels are still very low. Overall, performance has significantly improved.

This new method for choosing the best clustering, based on characterization, is used throughout the remainder of this document, except where noted.

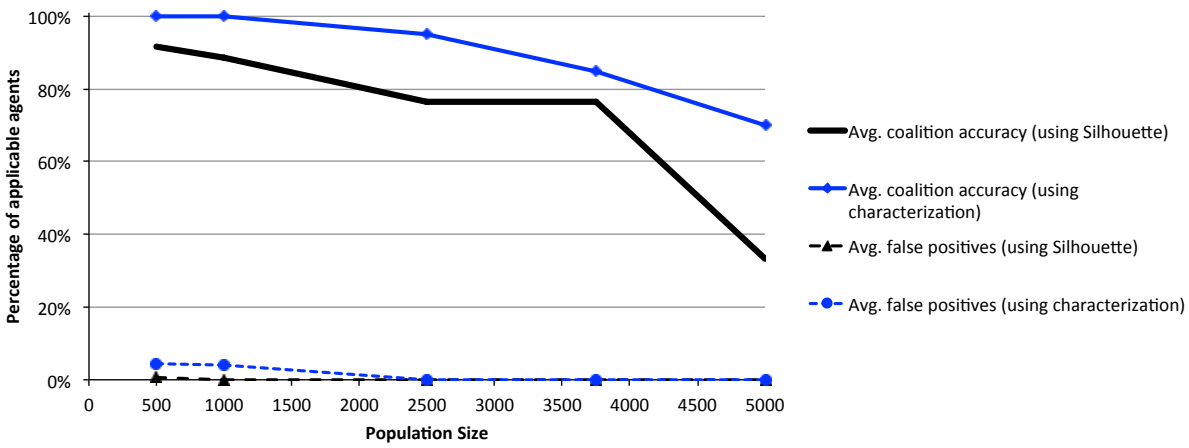

Figure 6.15: Performance using characterization to select the best clustering.

6.3.2 Multi-clustering

Note that, in step 1 of the improved method for determining the number of clusters (Algorithm 6.3), we did not say 'for each choice of number of clusters', but rather, 'for each clustering result'. If we seek a clustering that maximizes the number of coalition members detected, we need not be limited to only using a single clustering method. Rather, we can easily apply multiple clustering methods, and choose the result which gives us the greatest number of positives, regardless of which clusterer generated it.[6]

We will refrain from exploring a lengthy list of clustering methods; while a number were explored, none had clearly superior performance. Here, we limit our discussion to one that was found to be particularly complimentary to k-means: hierarchical clustering, specifically the 'single-link' method [28]. In this method, the dataset is initialized by placing each element into a separate cluster. Then, we repeatedly choose the two remaining clusters that are closest, and merge them. (In the single-link method, 'closest' is defined as the minimum distance between any point in the first cluster, and any point in the second cluster.) We continue to merge clusters until we have reached the desired number.[7]

In experiments, we found that k-means provided superior separation on some cases, and the hierarchical algorithm on others. However, in combination, they are extremely powerful. Our multi-clustering technique is presented in Algorithm 6.4. (Although not reflected in this algorithm summary, in practice, characterization can be performed efficiently by caching or precomputing the distributions of $\bar{\beta}_G$ and $\bar{\beta}_{\bar{G}}$ for given cluster sizes.)

While we used k-means and hierarchical clusterers in our implementation, other methods could be readily included.

Results

We apply this method to the same situation depicted in Figure 6.15, where only k-means clustering was used. The results are shown in Figure 6.16, with the results for

[6] We still limit ourselves to similarity-based clustering methods, however, rather than any arbitrary means of 'slicing up' the population. Based on the results in Section 6.2, there is reason to believe that similarity is an essential characteristic of coalitions, and so it continues to be the basis of our algorithms.

[7] Obtaining clusterings for a varying number of clusters can be implemented fairly efficiently using this method. For example, if we wanted to try numbers of clusters ranging from 2 to 10, we could simply begin by repeating the merge step until 10 clusters were remaining. Then, a single merge operation yields the set for 9 clusters, and so on.

Algorithm 6.4 Multi-clustering Coalition Detection and Identification

1. Map agents to points in benefit space.
2. For each clustering method M:
   (a) **Clustering:** For each number of clusters k to be tried:
      i. Apply clustering method M (for k clusters) to the input data set, obtaining clustering result R.
      ii. **Characterization:** For each cluster C in R:
      Characterize C as a coalition if its benefit to members is unusually high, or its benefit to outsiders is unusually low, as follows:
         A. Repeatedly select random sets of agents G (of size $|C|$) from the population, and for each G, calculate $\bar{\beta}_G$ (the average benefit from G to its own members) and $\bar{\beta}_{\bar{G}}$ (the average benefit from G to outsiders).
         B. If $\bar{\beta}_C$ is unusually high (given the distribution from the random samples G), or if $\bar{\beta}_{\bar{C}}$ is unusually low (given the distribution of from the random samples S)G
            - Classify C as a coalition, and all members as coalition members.

            Otherwise:
            - Classify C as a non-coalition, and all members as non-coalition-members.
      iii. If the number of colluders detected for R is the largest yet encountered, store the resulting coalitions as the best set R_{best}.
3. Return the best set of coalitions obtained, R_{best}.

the k-means-only method included for comparison. Coalition detection accuracy is outstanding; while all of the false-positives from the k-means algorithm are included in the multi-clustering results as well, levels are still very low. The multi-clustering algorithm represents a dramatic improvement over the k-means-only version (which was, itself, a significant improvement over the silhouette-based approach.)

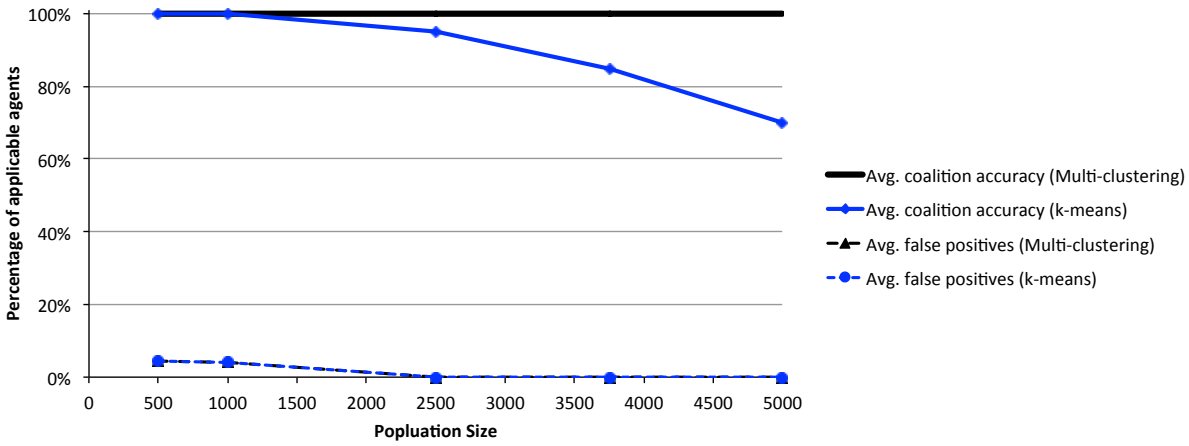

Figure 6.16: Performance using multi-clustering.

6.4 Improved characterization algorithm

The multi-clustering technique introduced in the previous section shows great potential for detection. However, relying on the characterization method (described in Section 5.3.3) as much as it does, it also reveals a flaw in that method (which was not an issue when Algorithm 5.1 was used with silhouette score). Figure 6.17 shows the results when the multi-clustering algorithm is applied to the bad-mouthing test case in which different trials have different bad-mouthing rates (i.e., the same test used in Figure 5.15). (In this, and subsequent charts, trials for coalitions of size 50 and 100 have been combined into single series, for clarity.) Performance here is horrific, with an immense false-positive rate.

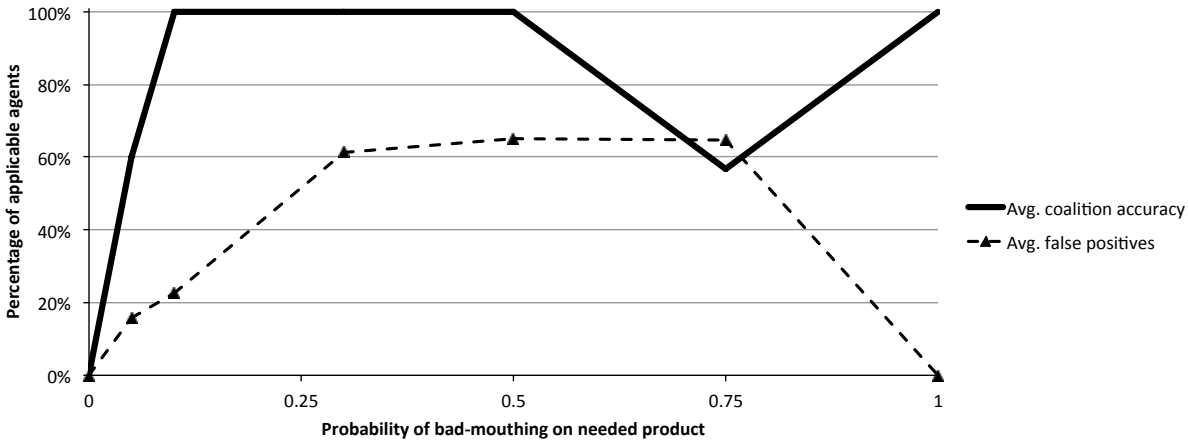

Figure 6.17: Bad-mouthing: performance of multi-clustering algorithm, with initial characterization method from Section 5.3.3.

6.4.1 The problem

Examination of the detailed analysis logs shed light on the issue. False positives tended to occur when:

- There was at least one coalition present in the population; and
- There was a large cluster consisting entirely of non-colluders.

Under these circumstances, the large cluster was often mistakenly classified as a coalition by the characterization method.[8] An example will help to illustrate why.

Consider a scenario where, in a population of 1,000, there are 200 ballot-stuffers (a group we will call C). One of the clusters within our clustering consists entirely of 400 non-colluders. Recall that, to characterize this cluster (which we will call D) using the method of Section 5.3.3, we take repeated random samples G of size 400, and for each

[8]This problem did not crop up when using Algorithm 5.1 with silhouette scoring. As noted in Section 6.3, silhouette was quite 'conservative' in favouring clusterings that resulted in fewer colluders being detected; similarly, it avoided situations in which this problem arises. The use of characterization to optimize the number of clusters, and the use of multi-clustering, is much more 'aggressive', accepting greater risk of false positives in order the increase the ability to detect colluders. The improvement made in this section reduces this risk dramatically, however, allowing us to use this more aggressive approach without high false positive rates.

G, calculate how much members of G benefit each other, on average, and how much they benefit outsiders. Using these samples we get a distribution of the sample means.

To determine if D is a coalition, we compute the average benefit members give to each other, and the average benefit they give to outsiders. If the benefit to members is high (or the benefit to outsiders low), relative to the distribution over G, then we conclude that the group is favouring its own members, and classify it as a coalition.

Now, consider the sampling process, as depicted in Figure 6.18. Because we are sampling a large portion of the population (400 out of 1000), samples G will often have many members of C inside them, and many members outside them. Consider one sample G, and an arbitrary pair of agents $\{a, b \in C, a \in G, b \notin G\}$, as illustrated in the diagram. Because a and b are ballot-stuffing, they benefit each other more than 'typical' agents. Because a is in G and b is outside G, this counts towards the measure of benefit flowing from G to outsiders: it inflates it, above the level of 'typical' agents. Because there will be many such pairs, in most of the samples, the overall benefit flowing to outsiders, across samples, is substantially inflated.

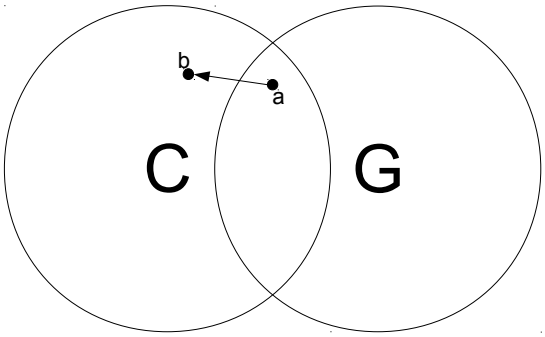

Figure 6.18: The sampling process.

When it comes time to characterize D, the situation illustrated in Figure 6.19 arises. Because C was separated from D by the clusterer, all of the colluding agents (all pairs $\{a, b\}$) are entirely outside of D. When we calculate the benefit flowing from D to outsiders, then, it is that of non-colluding, 'typical' agents. However, because the sampling average was so inflated, D falls far below the average over the samples G. D looks like it is benefitting outsiders far less than a 'random' coalition would, and hence, is wrongly classified as a coalition.

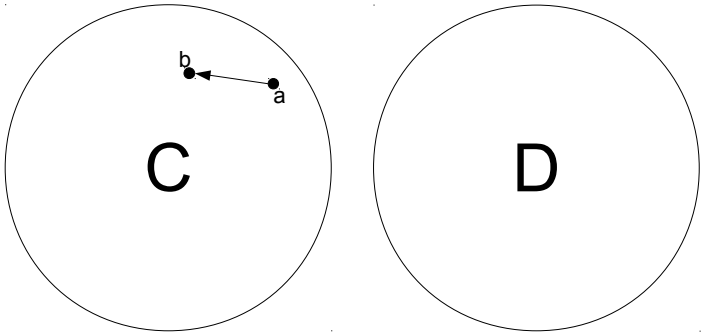

Figure 6.19: Characterization.

6.4.2 The solution: an improved characterization algorithm

Seen this way, the solution to the problem is fairly direct: clusters should be characterized compared to 'typical' agents, not to colluders. As such, we simply exclude suspected coalitions when taking samples for characterization.

This creates a chicken-or-egg problem: how can we exclude suspected coalitions from characterization, when we need to characterize a cluster to determine if it is a coalition? Our solution relies on two properties:

1. Large clusters are more likely to be impacted by this problem than small clusters, because larger samples are more likely to contain significant numbers of coalition members. Thus, we characterize clusters from the smallest to the largest, adding each detected coalition to the 'exclude list' as we proceed.

2. While larger clusters *tend* to be affected more than smaller ones, it is still often true that a smaller cluster is impacted by a larger coalition. However, we note that the sampling problem causes non-coalitions to be classified as coalitions (false positives); there is no corresponding phenomenon that causes coalitions to be mistakenly classified as non-coalitions (false negatives). Put another way, removing agents from the sample 'pool' may result in classifications switching from positive to negative, but usually does not result in switches from negative to positive. Thus, making a second pass over the list, with the exclusion list generated in the first pass (and removing clusters from the list that no longer 'fail' the characterization test),

is generally sufficient to eliminate false positives.[9]

We incorporate this improved characterization method into our multi-clustering algorithm (6.4), resulting in Algorithm 6.5. (Changes from Algorithm 6.4 are shown in boldface.)

Reduction in false positives

We return to the case where we encounter extreme levels of false positives using Algorithm 6.4 (Figure 6.17), but this time applying the improved algorithm (6.5). The results are shown in Figure 6.20. False positives have been nearly eliminated.

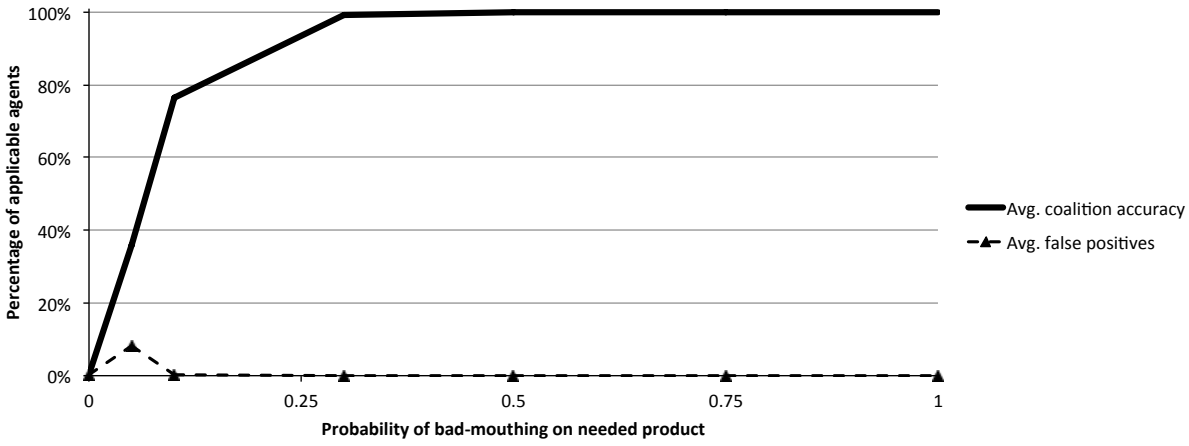

Figure 6.20: Bad-mouthing: performance of multi-clustering algorithm, with improved characterization algorithm.

6.4.3 Addressing difficult cases

In Section 5.5, a number of circumstances were noted to be difficult for the (initial) detection algorithm (5.1). Earlier in this section, the issue of increasing population sizes was addressed. Now, with this improved algorithm (6.5) in hand, we return to the other cases. First, we consider the issue of varying rates of collusion.

[9] One might, for certainty, continue to make passes over the clusters until a static state is reached, but: a) this is likely unnecessary—in most cases, two passes should be sufficient; b) we offer no guarantees that this approach would terminate.

Algorithm 6.5 Multi-clustering Coalition Detection and Identification with Improved Characterization

1. Map agents to points in benefit space.
2. For each clustering method M:
   (a) **Clustering:** For each number of clusters k to be tried:
      i. Apply clustering method M (for k clusters) to the input data set, obtaining clustering result R.
      ii. **Initialize set of suspected coalitions $C_{suspect}$ to empty**
      iii. **Repeat twice:**
         - **Characterization:** For each cluster C (from smallest to largest) in R: Characterize C as a coalition if its benefit to members is unusually high, or its benefit to outsiders is unusually low, as follows:
           A. **If C is in $C_{suspect}$, remove it**
           B. Repeatedly select random sets of agents G (of size $|C|$) from **(population $\setminus C_{suspect}$)** , and for each G, calculate $\bar{\beta}_G$ (the average benefit from G to its own members) and $\bar{\beta}_{\bar{G}}$ (the average benefit from G to outsiders).
           C. If $\bar{\beta}_C$ is unusually high (given the distribution from the random samples G), or if $\bar{\beta}_{\bar{C}}$ is unusually low (given the distribution of from the random samples G):
              – Classify C as a coalition, and all members as coalition members; **add C to $C_{suspect}$**

              Otherwise:
              – Classify C as a non-coalition, and all members as non-coalition-members.
         - If the number of colluders detected for R is the largest yet encountered, store the resulting coalitions as the best set R_{best}.
3. Return the best set of coalitions obtained, R_{best}.

Varying rates of collusive activity

Applying the new algorithm (6.5) to the test in which bad-mouthing rates (Figures 5.17 and 5.19) and ballot-stuffing rates (Figures 5.18 and 5.20) vary per agent, yields the results shown in Figures 6.21 and 6.22, respectively. Again, trials for clusters of size 50 and 100 have been combined into single series, for clarity; the performance of the original algorithm (5.1) has been included for comparison.

In both cases, the multi-clustering algorithm provides substantially better coalition detection, with no increase in false positives.

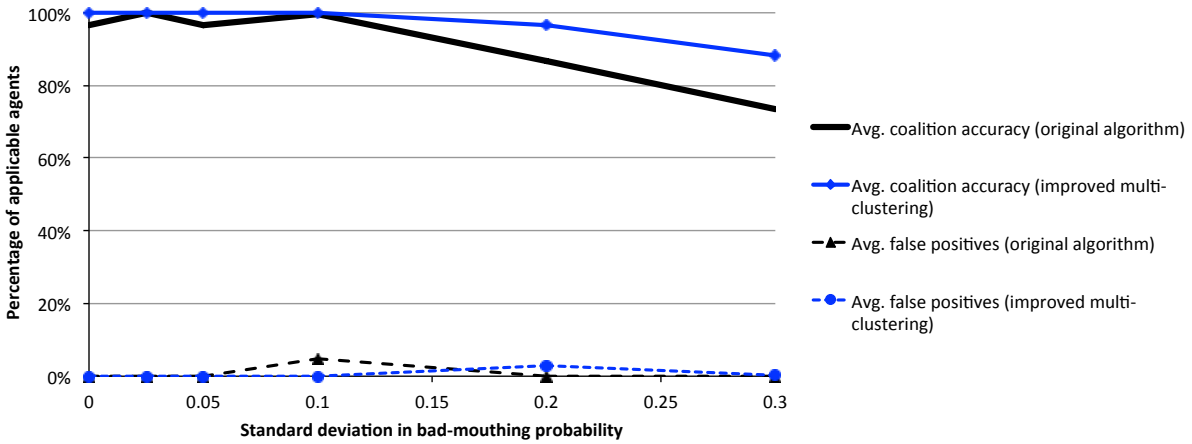

Figure 6.21: Bad-mouthing: Performance with random probability of collusive behaviour.

Low levels of collusive activity

As we learned in Figures 5.19 and 5.20, however, detection performance seems to relate more to individual agents' collusion rates, than to the variability parameter for a given trial. Figures 6.23 and 6.24 show detection performance broken down on the x-axis by the collusion rates of individual agents. (As in Chapter 5, for display, collusion rates were discretized, with bins of 0.05 in width.) In the bad-mouthing case (Figure 6.23), the threshold has moved dramatically to the left: the multi-clustering algorithm is successfully detecting agents at much lower levels of collusive activity. In the ballot-stuffing case (Figure 6.24), we see a slight shift to the left, as well as much stronger performance across the entire range of ballot-stuffing activity.

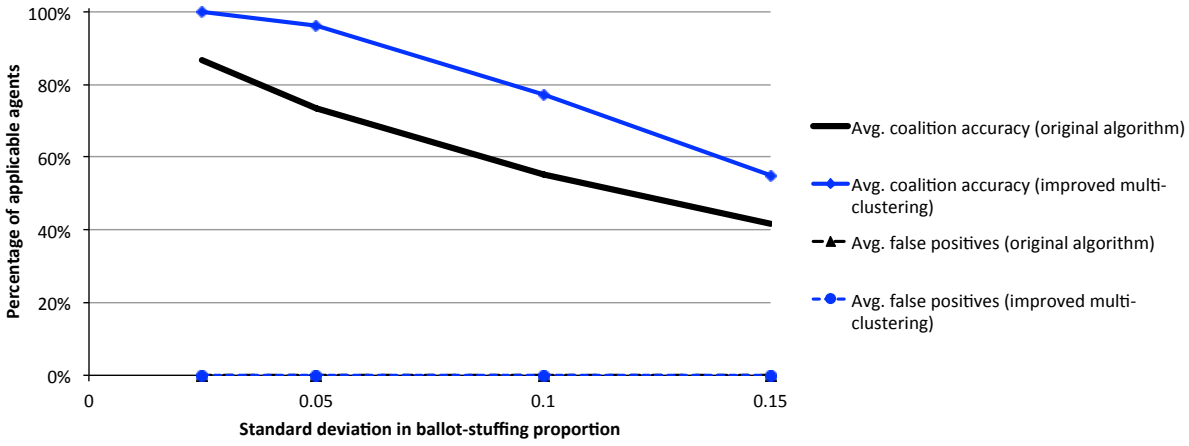

Figure 6.22: Ballot-stuffing: Performance with random proportion of collusive behaviour.

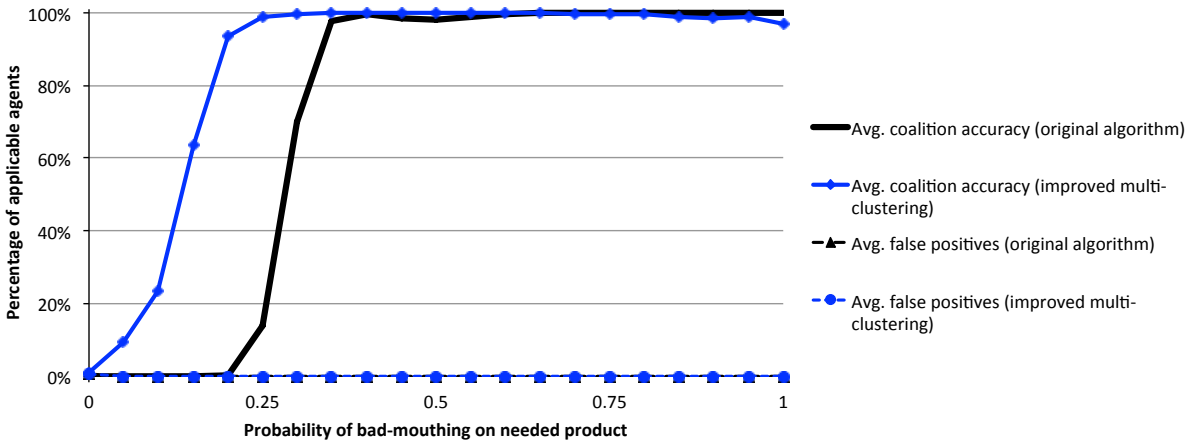

Figure 6.23: Bad-mouthing: Performance with random collusive probability, by agent's individual probability.

Small coalitions

Having improved the algorithm's performance on other cases noted in Section 5.5 to be challenging, we turn to the remaining issue: small coalitions. Figures 5.7 and 5.9 presented the coalition detection accuracy of the original algorithm (5.1) for coalitions as small as 25 members. Performance was quite strong against bad-mouthing, but there was a marked deterioration in ballot-stuffing detection for smaller coalitions. Now, we

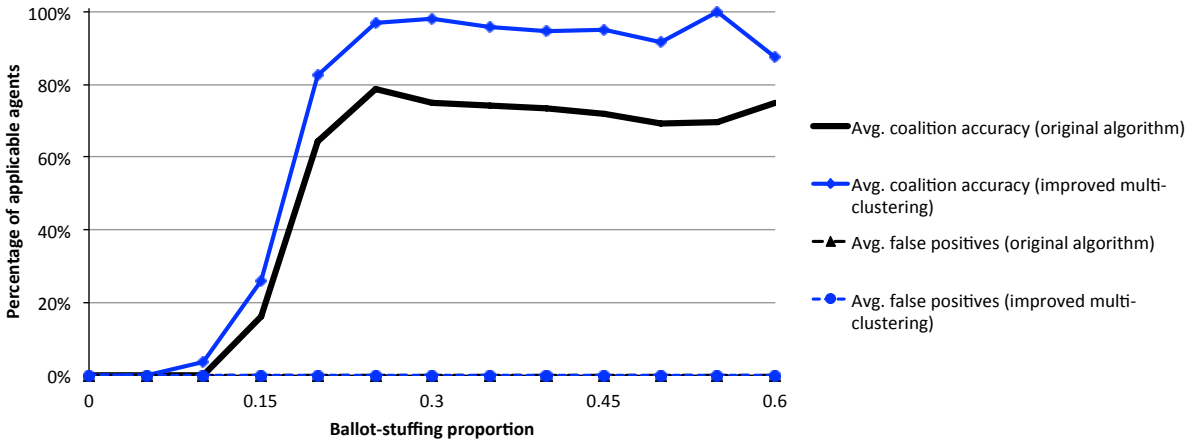

Figure 6.24: Ballot-stuffing: Performance with random collusive proportion, by agent's individual probability.

explore the issue further, considering coalitions ranging in size from five to 25 members, and applying the improved multi-clustering algorithm (6.5).

Bad-mouthing: While Algorithm 5.1 performed well for ballot-stuffing coalitions as small as 25 members, Figure 6.25 shows that detection performance does, indeed, decline for the smallest of groups. (We present only the coalition detection accuracy results throughout this section, for brevity; false positive rates were vanishingly small, in all cases.)

In comparison, Figure 6.26 shows excellent performance, even for the smallest of coalitions.

Ballot-stuffing: The improvement is even more dramatic in the base of ballot-stuffing. Figure 6.27 shows the limited performance of the Algorithm 5.1; in contrast, Figure 6.28 achieves complete accuracy.

The improvements made in this chapter appear to have remedied the weakness of the algorithm to small coalitions.

The improved multi-clustering algorithm (6.5), which demonstrates strong performance even in difficult circumstances, is used throughout the remainder of this docu-

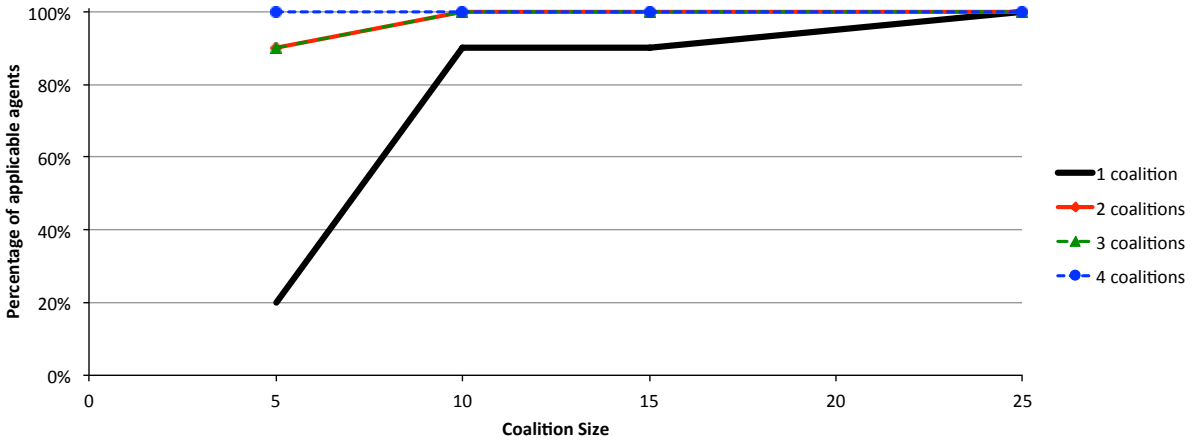

Figure 6.25: **Original algorithm**: Bad-mouthing detection performance against small coalitions.

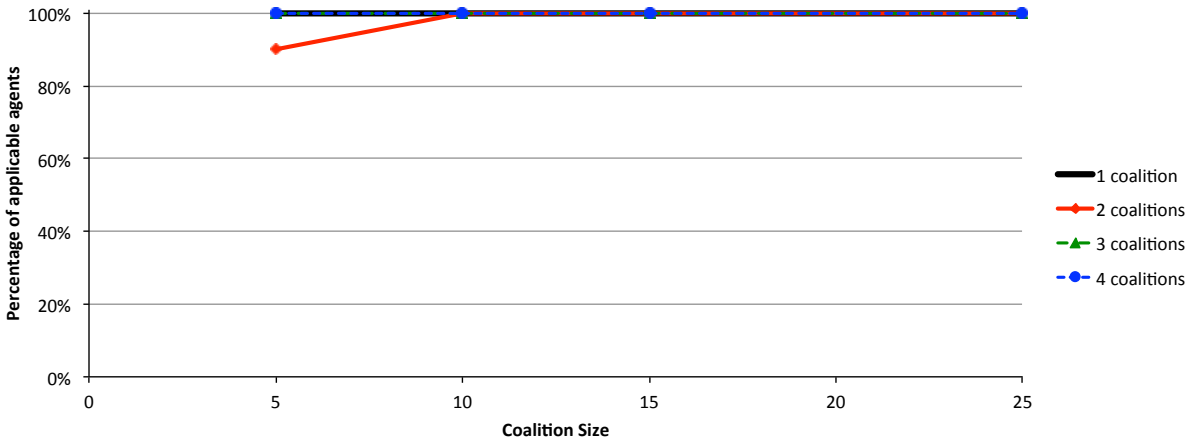

Figure 6.26: **Improved multi-clustering algorithm**: Bad-mouthing detection performance against small coalitions.

ment, unless otherwise noted.

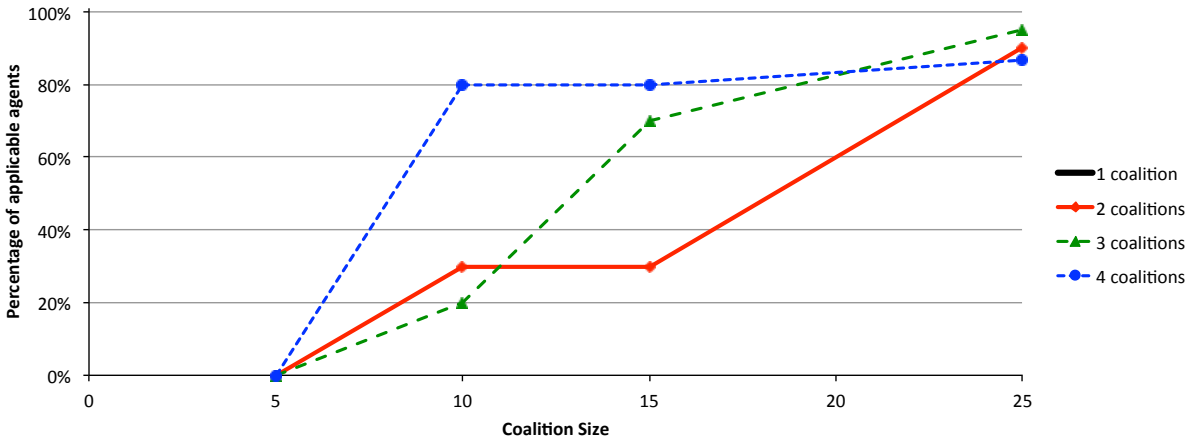

Figure 6.27: **Original algorithm**: Ballot-stuffing detection performance against small coalitions.

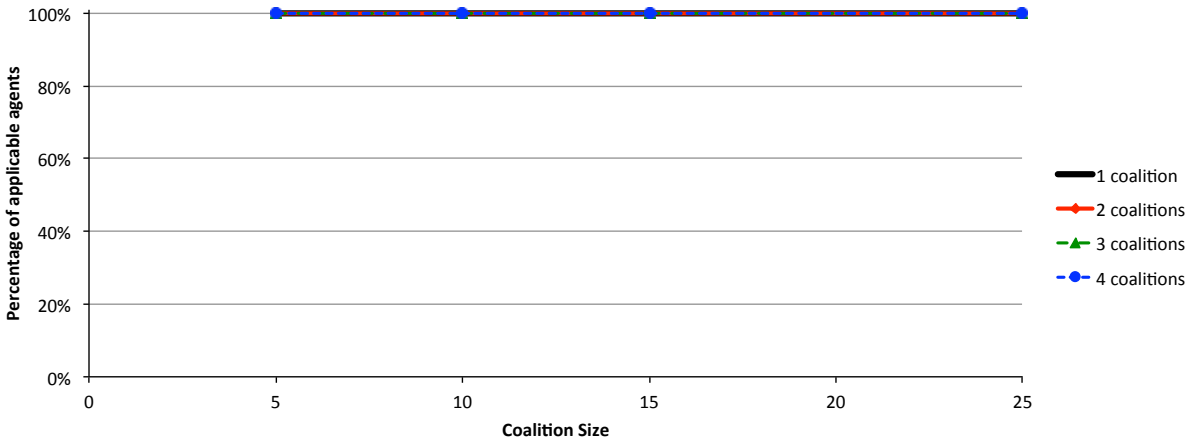

Figure 6.28: **Improved multi-clustering algorithm**: Ballot-stuffing detection performance against small coalitions.

6.4.4 From here

While the refinements presented in this chapter have significantly improved performance, and addressed the cases found challenging by our original algorithm, opportunities still exist for further gains. In the next chapter, we introduce the time dimension to our analysis. This allows us to explore other aspects of coalition detection, and to pursue further

enhancements.

Chapter 7

The Time Dimension, and Dynamic Coalitions

To this point, all of our analysis has been done at one 'snap-shot' in time—specifically, we have compiled all data during each simulation run, and analyzed it after the 1000^{th} round. A number of questions, and possibilities, arise when we consider the time element of marketplace operation. We explore several such issues here.

7.1 Speed of convergence

Just as it may be useful to detect coalitions, it may also be useful to do so *quickly*. For example, if we are hoping to detect colluders in a marketplace, the sooner we find them, the sooner we might take corrective action. Thus far, our methods have shown a strong accuracy in detecting coalitions; here we consider how quickly (or, with how little information) it can achieve that accuracy.

To explore this issue, we conducted experiments where benefit data was aggregated round-by-round, and detection was performed at the end of each round. Figure 7.1 presents the performance of the improved multi-clustering algorithm (6.5) over an example set of trials, in each of which a single bad-mouthing coalition of size 100 was present in a population of 1000. As in previous tests, the coalition members only bad-mouthed when purchasing products for legitimate needs. Round number is mapped to the x-axis—the chart depicts performance as time progresses. (Because we are interested

in the speed with which detection performance reaches its typical long-term level, only the initial rounds of the trials are shown.)

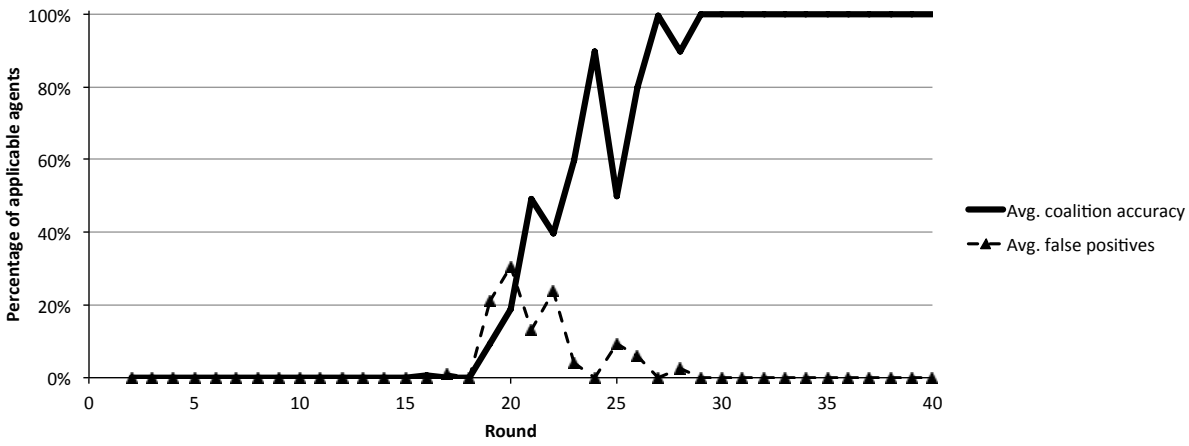

Figure 7.1: Detection accuracy by round, as information is accumulated.

Note that in our simulations, there is a delay (14 rounds) between the time of purchase (i.e., when payment is made), and the time at which the good is received and evaluated (and hence, would turn up in reviews). Thus, no detection is possible until review data first becomes available in round 15 (i.e., for purchases made in round 1).

As can be seen in Figure 7.1, the very first detection occurs in round 19, with only four rounds' worth of data. Performance is limited with so little data, however: there are some false positives present in the first few rounds of detection, and coalition detection accuracy is low. By round 27, however (only 12 rounds after data first started to be accumulated), classification accuracy has converged to customary levels. This is encouraging, indicating that classifications can be made quite quickly.

7.2 Dynamic Coalitions

As we consider the operation of a marketplace over time, it is natural to consider the behaviour of coalitions over time as well. As we seek to identify coalition membership, perhaps the most interesting question might be, what if coalition membership changes?

Indeed, depending on the nature of the coalition, changes in membership may be quite likely. A strongly coordinated coalition (in the extreme case, a set of accounts

all controlled by one entity) might be entirely stable; in contrast, a casual group of self-interested individuals, who collaborate only out of convenience, may be subject to frequent change. Agents may join coalitions, switch between coalitions, or stop collaborating and continue as individuals.

To determine current memberships at any given moment, one might restrict analysis to very recent information, discounting older (and possibly obsolete) data. As we have seen in the previous section, however, some quantity of information is required before reasonably accurate classifications can be made—relying only on the most recent data may prove inadequate.

A solution to this problem, which has been used frequently in the area of trust and reputation, is the use of a *forgetting factor* (e.g., [29, 52, 86, 88]). Essentially, the weight of each review is reduced with passage of time; historical information is incorporated into analysis, but more recent information is favoured.

We follow the form of forgetting factor introduced by Jøsang [29]. In our implementation, at any given moment a benefit score b is used in analysis, where b is a weighted sum of the most recent information, and accumulated historical information. Specifically, for every source agent a, and every target agent i, the benefit score b used in time step n is

$$b_{i,n}(a) = w_c \beta_{i,n}(a) + w_p b_{i,n-1}(a) \tag{7.1}$$

(recalling that $\beta_{i,n}(a)$ is the amount of benefit flowing from agent a to agent i, here in time step n) where w_c and w_p are the weights applied to current information and past information respectively. Where $w_p < 1$, each act of benefit will decrease in importance with each additional time step. The benefit scores b are the inputs to the detection algorithm, rather than the raw benefit values β.

7.2.1 Results

We apply this approach to a scenario where the membership of ballot-stuffing coalitions changes as time progresses, as shown in Figure 7.2. In this test, there are two coalitions of size 100 present. In addition, there are 100 'inactive' colluders present as well—agents that do not currently belong to a coalition, but may join one at a later time—for a total of 300 potential colluders, in a population of 1000. For ease of interpretation, colluders change memberships at two points during the simulation run: in round 400, and round 800. At that time, 50 members leave each coalition, to be replaced by 50 new members. A member departing a coalition may join the other coalition, or may

become inactive; similarly, a member joining a coalition may have come from the other coalition, or may have come from the pool of inactives. In this test, we consider it correct to classify currently-inactive members as non-colluders, because they have no coalition membership at the present time.

For efficiency, we aggregate benefit (which we refer to as *polling*) every tenth round, rather than every round.[1] The improved multi-clustering algorithm (6.5) was used, and weights of $w_c = 1.0$ (current) and $w_p = 0.8$ (past) were used. (This means that data would carry a weight of 1 in the period in which it occurs, 0.8 one period later, $0.8^2 = 0.64$ two periods after it occured, and so on.)

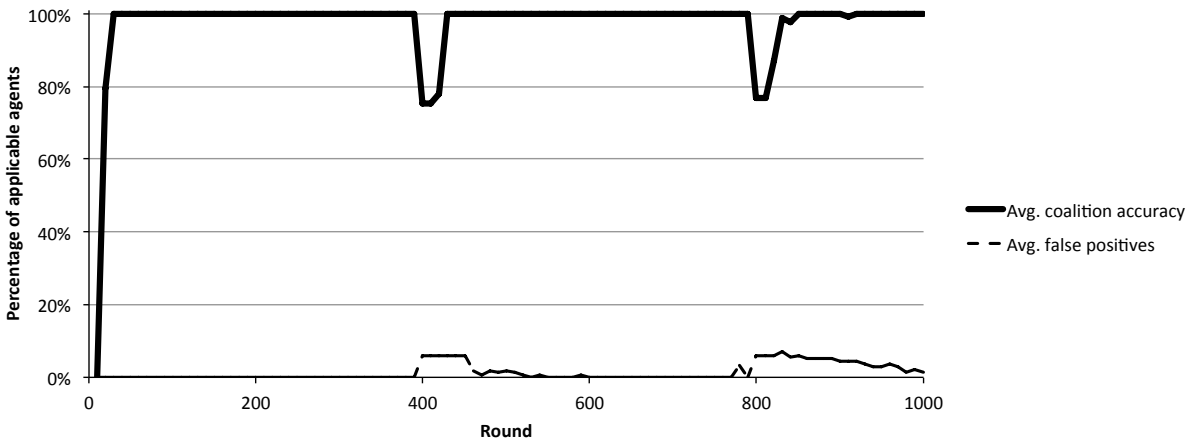

Figure 7.2: Detection accuracy, dynamic coalitions.

As can be seen in Figure 7.2, there are sudden drops in detection accuracy, and increases in false positives, immediately following the changes in coalition membership in rounds 400 and 800. This is expected—an agent may have left a coalition, for example, but he will need to engage in some transactions before this new strategy is visible to others. Note, however, that detection accuracy jumps back to near-optimal levels very quickly after each change in coalition memberships. False positives return to very low levels as well, albeit not as quickly as detection accuracy is restored.

[1]Here, we do not refer to the efficiency of the algorithm itself, but rather, the efficiency of our experimental setup. Our testbed is designed to write simulation data to file (rather than having it analyzed online during simulation operation); this is necessary, if we are to be able to subsequently analyze the data with a variety of algorithms. Polling requires (large) files capturing all transaction data to be written at every polling interval; analysis subsequently requires every such file to be read in its entirely. Thus, polling (particularly a high polling frequency) dramatically increases I/O time, in a way that need not occur in practice.

As explained in Section 6.1, detection accuracy and false positives address whether or not an agent has individually been classified as a colluder, but do not consider the grouping/separation of agents. Here, then, these measures speak primarily to the issue of whether an agent has stopped or started colluding.

Of course, an important aspect of dynamic coalitions is that members also move from one coalition to another. To determine our degree of success in detecting these changes in membership, we look again to purity. Figure 7.3 depicts the purity results for the analysis shown in Figure 7.2.

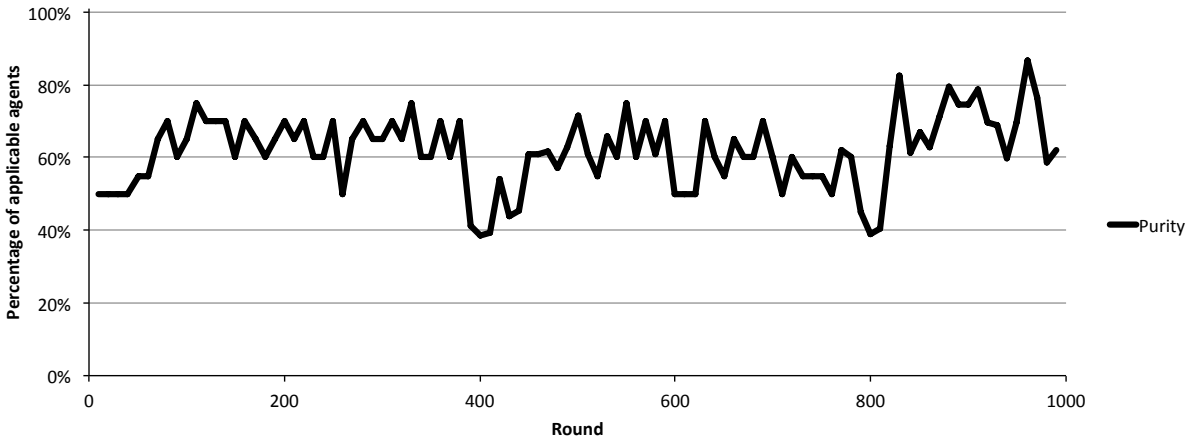

Figure 7.3: Purity, dynamic coalitions.

Two points are evident in this figure. First, there is a drop in purity after rounds 400, and 800, as coalition membership is shuffled; purity subsequently increases as the detection algorithm makes use of new information. Second, the purity scores are not particularly strong. (Consider that if we placed all 300 colluders (two active coalitions of 100, and 100 inactive agents) into the same coalition, the result would be a purity of $100/300 \approx 0.33$.) For this reason, the pattern noted in the first point is much subtler than for detection accuracy.

These purity results are not unexpected, however. As noted in Section 6.1, the algorithm (without further refinement) has limited success in separating coalitions from one another; this was the very inspiration for recursive refinement. We applied recursive refinement (Algorithm 6.1), then, to our analysis, yielding the purity results in Figure 7.4.

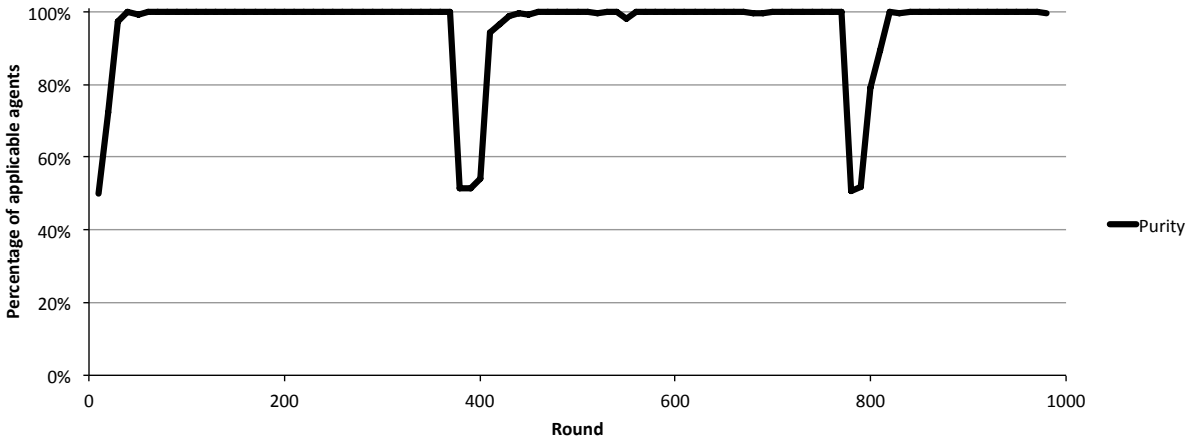

Figure 7.4: Purity, with recursive refinement, for dynamic coalitions.

Here, we see very strong performance. Purity drops each time members move between coalitions, but is restored very quickly to near-optimal levels. With the application of recursive refinement, our method separates members of coalitions well, and adapts swiftly to changes in coalition membership.

Nevertheless, as noted in Section 6.1, purity only tells part of the story: whether members of different coalitions have been separated. Rand index sheds light on the issue of whether members of the same coalition have been grouped together (an important concern, albeit one that is not as critical as purity). Figure 7.5 shows the RI results after the application of recursive refinement.

These results, taken in combination with those for purity, indicate that members of different coalitions are separated into different clusters in almost all cases, and in the majority of cases, members of the same coalition are placed into the same cluster.

7.3 A new feature set: TF-IDF

Introducing the time dimension provides the opportunity to deepen our analysis of agent behaviour. Most of the preceding chapters have focused on techniques to analyze benefit data; now we revisit the question, how can we measure benefit? Of course, we are still limited to the same set of observable actions as before, but the timing of such actions provides additional information: actions may be more or less beneficial, depending on

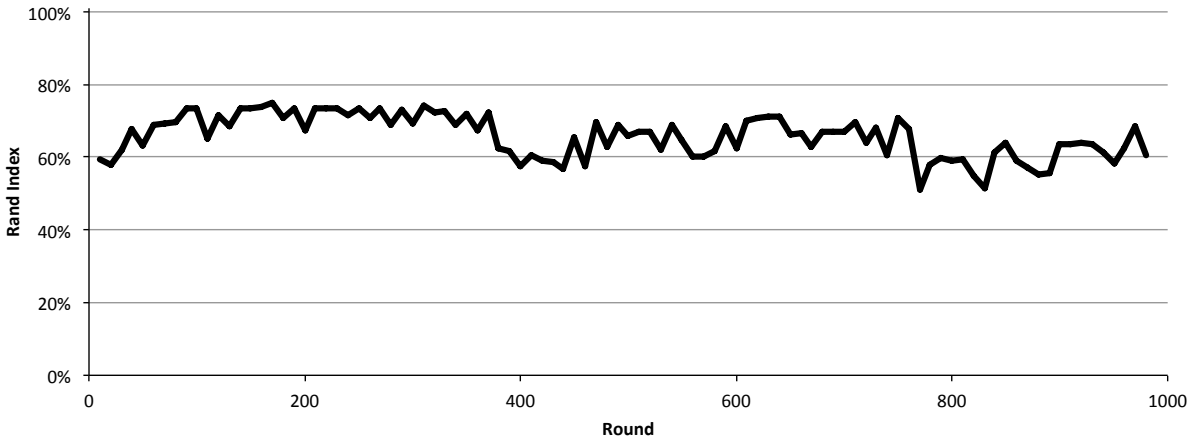

Figure 7.5: Rand index, with recursive refinement, for dynamic coalitions.

their timing, and the situation at the time they occur.

For example, consider two agents a and b, each attempting to make sales. Further, consider that a has made very few sales to date (say, five, all with positive reviews), and hence has very little reputation. In contrast, b has made 1000 sales (each with a positive review). Now, consider that these agents are the beneficiaries of ballot-stuffing. In most reputation systems, the incremental benefit of one additional positive review will be higher for a (who has very few such reviews) than for b. Note too, that a (ballot-stuffing) purchase probably constitutes a large portion of a's sales for that period—because he has little reputation, he wins few sales. An additional purchase from b, however, is probably just one of many such sales. Thus, the sale is of much more benefit to a, and it is also more *unusual*.

At a later moment in time, after a has won many sales and reached a level of success comparable to b, an additional (ballot-stuffing) sale will no longer be so unusual—nor so beneficial.

7.3.1 TF-IDF

The notion of *unusualness* is reminiscent of the TF-IDF (term frequency-inverse document frequency) measure [64] well known in information retrieval. When considering a document within a corpus, TF-IDF seeks to determine the importance of a given word to that document (for example, to attempt to characterize the content of that document or

to identify key topics within it.) The 'TF' portion, term frequency, refers to the number of times that the word in question appears in the document[2]—the reasonable presumption is made that the more frequently a word appears in a document, the more representative it might be of that document's content. For example, if the word 'weed' appeared once in a document, it might have little to do with that document's theme: for instance, the author may have used the expression 'grow like a weed'. In contrast, if 'weed' appeared 15 times, it would be much more likely to indicate that the document is weed-related: invasive weeds, weed control, etc.

Counter to this, is the fact that some words occur very frequently, but carry little information about the content of a document: e.g., 'the', 'go', 'one', etc. Thus, the importance of term frequency is reduced for words that have high 'document frequency', meaning they occur frequently in the corpus. IDF, the inverse document frequency is the (inverse) proportion of documents in the corpus that contain the word in question.[3]

TF-IDF, the product of term frequency and inverse document frequency, then, provides a score indicative of how important or noteworthy a word is to a document.

7.3.2 TF-IDF and coalition detection

The concepts of TF-IDF also apply to our problem. We are trying to characterize the behaviour of agents, similar to how TF-IDF can be used to characterize the content of documents. Until now, we have considered the quantity of beneficial actions occurring between agents; this represents the the 'TF' portion—but only to a point. As explained in Section 5.2.3, we have used 'net benefit' as our measure: benefit minus harm. From a TF-IDF perspective, beneficial actions and harmful actions need to be considered separately because, for example, positive reviews of a given target agent may be common, but negative ones might be unusual. Further, to know if a's negative review of b is unusual, we need to know what constitutes 'unusual': we have not considered the 'IDF' issue at all.

On this basis, we developed a benefit measure similar to TF-IDF. Where TF-IDF uses counts of words/documents, however, we do not use the most direct analogs (e.g., counts of actions/agents), for two reasons. First, as we noted earlier, domain expertise is required to identify those actions that are truly beneficial in a given scenario. In a given

[2]In fact, term frequency is normalized for document length, so it is more accurate to say 'the number of times the word appears in the document, as a proportion of the total words'.

[3]In fact, the logarithm of IDF is used, so that reasonably frequently occurring words (e.g., 'five') aren't given substantially higher weights than *very* common words (e.g., 'the').

domain, counts of actions might be the best measure (e.g., number of reviews), or actions might have different weights (e.g., review score, or dollar value of transaction). Second, in a busy, well-connected marketplace, it may be the case that most agents have interacted with one another: counts of the number of agents are of little utility. Thus, we have defined our formula such that any measure of benefit β can be used.

$b_{i,n}(a)$ denotes the benefit score used in analysis, at time step n, for the benefit flowing from agent a to agent i. In each time step, the new benefit score is calculated by adding an update, Δb, to the benefit score from the last time step:

$$b_{i,n}(a) = b_{i,n-1}(a) + \Delta b_{i,n}(a) \tag{7.2}$$

Implicit, here, is the fact that we now allow multiple dimensions per agent. For example, in our application, there were two separate dimensions for each target agent: benefit and harm. Formally, the b values should also be subscripted by the number of dimensions being tracked per agent pair, but we have omitted this for clarity of notation.

The calculation of the updates Δb is where the TF-IDF-like computation occurs:

$$\Delta b_{i,n}(a) = \beta_{i,n}(a) * \log \frac{\sum_{j \in P, k \in P} |\beta_k(j)|}{\sum_{j \in P} \beta_i(j)} \tag{7.3}$$

Essentially, the 'TF' component is the amount of benefit (or harm) flowing from a to i in the most recent time period, $\beta_{i,n}(a)$. The 'DF' component is the amount of benefit (or harm) flowing from all agents in population P to agent i, as a proportion of the total benefit amongst all agents in the system. (DF includes all recorded transactions, not just those from the most recent time period. Note that acts of benefit are still likely to have positive values, and acts of harm are likely to have negative values. It is for this reason that we take the absolute value when summing the grand total of all actions.) The TF portion represents the amount of benefit/harm flowing from a to i; multiplying it by the inverse DF reduces the weight of the TF portion, to the degree that similar actions towards i have been common in the past.[4]

7.3.3 Results

We tested this measure to determine if it could improve detection performance, beyond the 'raw' benefit scores used previously. In previous tests, enhancements had pushed

[4]Again, a forgetting factor can be applied to b values and β values from past time steps, in order to favour recent activity.

many of our benchmarks to the point where there was little opportunity to make (or at least, little opportunity to discern) substantial performance improvements. Thus, we have developed two new test scenarios that combine several complicating factors.

In these tests, there are two small coalitions present (of size 25). Agents cheat with varying probability by individual (mean probability 0.25, with standard deviation of 0.2), and also vary individually in their overall buying rates (mean of 4, standard deviation of 1). Collusion rates are also variable by individual (bad-mouthing: mean of 0.3, standard deviation of {0.025, 0.05, 0.1} on different trials; ballot-stuffing: mean of 0.2, standard deviation of {0.05, 0.1, 0.15} on different trials).

We made use of the Algorithm 6.5; the results using our 'TF-IDF' measure are compared to those using 'raw' benefit. Figure 7.6 depicts performance in the bad-mouthing case.

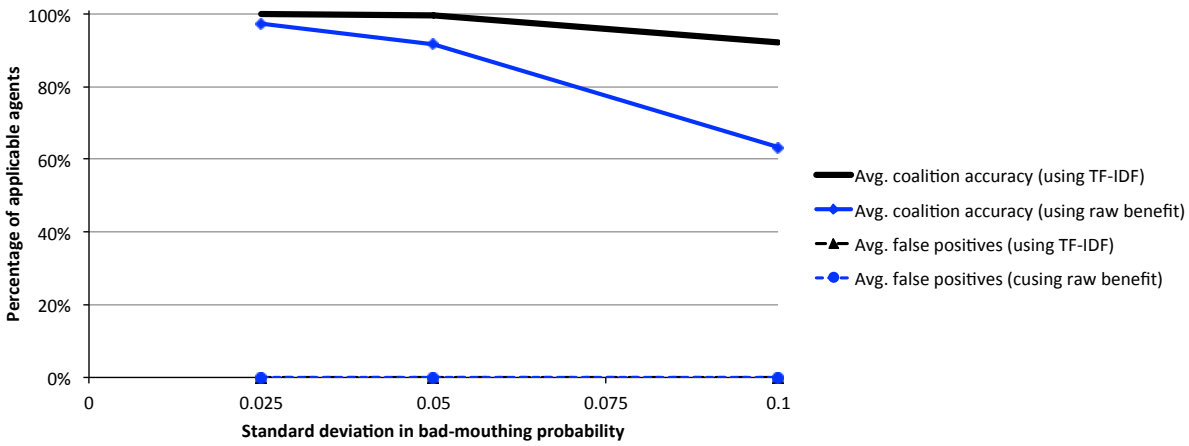

Figure 7.6: Bad-mouthing: performance of 'TF-IDF' vs. 'raw' benefit.

There is a clear, substantial improvement in detection performance.

As we have done previously, we show the same results, but broken down by individual bad-mouthing rate, in Figure 7.7. The threshold has moved substantially to the left using TF-IDF, meaning that colluders are being detected at much lower individual collusion rates.

The results against ballot-stuffing are shown in Figure 7.8. Here, the performance using raw benefit exceeds that using TF-IDF. Obviously, this is not the ideal result, but there is still interesting potential here, as discussed below.

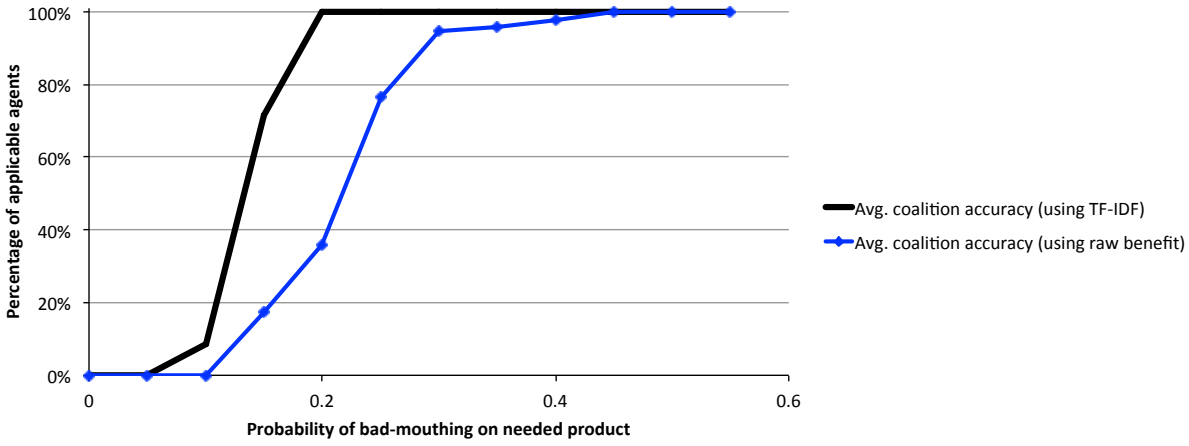

Figure 7.7: Bad-mouthing: performance of 'TF-IDF' vs. 'raw' benefit, by individual bad-mouthing rate

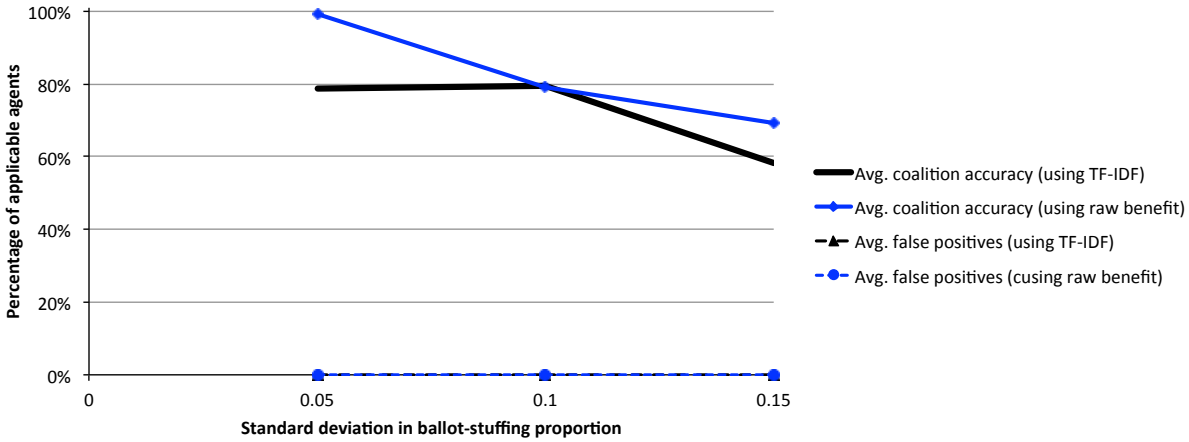

Figure 7.8: Ballot-stuffing: performance of 'TF-IDF' vs. 'raw' benefit.

Multi-measure detection

As explained in Section 6.3.2, we are able to use multiple clustering methods in parallel: because our characterization method is resistant to false positives, we can maximize detection by choosing the result with the largest number of detected colluders.

Looking at Figure 7.8, we can see that the same situation holds: although TF-IDF did

not provide improvements in detection, it did not introduce significant false-positives, either. The fact that TF-IDF does improve detection in some situations, without appearing to introduce false positives, suggests that it may be useful to combine TF-IDF with other measures. In addition to using multiple clustering algorithms in parallel (as in Algorithm 6.5), we might also use multiple benefit measures in parallel. In this way, we might obtain improved performance in many situations (like Figure 7.6), but without detrimental impact in others. Exploration of this is future work.

7.4 Discussion

Consideration of the time dimension both introduces new issues, and provides new opportunities for increasingly sophisticated detection methods.

Beyond the issues discussed above, two other conclusions might be drawn from our results using the 'TF-IDF' measure.

First, it seems evident that improved measures of benefit have the potential to improve performance. The importance of developing such measures, and of domain expertise in identifying them, seems clear.

Second, these results hold implications for the application of our work to other domains. While derived from the same data, raw benefit and TF-IDF are entirely different measures. That both would provide strong results validates the ability of our method to handle a variety of data representations. This suggests promise for applying our techniques in other domains: our methods may, in fact, cope with data generated in different scenarios, just as they have done for very different measures generated from the same scenario.

Chapter 8

Applying Coalition Detection: A Collusion-Resistant Reputation System

We now return to the original problem that inspired our investigation of coalition detection: the vulnerability of trust and reputation systems (TRSes) to collusive attacks. As demonstrated in Chapter 4, there is widespread vulnerability amongst TRSes to bad-mouthing and ballot-stuffing attacks: coalitions employing these tactics are much more profitable than non-colluding agents, meaning that there is an incentive to engage in such activity.

In this chapter, we present a reputation system that is resistant to such collusive attacks. To develop such a system, one might construct an entirely new mathematical model; the principles of our work (such as the importance of similarity, and of mutually beneficial benefit flows) might be the fundamental basis of such a system. A likely goal would be for the system to be 'collusion proof', in the sense that it could be proven that collusion was *less* profitable than independence, so that there would be a disincentive for collaboration. (We discuss issues around the development of such a system later in this chapter.) Here, however, we take a different approach: incorporation of our coalition detection method into an existing TRS, to make it resistant to collusion. We do so to illustrate one potential usage of the tool we have developed, and specifically, to show how our generally-applicable tool (coalition detection) can be applied to domain-specific problems. Our focus here is on electronic marketplaces.

8.1 Collusion-Resistant Beta Reputation System

We make use of the Beta Reputation System (BRS) [29] (as we have in preceding chapters), because it is frequently cited and well studied. Further, its clear and direct formulation make it quite compatible with the incorporation of a system such as our detection algorithm.[1]

As noted in Chapter 4, BRS in its original form is vulnerable to ballot-stuffing and bad-mouthing.

In BRS, agents make use of their own experience in evaluating the trustworthiness of others, but also make use of advice provided by other agents. In presenting BRS, the authors note that advice received from other agents may not be reliable. For example, when agent X requests advice from agent Y about target agent T, Y's review may not reflect T's true character. A system for coping with this issue is presented, *reputation discounting*, where X's faith in Y's review of T is dependent on X's experience of Y's trustworthiness. [29]

The system we propose here is consistent with this reputation discounting in principle, although the mechanism is different. Reputation discounting suggests that we should place less faith in information provided by agents if we don't know them to be trustworthy. Here, we take the position that if we believe agents to be colluders, we cannot trust them at all. We propose that, beyond each agent relying on its own experience of those agents providing reviews, it also take into account the findings of the coalition detection algorithm (here, the improved multi-clustering algorithm, 6.5). Specifically, at each desired interval (e.g., once per day), the detection algorithm is run on all reviews that have been registered in the past. If coalitions are detected, reviews likely to be resulting from collusive behaviour are discounted. Specifically:

- If a coalition is detected, we cannot trust the reviews that suspected members have *given* in the past, because each such review may be a collusive action.[2] We cannot determine with certainly exactly how long the collusive activity has been occurring,

[1] As we have done throughout this work, we have assumed complete connectivity—that all agents receive reviews from all other agents. Equivalently, for our purposes, we could assume that a central repository holds all reviews.

[2] While a detected coalition is a group that is treating its members substantially better than it is treating outsiders, we have made no guarantee that such a detected coalition is necessarily malicious. Despite this, because we have no means of inferring intention, we conservatively treat all detected coalitions as suspicious.

nor exactly who might be the beneficiaries or victims, so we disregard all previous reviews given by suspected colluders.

- At the same time, if a coalition is detected, we cannot trust the reviews that suspected members have *received* in the past. Some of the reviews might have been collusive actions, in the form of ballot-stuffing. Others, however, may be legitimate reviews from non-members, but have come on sales gained due to successful collusive activity. As such, these too represent an unfair advantage. Thus, we disregard all previous reviews received by suspected colluders.

We incorporate these policies into BRS, resulting in what we refer to as the Collusion Resistant Beta Reputation System.

These actions are more drastic than reputation discounting—suspected colluders lose all of their reputation, as well as all of the impact of their reviews. This may seem potentially unfair, given the prospect of false positives, but there are numerous factors suggesting these concerns may be misplaced:

- First, as demonstrated, the detection algorithm is quite resistant to false positives.
- Overall, the system provides agents with protection, which makes the marketplace more attractive. Buyers are protected from being drawn into making purchases from colluding agents with potentially malicious motives, while sellers are protected from losing sales to colluders. This provides an incentive for (honest) agents to participate, despite the risk of false positives.
- As will be seen in the experimental results, honest sellers are still very profitable. (Our system does not reduce the overall number of sales, but may simply redistribute those sales as reputations are adjusted.)
- Finally, and perhaps most importantly, fairness is not of utmost importance. It is worthwhile to recall that many TRSes, including BRS, can be implemented as distributed systems in which agents (here, buyers) individually evaluate the trustworthiness of other agents (here, sellers). Such a buyer is primarily concerned with his own protection, not with ensuring equitable treatment of all sellers. Thus, our system is compatible with the goals of those agents who might employ it.

Certainly, other implementations with less drastic corrective actions might be attractive; exploration of such possibilities is potential future work.

8.2 Experimental results

To evaluate this system, we constructed a set of trials similar to those in Section 4.7. Here, we are interested only in the performance of the TRS (as opposed to, for example, exploring the nature of the attack relative to other possible behaviours). Thus, trials are configured as follows:

- All agents in the system (colluders and non-colluders) make use of the TRS in place for the trial (either the original Beta formulation, or the Collusion-Resistant Beta system.)
- All agents engage in both buying and selling.
- There are 500 non-colluding agents present, and a single coalition of size 100.
- For each TRS, 10 trials are run with bad-mouthing agents, and 10 with ballot-stuffers.
- As in Section 4.7, in the bad-mouthing trials, colluders bad-mouth on all needed purchases; in the ballot-stuffing trials, colluders have ballot-stuff proportions of 0.5.
- Other than the collusive actions, agents are otherwise honest—they fulfill sales faithfully.

Because of differences in configuration and agent population, these results are not comparable to those from Section 4.7. We discuss the issue of parameter choice further, following the experimental results.

8.2.1 Bad-mouthing

The performance of the TRSes against bad-mouthing is shown in Table 8.1. As in Chapter 4, the first column of the table represents the profit (per agent) of colluders, relative to non-colluders. We would hope this value would be negative: when colluders earn less than non-colluders, then collusion is unattractive. The second column shows the percentage of trials in which colluders earn more than non-colluders ('failures').

Using Beta without coalition detection, colluders are dramatically more profitable. In contrast, the use of coalition detection not only negates the effects of collusion, but

actually penalizes those who attempt to engage in it. The inclusion of coalition detection makes collusion dramatically less attractive.

Table 8.1: Performance of Beta/Collusion-Resistant Beta, against bad-mouthing.

TRS	**Colluder profit (relative to honest)**	**Trials failed by TRS (% of 10 trials)**
Beta	+429.48%	100%
Collusion-Resistant Beta	-87.03%	0%

Figure 8.1 depicts the operation, during one bad-mouthing trial, of Beta without coalition detection. The colluders quickly gain a sustained advantage, attaining dramatically higher sales and profit levels. In contrast, Figure 8.2 depicts a corresponding trial when Beta is supplemented with coalition detection. Colluders have an early advantage, as the effects of their bad-mouthing are helping them to win sales. Very quickly, however, enough data is accumulated to allow accurate coalition detection—the situation is flipped, with non-colluders dominating the marketplace.

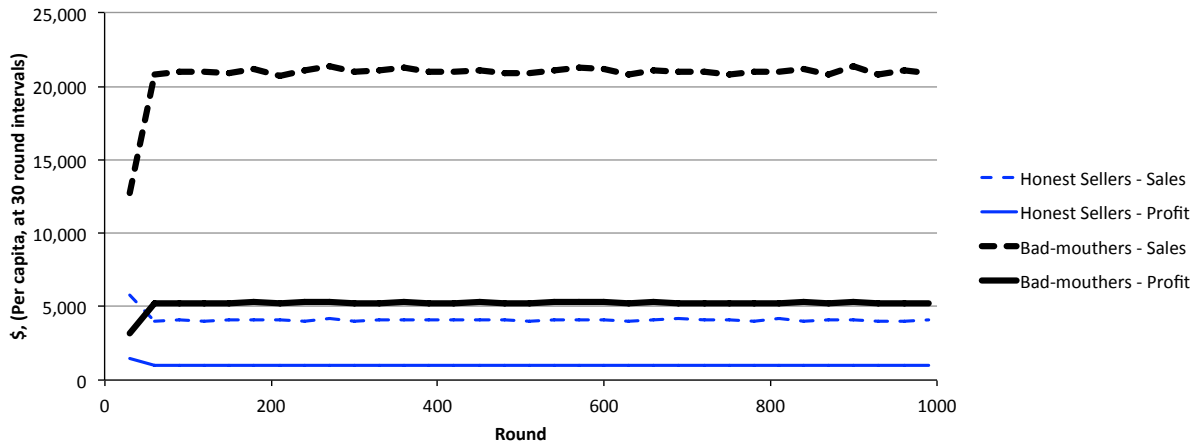

Figure 8.1: Bad-mouthing: performance of BRS without coalition detection.

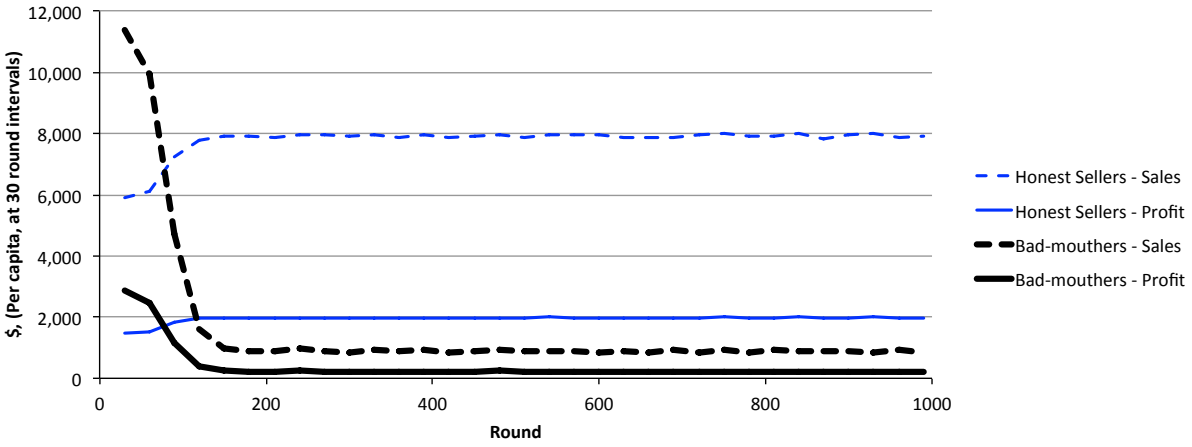

Figure 8.2: Bad-mouthing: performance of BRS with coalition detection.

8.2.2 Ballot-stuffing

Table 8.2 shows the performance, over 10 trials, of each TRS when faced with ballot-stuffing. (Note that unlike our bad-mouthing agents, the ballot-stuffers generate fake purchases while pursuing their strategy—thus, these sales/profit figures include the buying activity as well as the selling.) As with bad-mouthing, when coalition detection is not used, coalition members make more money than non-members. With coalition detection in use, the opposite is true, making collusion unattractive.

Table 8.2: Performance of Beta/Collusion-Resistant Beta, against ballot-stuffing.

TRS	**Colluder profit (relative to honest)**	**Trials failed by TRS (% of 10 trials)**
Beta	+158.92%	100%
Collusion-Resistant Beta	-63.78%	0%

Figure 8.3 portrays the operation of Beta (without coalition detection) during one trial. The situation is fairly similar to that for bad-mouthing: colluders quickly gain an advantage, which is retained throughout operation. Figure 8.4 shows a corresponding trial where coalition detection was incorporated into Beta. As in the bad-mouthing case, the early advantage of the colluders is quickly reversed.

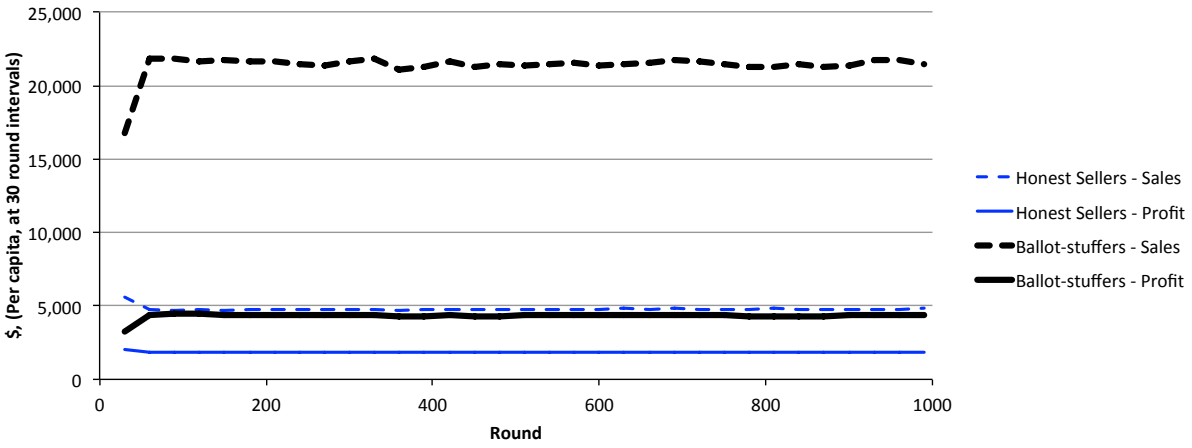

Figure 8.3: Ballot-stuffing: performance of BRS without coalition detection.

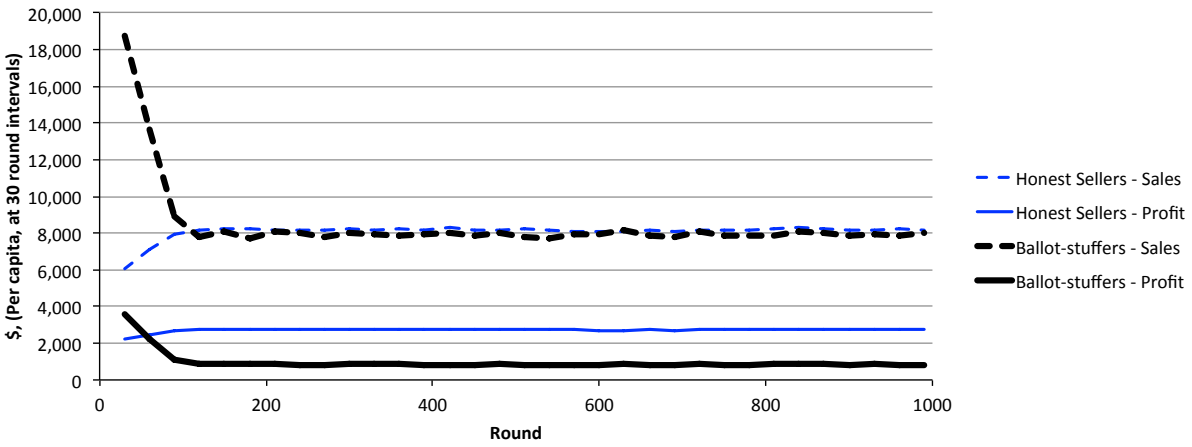

Figure 8.4: Ballot-stuffing: performance of BRS with coalition detection.

The introduction of coalition detection to BRS has provided agents with substantial protection against colluders. Further, agents are actually penalized for collusion—which provides a strong disincentive for such activity.

8.2.3 Experimental parameters

As we have seen in preceding chapters, there is an enormous parameter space: coalition sizes, number of coalitions, collusion rates, variability in collusion rates, cheating rates, variability in collusion rates, etc. We could fill many pages exploring these parameters, but fortunately, this is not necessary. Under any of the circumstances in which our algorithm successfully distinguished between colluders and non-colluders (as explored in earlier chapters), the system here would likewise detect the colluders, and thus be able to counter the effects of the collusive behaviour.

8.3 Discussion

Despite impressive performance, we do not suggest that this system is a perfect, collusion-proof system. Rather, we have presented it as an example. It is intended to illustrate how our work might be used directly, in one of the domains to which it might apply, to address issues posed by coalitions.

Indeed, for a TRS to be truly 'collusion proof', it must be proven that collusion cannot take place (or at least, that collusion is less beneficial than non-collusive behaviour, so that rational agents will choose not to collude). No such system has yet been developed which is applicable for marketplace use. This suggests the difficulty in developing such a system.

Our detection algorithm is unlikely to become the basis for a provably collusion-proof mechanism, because we cannot give hard guarantees of the probability with which colluders might be detected. That said, principles of our work might provide the foundation of such a system. In particular, we have noted that similarity seems to be essential to cooperation. A TRS that, for example, discounts the values of reviews to the degree that the reviewers' opinions are similar, might have promise.

There are a number of requirements that a well-functioning and 'collusion-proof' TRS would need to fulfill. In fact, while we propose no such system here, we have identified certain constraints on the properties of such a system—constraints which restrict possible formulations. We sketch some noteworthy constraints here.

8.3.1 Requirements for a 'collusion-proof' system

An attempt to develop a 'collusion-proof' system might be approached from the perspective of *game theory* and *mechanism design*. Game theory is "the mathematical study of interaction among independent, self-interested agents" [67]. Mechanism design, a field within game theory, is particularly concerned with designing the rules of games so that agents, in seeking to maximize their own rewards, make the choices that are desired by the system designer [67].

In the typical approach, mechanism designers would not seek to prohibit collusion from occurring. Rather, designers would seek to develop marketplace rules under which non-collusive behaviour always results in greater expected reward than collusive behaviour (i.e., that non-collusive strategies are *dominant strategies*), so that rational, self-interested agents would choose not to collude.

This is very challenging for a TRS, in part because there are two distinct, yet interrelated types of activities, and we wish to exert control over both of them. First, there are the actions themselves (here, sales): we want agents who are making sales to fulfill each faithfully, providing the promised good. Second, there are the reviews of those actions: we want agents to provide honest (and non-collusive) reviews. Moreover, while distinct, these activities are fundamentally intertwined—for example, if reviews are not honest, then a seller may not feel compelled to fulfill sales faithfully.

Under the system we desire, then, (at least) two properties must hold:

1. It must be provable that under all conditions[3] (whether acting alone or in cooperation with others), fulfilling sales faithfully is expected to be more profitable than failing to do so;

2. It must be provable that under all conditions[3], the expected gain from collusion is less than that from acting individually.

Measurable expectation

Implicit in these properties is the requirement that the expectation associated with various actions must be *measurable*. This may sound like a trivial requirement, but in fact, most TRS formulations do not allow expected gain to be determined in general.

[3]More specifically, it must be provable subject to the conditions or assumptions under which the mechanism is claimed to apply. Designers often specify constraints or assumptions that must hold for the mechanism to operate properly.

For example, consider a (common) structure where each agent has a reputation score, and the agent with the highest score is selected. If I am honest (or ballot-stuff), and gain one additional positive review, what is the effect on my future expectation? The answer is entirely dependent on the situation. If my reputation score is less than the highest of my competitors, and the additional positive review does not allow me to surpass that competitor, then the additional review does not change my (immediate) expectation at all. (It may have value in the future, but this, too, is problematic to project, if we cannot forecast the future with certainty.) If I was already the leader, then the result is the same: the outcome of the next sale does not change. On the other hand, if I was previously trailing the leader, and the additional review allows me to pull ahead, then my increase in expectation might be enormous: not only will I win the next sale, but the enhanced reputation from that sale may allow me to win further sales (and hence, more positive reviews), and so on. Under such a system, determining the change in expectation (especially at design time) due to a positive review is challenging.

There are (at least) two ways in which one might attempt to design a system in which the rewards/penalties from positive/negative reviews are measurable:

- A system might be developed where trust has intrinsic, measurable value. In this way, the reward from gaining (or losing) trust can be directly evaluated. This was the approach taken in earlier work of ours, the Commodity Trunits system [43].

  Trunits are units of trust, which are gained through positive reviews, and lost through negative reviews. Trunits can be bought and sold. These two points, taken together, mean that the reward/penalty from *receiving* a review can be established precisely: the market value of the trunits gained/lost. This allows evaluation of actions such as honest sales, and ballot-stuffing reviews.

  Unfortunately, Trunits does not fully meet our requirements, because the reward from *giving* a review cannot be readily established. For example, if I engage in a bad-mouthing transaction, I may cause a competitor to lose trunits (and hence, money), but that action does not necessarily increase *my* reward. I may or may not improve my chance of winning future sales, because the targeted competitor may or may not decide to continue selling products after being victimized.

- A system might explicitly use a probability function to determine every agent's individual probability of winning a sale; such a probability function would likely take into account agents' previous histories of reviews. In this way, the effect of a review on reward can be determined: we can consider an agent's probability of winning the next sale before the review, and after the review, and calculate

the change of expectation. Both ballot-stuffing and bad-mouthing transactions are factored in, because both appear in the review histories.

(Note that we are only identifying characteristics that could make expectation measurable. We are not proposing specific such models, nor are we claiming that these characteristics alone would make a system 'collusion-proof'.)

Approach

We presume that no detection system is in use; as noted above, it is very difficult to establish probabilities of detection, which would make it difficult to render the types of proofs required. Thus, the system must ensure that collusive actions do not result in greater profit than non-collusive ones, but without knowledge of which actions are collusive, or which agents are colluders. From a mechanism design perspective, the goal would be to design rules that intrinsically result in non-collusive behaviour being more profitable than collusion, without consideration of the possibly-collusive intentions of traders. As noted, one possibility might be to treat actions differently depending on the degree of similarity of the agents' histories. For example, one might give reviews from agents with similar histories less weight than for those of agents with dissimilar histories. In this way, the impact of collusion might be reduced (because the collusive actions are similar), but there is no knowledge or consideration of the the real intentions or affiliations of agents during operation.

In a large marketplace with many agents, the number of possible sequences of events is enormous. Attempting to construct proofs that consider sequences of multiple transactions, then, is difficult at best. A more manageable approach is to consider any arbitrary, individual transaction, and to prove that after the transaction, the state of the marketplace continues to be such that properties 1 and 2 will hold. This is the perspective we take here.

8.3.2 Requirement 1: Faithful fulfillment

Consider an agent who is about to engage in a sale, and is contemplating whether or not to fulfill the sale honestly. The seller will receive the selling price S, as revenue, but will also incur cost C. We can decompose C into two factors: C_P, the cost to furnish the good (i.e., to produce it, or to purchase it for resale), and C_T, the costs associated in executing the transaction (primarily commission and/or fees, in the marketplace environment).

To simplify, we assume that C_P and C_T are constant proportions of selling price S, $r_P \in [0,1]$ and $r_T \in [0,1]$ respectively. (This assumption is not entirely unrealistic. Vendors often apply the same, or similar, markup percentages across the products they sell, and transaction costs such as commissions are often directly proportional to selling price.)

The profit on the sale, if the seller executes it faithfully, is

$$S(1 - r_P - r_T) \tag{8.1}$$

If the seller decides to cheat, we assume she does so maximally (i.e., does not even ship the good), and avoids the cost C_P entirely. Thus, the profit if the agent decides to cheat is

$$S(1 - r_T) \tag{8.2}$$

Subtracting the first from the second, it is clear that, on the immediate sale, cheating is more profitable:

$$S(1 - r_T) - S(1 - r_P - r_T) = Sr_P \tag{8.3}$$

The cost to acquire/produce goods is often quite a high percentage of selling price, so the difference is substantial. This fact is the very reason why TRSes are applied in marketplaces: cheating will yield a higher profit *on that specific transaction*. TRSes, however, can address this advantage by ensuring that the impact of this decision is felt in future transactions. If a seller cheats, she may earn more money immediately, but may lose future sales—in total, the cheating act may result in less profit rather than more.

Thus, we consider the effects on future sales. We begin by considering a hypothetical system where the effects of the positive/negative reviews are felt entirely in the next transaction (i.e., that cheating on sale t affects the probability of winning sale $t+1$, but does not affect $t+2$ or beyond). If the seller is honest on the current sale t (a case which we denote by H), then her probability of winning the next sale $t+1$ is $P(win|H)$, while if she is dishonest, the probability of winning is $P(win|\neg H)$. (Of course, $P(win|H) + P(win|\neg H) = 1$.) Details of $P(win)$ are discussed later in this section, but are not important for this analysis. For simplicity, we assume that both sales are of the same price.[4]

[4]We might, for example, have separate reputation ratings for different products or different tiers of prices, so the reputation earned from this sale can only be used for future sales at the same selling price. Alternatively, we might simply use a TRS that is immune to value imbalance—in this case, the selling prices on different sales might vary (and the analysis might be complex), but the ultimate outcome would parallel that outlined here.

In considering the possibility of a system where the impact of honesty/dishonesty is felt entirely within the next transaction, if we assume the most favourable treatment for an honest seller, then $P(win|H) = 1$ (and hence, $P(win|\neg H) = 0$).

The seller's expected profit includes both the amount earned for sale t (as above), as well as her expectation for sale $t+1$ depending on her choice. We wish honesty (on every transaction) to be more profitable than cheating on any transaction. Thus, we compare the profit from honesty with the two possible cases where cheating occurs. In the first, the seller is dishonest on t (and because $P(win|\neg H) = 0$, does not win $t+1$).[5] In the second case, the seller is honest on t, but dishonest on $t+1$.

Consider the first case, where the seller may be dishonest on t. For honesty to be more profitable than dishonesty, it must the case that

$$\begin{aligned} S(1-r_P-r_T)+P(win|H)S(1-r_P-r_T) &> S(1-r_T)+P(win|\neg H)S(1-r_T) \\ S(1-r_P-r_T)+(1)S(1-r_P-r_T) &> S(1-r_T)+(0)S(1-r_T) \\ 2(1-r_P-r_T) &> 1-r_T \\ 1-2r_P-r_T > 0 & \end{aligned} \tag{8.4}$$

Solving for r_P:

$$r_P < \frac{1-r_T}{2} \tag{8.5}$$

Considering the range of values for r_T is informative. If r_T was at its maximum of 1 (we presume agents don't sell at prices below transaction cost), it must be the case that $r_P < 0$— production cost must be less than 0, which is extremely unlikely. Considering the other (more reasonable) extreme of the range, $r_T = 0$ (e.g., no commission is charged), results in $r_P < 0.5$. This would require that on every transaction and for every seller, the seller's cost for the product was less than 50% of selling price, or stated another way, that sellers always apply more than a 100% markup on their cost. This cannot be guaranteed, especially if competition is present—in fact, this condition does not hold for many real world products/marketplaces.

Consider now the second case, where the seller may be honest on t but dishonest on

[5] Note that this then precludes the case where the seller is dishonest on t, and then honest on $t+1$.

$t+1$. For honesty to be more profitable than dishonesty, it must be the case that

$$\begin{aligned} S(1-r_P-r_T)+P(win|H)S(1-r_P-r_T) &> S(1-r_P-r_T)+P(win|H)S(1-r_T) \\ S(1-r_P-r_T)+(1)S(1-r_P-r_T) &> S(1-r_P-r_T)+(1)S(1-r_T) \\ S(1-r_P-r_T) &> S(1-r_T) \\ Sr_p &< 0 \end{aligned} \tag{8.6}$$

This would require the cost of production to be negative, which is untrue by assumption (and generally, in practice). Thus, by contradiction, honesty cannot be more profitable than dishonesty in this case.

Both cases yield essentially the same result. The conclusion (subject to our assumptions) is that even with the most favourable treatment for honesty, honesty cannot be guaranteed to be more profitable than dishonesty, if the payoff for honesty occurs only within a single round after the transaction. The reward for honesty must extend multiple transactions into the future.

8.3.3 Ensuring collusion is disadvantageous

Thus, we turn our attention to a hypothetical system where the effects of a positive review are felt for multiple future transactions. This is typical of many TRSes, in fact: a positive review is added to one's history, where it is used in all future purchase decisions. We consider such a system from the perspective of the second requirement, that collusive activity should not be advantageous.

We now subscript $P_i(win)$ to indicate the probability of winning sale i. Consider some sale t, and the impact of its result on winning future sales $i > t$. If the seller were not to win t (e.g., the seller decided not to participate, or another agent was selected), he would still have some probability $P_i(win)$ of winning each future sale i. But now, consider the case where the seller wins and receives a positive review on sale t. This review would be incorporated into his history, potentially affecting his probability of winning each future sale i. We denote the change, as a result of the positive review, in the seller's probability to win sale i, as $\Delta P_i(win)$. The seller's change in future expectation (assuming honesty on each sale, a required property for our system), then, as a result of the positive review on t, is

$$\Delta E = \Delta P_{t+1}(win)S(1-r_P-r_T)+\Delta P_{t+2}(win)S(1-r_P-r_T)+\cdots \tag{8.7}$$

Now, consider the case that the positive review received on sale t was a ballot-stuff. For ballot-stuffing to be less profitable than refraining from ballot-stuffing, it must be the case that

$$\begin{aligned} C &> \Delta P_{t+1}(win)S(1-r_P-r_T)+\Delta P_{t+2}(win)S(1-r_P-r_T)+\cdots \\ C &> \sum_{i=t+1}^{\infty}\Delta P_i(win)S(1-r_P-r_T) \end{aligned} \tag{8.8}$$

for the cost C incurred in the process of ballot-stuffing. This might simply be the cost of a fake ballot-stuffing transaction, Sr_T, or we may have other fees in place (e.g., membership fees, etc.)

Note that Equation 8.8 is an infinite series. The probability function will have to be very carefully constructed to make sure that the summation is less than C, which is constant—even very small values of $\Delta P_i(win)$ can cause the series to run to infinity. Mathematically, there are several ways this might be avoided, including:

- Construct $P_i(win)$ so that $\Delta P_i(win)=0$ for i beyond some k. This might come in the form of a 'window', a number of rounds beyond which reviews are discarded.
- Construct $P_i(win)$ so that $\Delta P_i(win)$ is negative for some i, balancing out the positive values. It seems very strange, however, to have positive reviews reducing the probability of winning, especially because we have no certainty that any given review was the result of a ballot-stuff.
- Construct $P_i(win)$ so that $\Delta P_i(win)$ decays as i increases, so that the summation asymptotically approaches some constant $k<C$.

All of these prospects seem challenging, but the difficulty may be worse than first realized. The reputation earned on sale t impacts the expectation for all $i>t$. But consider sale $t+1$: if the seller wins that sale as well (with increased probability due to the positive review on t), then the reputation from $t+1$ will *also impact every sale* $i>t+1$. $\Delta P_{t+2}(win)$, then, is impacted by reviews on t and $t+1$, and so on: $\Delta P_i(win)$ may be affected by positive reviews gained on all transactions $\{t,t+1,\ldots,i-1\}$. Recall that all of these reviews may have been acquired directly as a result of the ballot-stuff on t, without which the seller might not have won $t+1$. This might make it extremely difficult to construct $P_i(win)$ according to any of the options above. For example, is it reasonable to set $\Delta P_i(win)=0$ for i beyond some k, if the seller receives a positive review on sale $i-1$?

There are certain to be other requirements as well, but these two constraints significantly limit the set of possible solutions, at least for formulations of the sort that we outline. To ensure that both honesty and non-collusive behaviour are advantageous, the system designer must ensure that the effects of reviews are felt for more than one turn, but at the same time must limit the duration of such effects, even as reputation potentially grows dramatically.

8.4 Conclusion

A TRS in which collusion is provably disadvantageous is the ideal, but such a solution is likely to be difficult to obtain.

In contrast, while our collusion-resistant TRS approach does not readily allow such proofs, our results suggest it may be very effective in practice. The collusion detection algorithm has shown itself to be effective under a wide range of circumstances; moreover, punishment for collusion under our system is severe enough to deter many would-be colluders.

Chapter 9

Discussion, Future Work and Conclusion

In closing, we reflect on the novelty of our work in comparison with that of other researchers, we identify a number of possible directions for future research, and we summarize the specific contributions offered by this thesis.

9.1 Discussion

When first considering the problem of coalition detection, it appears to be closely related to other research areas; in fact, it may seem to be simply a particular instance or special case of an established research problem. Looking deeper, however, it becomes clear that there are essential differences between the problem we study, and those currently addressed in other fields. In fact, while some research areas study 'coalitions'/'teams'/'communities'/etc. by name, our work addresses fundamentally different problems and scenarios.

Game theorists have studied coalitions and coalitional games extensively, but from an entirely different perspective than our own. For example, work in this area typically focuses on questions such as: given information about payouts, utility functions, available resources, etc., what coalition will form?' In contrast, in the scenarios with which we are concerned, we are unlikely to have any such information. One might try to draw connections between the scenarios, such as linking the notion of 'benefit' with 'payout', but this is problematic. For example, in electronic marketplace settings, what is the 'payout' of

bad-mouthing activity? The actual act of bad-mouthing does not impact coalition members directly at all (only the targets' reputations are directly affected) and will not be felt immediately. Instead, any payout to the coalition will be realized in the future—if at all—over an indeterminate period and with unknown quantity. This is just one example of the areas of divergence between game theoretic work and our own.

Multiagent plan recognition seeks to identify the plan(s) in use by a team, while potentially identifying sub-teams as well. This work typically assumes access to a complete plan library and knowledge of team membership—the key questions here is, *what* is the team doing? In contrast, we assume no access to such a library, and no knowledge of team membership—the key question for us becomes, *who* is the team?

Community finding research seeks to identify smaller subsets of populations that constitute communities unto themselves. Superficially, this looks very similar to our problem. However, community finding researchers typically view interaction as the hallmark of community, while benefit is the central characteristic of cooperation. Interaction and benefit are not the the same, and in fact, need not even be connected. Members may benefit one another greatly without interacting at all (e.g., by bad-mouthing). Thus, existing community finding work is not directly applicable to our problem.

Our work represents an important step towards addressing the issues of coalitions and collusion: the ability to detect coalitions, and identify members, can allow measures to be taken that are appropriate to the situation. While our work was inspired by problems encountered in trust and reputation, and can be directly applied there, our work is not a trust and reputation proposal. Just as the issues of coalitions and collusion cut across many areas of multiagent systems (and the real world), the ability to detect coalitions (and our method for doing so) is likely to be applicable to many domains. The reputation system explored in Chapter 8 is just one example of how our methods might be applied to domain-specific problems. This work is likely to be interest in many of the areas of multiagent systems where the possible presence of unknown coalitions is of concern.

In the following section, we note a number of areas for future exploration. But even as one considers how to proceed in *conducting* this research, it is also interesting to consider how one might proceed in *using* this research. Presumably, one seeks to detect coalitions because one intends to do something with that information. The intended uses, however, may vary as much as the scenarios to which this work might be applied. As noted, selecting appropriate uses is likely to require domain expertise. In Chapter 8, we demonstrated one such usage: taking measures to 'correct' the unreliable information introduced by collusive manipulation. As the operator of a system, one might also use the information to, for example, penalize or fine coalition members, eject coalition members,

take action to compensate victims of collusion, etc. But the uses for a coalition detection system are potentially much broader than this. For example, an individual participating in a game or contest might use such a system to determine which competitors were acting in teams, in order to make strategic decisions (or simply to 'know what one is up against'). Alternatively, an individual might make use of such information in deciding if they wish to participate at all. An operator of a system may wish to use this tool to substantiate that no (detectable) coalitions are present in the system, in order to improve credibility and attract participants.

It might be unsatisfying to say (or read) that domain expertise is required to identify appropriate actions. Nevertheless, considering even the small sampling of possible uses noted here, and possible domains discussed in the next section, it should be clear that this is truly the case.

9.2 Future Work

We believe this to be the first work proposing broadly-applicable methods of detecting coalitions. As such, while a range of issues have been explored, there are many potential directions for future research. Here, we identify several noteworthy areas for continuing exploration.

Overlapping coalitions

In our work, we have assumed that agents belong to at most one coalition. This assumption is likely valid for many real-world coalitions and agents. Our algorithms, then, make use of hard partitioning between coalitions. The prospect exists, however, for agents who try to improve their position by belonging to multiple coalitions simultaneously. To address this issue, a first step would be to explore the use of 'fuzzy' clustering algorithms [27], which rather than returning a partitioning of agents, return a probability function over every agent-cluster pairing.

Disjoint sets of agents

An important area for investigation is situations where the sets of 'benefit givers' and 'benefit receivers' are disjoint.

In the example scenarios used when discussing trust and reputation systems for electronic marketplaces, agents may act as both buyers and sellers— each agent can give benefit (money and/or reviews), as well as receive it. This is reflective of many real-world marketplaces; it is also applies to numerous domains of interest: battlefields, games, etc.

In some scenarios, however, the sets of givers and receivers of benefit may be strictly disjoint. For example, in a marketplace there may be a strict partitioning between buyers and sellers. A more commonly-seen example is that of online reviews on web sites such as Amazon or TripAdvisor. Here, the items being reviewed (products, or hotels, for example) are a separate set from the entities providing reviews (e.g., consumers, or visitors). A coalition detection system would likely be of great value in combatting coordinated efforts to manipulate scores. Our current work would require modification to do so, however. Our characterization algorithm identifies coalitions by determining whether agents in a group benefit one another substantially more than outsiders. This approach is inapplicable when those giving and receiving benefit are disjoint: a group of 'givers' can't benefit themselves, because they do not receive benefit. Under these circumstances, a new characterization approach is required. One possibility might be to not only use similarity for clustering, but to use the degree of similarity to identify coalitions. Non-colluders making legitimate sales are likely to have reasonably varied purchasing patterns, but colluders engaged primarily in manipulating reviews may target unusually small/similar sets of products.

Agglomerative refinement

In Chapter 6.1, we propose a recursive refinement algorithm that allows the decomposition of clusters, to achieve more effective separation between members of different coalitions. As Figures 6.7 and 6.9 reveal, while separating members of different coalitions effectively, the recursive technique simultaneously does an admirable job of keeping members of the same coalition together. As shown in Figure 7.5, however, there is room here for performance to be further improved. One approach might be an agglomerative algorithm which, once decomposition has been completed, merges clusters that appear to consist of members of the same coalition.

Further enhancements

In our work, we make use of two different clustering methods (k-means and hierarchical) and two different means of measuring benefit ('raw' benefit, and our 'TF-IDF' approach).

As noted, there are many other clustering methods (and similarity measures, beyond Euclidean distance) that may be useful. Similarly, there are potentially many other ways of measuring benefit, even within the marketplace trust and reputation scenario. Moreover, in Section 6.3.2 we demonstrated how multiple clustering algorithms can be used simultaneously; in Section 7.3.3 we highlighted how the same approach could be used for multiple measures of benefit (or multiple similarity measures). Exploring a variety of different clustering methods, benefit measures, similarity measures, etc., is likely to yield further performance improvements.

Furthermore, it is likely that the running time of our algorithm can be improved. For example, benefit space has a large number of dimensions. It is likely that some of these dimensions are more useful than others in identifying the behaviour of agents. Dimensionality reduction and feature selection techniques might be applied here, to reduce the computation required.

Application to other domains

As noted throughout this document, we believe our approach to be applicable to a broad range of distinct domains.

While our work was inspired by problems in trust and reputation, and TRSes were the example domain in this document, our method is not fundamentally based on the characteristics of that domain. Certainly, we developed measures of benefit that were appropriate to that scenario—any such domain would require the same—but beyond this, the method is largely independent of, and ignorant of, the application scenario. A benefit measure from any domain could be used to map agents into benefit space; from here, the algorithm would proceed in exactly the same way as it has in this document. Our results in Section 7.3.3 support this idea: two entirely different benefit formulations both yielded strong results.

Of course, this is no guarantee that our techniques will be successful in any given domain. Indeed, success would require that:

- Key forms of benefit (specifically, the means by which a coalition advances its own purposes at the expense of outsiders) must be associated with observable actions. This is true in the case of trust and reputation; is is likely true for other domains as well, as we argue below.
- The quantity of collusive action required for the group to benefit meaningfully must be large enough to be detectable.

Returning to some of the possible applications cited in Chapter 1, we note actions and measures that might meet the above requirements. Of course, we are not experts in these domains, so we have limited insight into useful measures of benefit. Based on this limited knowledge, however, we suggest possible measures, to illustrate the potential applicability of our methods.

- In forms of gambling such as poker, collusion can yield significant advantages. For example, coalitions in poker can unfairly increase their winnings by betting against outsiders more aggressively than partners, ensuring that only the teammate with the strongest cards contests each hand (so that for a team of size n, each outsider's individual hand is competing with the coalitions 'best of n' hand). In this case, benefit would be observable: betting against someone would constitute an act of harm or aggression, while folding (i.e., dropping out without betting) would be a lack of harm (i.e., a form of benefit.)

- In contemporary insurgent/battlefield scenarios, it may be difficult to determine the allegiances of entities. There may be small groups of combatants obscuring their identities within a larger population, for example. It is possible that there are more than two groups of belligerents, as well. In a scenario like this, certain forms of benefit/harm seem obvious (and observable): for example, firing a weapon at someone is a clear act of harm. There are subtler observable actions, however, that are likely to indicate benefit/support, or harm/aggression, as well. For example moving in the same direction (e.g., moving in formation) might be a form of support, while moving on intersecting paths might indicate aggression. Shooting while another entity moves (i.e., 'covering fire') might be a form of benefit, as might moving to a wounded entity's position.

- In online games, many of the same principles noted above might apply. In such environments, though, acts of benefit/harm might be even more obvious. For example, many such games allow players to perform acts of healing on other players who are wounded—a clearly observable form of benefit. Giving equipment or supplies would be a form of benefit, while stealing would be a form of harm.

- In multi-party negotiations or diplomacy, advocating a point or action that is helpful to others may be a form of benefit (and/or harm to those whom the action does not help), while opposing such a point would be the opposite.

- In the commercial sphere, there are multitudes of possibly beneficial/harmful actions, depending on the situation. For example, supporting and/or adopting certain standards (e.g., Blu-Ray vs. HD-DVD) might benefit some companies, while

harming others. Licensing patents might be an act of benefit; refusing such license agreements, and/or law suits, would be clear acts of harm.

- In the types of cooperative multiagent systems noted in Chapter 1, there are likely to be clearly observable forms of benefit, as well: assisting an agent in a task, cooperating with another agent to achieve a mutually beneficial goal, etc.

It seems probable that our work will be applicable in a wide range of domains; this list is limited primarily by our own experience and imagination. Exploration of other domains is a key goal for future work, both to validate the broad applicability of our model, and to gain insights that may aid in its continuing improvement.

Real-world experimentation

Ideally, we would perform experiments using real-world data to validate our techniques. As noted, however, obtaining real-world data for our scenario is problematic: colluders in real-world marketplaces are not eager to identify themselves as such, and without labelled data, measuring accuracy is a problem. Unfortunately, it is not feasible to run a substantial, real-world marketplace for the purpose of gathering experimental data.

As we look to other application domains, however, real world experimentation seems much more practical. Experiments with live users playing games (e.g., poker, or FPSes) could be conducted, with subsets of players assigned to known coalitions. Such experiments might even be performed on a large scale, if they are conducted on the internet.

Multiagent plan recognition, and plan libraries

While we have outlined a variety of potential applications above, this work may present new research opportunities, as well. For example, while our work does not rely on known plan libraries, and in itself makes no effort to identify behaviours, it might be helpful for such purposes. One may wish to determine not only who is colluding, but also what strategies/tactics are in use. For instance, one might possess a partial plan library, and with knowledge of coalition membership, one might be able to identify some of the strategies in use. Alternatively, the ability to detect coalitions may allow for previously-unknown plans to be discovered, based on analysis of the actions of the identified agents.

Our work may help in another way to identify strategies in use. One key difference between our work, and that in the area of multiagent plan recognition, is our assumption

that there may be small coalitions embedded in a large population. In contrast, multiagent plan recognition work typically assumes that team membership is known; pattern matching is then used to identify the plan in use. In principle, it seems that one might use the same pattern-matching approach to identify small teams within a large population: matches would be likely to correspond to real teams executing a plan, while non-matches would be discarded. Unfortunately, this approach will be extremely computationally expensive, with even relatively small populations. A better approach might be to use our work as a first stage, in order to identify possible teams, and then use the multiagent plan recognition tools to match the actions of the identified teams against the plan library. In this way, the search space could be dramatically reduced, and the capabilities of the plan recognition algorithms expanded.

9.3 Conclusion

We believe that this document presents the first coalition detection and identification method in the literature that is likely to be broadly applicable across multiagent domains. The accuracy and robustness demonstrated in our validations are impressive, particularly given the novelty of the problem we seek to address. Contributions of this work include:

- TREET, the Trust and Reputation Experimentation and Evaluation Testbed, a platform allowing far more flexible experimentation and extensive evaluation than any tool previously available to trust and reputation researchers. This tool has value in the design, validation, and improvement of trust and reputation technologies. (Chapter 3)
- An experimental substantiation of practical attacks against vulnerabilities in TRSes. This novel work demonstrated that such weaknesses are pervasive, and substantial. (Chapter 4)
- The introduction of the idea that benefit, and similarity of benefit, is central to the detection of coalitions. (Section 5.2)
- The benefit space representation of agents, which allows agents who are similar in terms of who they are benefiting to be identified, e.g., by clustering. This representation (and the algorithms that make use of it) seem likely to be applicable to a variety of distinct domains. (Section 5.2.3)

- A characterization algorithm that allows groups of similar agents to be judged as either a coalition, or not a coalition. This algorithm was demonstrated to be very reliable, and particularly resistant to false positives. (Sections 5.3.3, 6.4.2)
- A technique for detecting coalitions based on clustering in benefit space and characterization, which does not rely on any knowledge of coalition configurations or strategies. This technique, the first general-purpose tool of its kind, has proven extremely accurate in testing. The combination of clustering in benefit space and characterization allows us to build an accurate classifier without training data. (Section 5.3)
- A recursive refinement algorithm that substantially improves the ability to separate agents from different coalitions into different clusters. This refinement technique allows clusters to be recursively subdivided—our characterization algorithm provides the basis for an effective stopping rule. (Section 6.1)
- A new method (specific to our purpose) to better determine the optimal number of clusters in a benefit data set, based on the characterization algorithm. (Section 6.3.1)
- A multi-clustering technique, which makes use of the false-positive-resistance of the characterization algorithm to allow multiple clustering methods to be used on any given benefit dataset, and the best result to be automatically selected. This approach, along with the clustering-optimization technique mentioned in the previous point, dramatically improved accuracy. (Section 6.3.2)
- A method for continuing to accurately identify coalition members and membership, even as agents' involvement with and membership in coalitions changes over time, based on the use of a forgetting factor. (Section 7.2)
- A new method for measuring benefit which is specific to the trust and reputation domain, using an approach modelled after the TF-IDF measure from information retrieval. (Section 7.3)
- A proposal for integrating multiple measures of benefit into the coalition detection process, based on the multi-clustering technique noted above. (Section 7.3.3)
- A collusion-resistant TRS, based on the incorporation of our coalition detection methods into the Beta Reputation System. This TRS was shown to be extremely effective in countering collusive activity; we are aware of no such TRS that has demonstrated this capability. (Chapter 8)

Our work represents significant strides towards addressing an issue that, despite its importance, has seen little progress. This work is likely to have value for both research, and real-world application.

References

[1] Charu C Aggarwal. A human-computer interactive method for projected clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 16(4):448–460, 2004.

[2] Robert J Aumann and Jacques H Dreze. Cooperative games with coalition structures. *International Journal of game theory*, 3(4):217–237, 1974.

[3] B.Douglas Bernheim, Bezalel Peleg, and Michael D Whinston. Coalition-proof nash equilibria i. concepts. *Journal of Economic Theory*, 42(1):1 – 12, 1987.

[4] Rajat Bhattacharjee and Ashish Goel. Avoiding ballot stuffing in ebay-like reputation systems. In *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 133–137, New York, NY, USA, 2005. ACM Press.

[5] Nikita Borisov. Computational puzzles as sybil defenses. In *Peer-to-Peer Computing, 2006. P2P 2006. Sixth IEEE International Conference on*, pages 171–176. IEEE, 2006.

[6] Michael Bowling, Brett Browning, and Manuela Veloso. Plays as effective multiagent plans enabling opponent-adaptive play selection. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS'04)*, 2004. in press.

[7] S. Braynov and T. Sandholm. Trust revelation in multiagent interaction, 2002.

[8] Sviatoslav Braynov and Tuomas Sandholm. Incentive compatible mechanism for trust revelation. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 310–311, New York, NY, USA, 2002. ACM Press.

[9] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Classification features for attack detection in collaborative recommender systems. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 547. ACM, 2006.

[10] Sandra Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, 2001.

[11] P.A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, page 74. ACM, 2005.

[12] Viet Dung Dang, Rajdeep K Dash, Alex Rogers, and Nicholas R Jennings. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 635. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[13] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 150–157. ACM, 2000.

[14] Gabrielle Demange. Intermediate preferences and stable coalition structures. *Journal of Mathematical Economics*, 23(1):45 – 58, 1994.

[15] J. Douceur. The sybil attack. *Peer-to-peer Systems*, pages 251–260, 2002.

[16] Eric Friedman, Paul Resnick, and Rahul Sami. Manipulation-resistant reputation systems. *Algorithmic Game Theory*, pages 677–697, 2007.

[17] Karen K. Fullam, Tomas B. Klos, Guillaume Muller, Jordi Sabater, Andreas Schlosser, Zvi Topol, K. Suzanne Barber, Jeffrey S. Rosenschein, Laurent Vercouter, and Marco Voss. A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 512–518, New York, NY, USA, 2005. ACM.

[18] W.A. Gamson. A theory of coalition formation. *American sociological review*, 26(3):373–382, 1961.

[19] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821, 2002.

[20] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):70, 1992.

[21] Valentin Goranko. Coalition games and alternating temporal logics. In *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge*, TARK '01, pages 259–272, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[22] Nathan Griffiths. Task delegation using experience-based multi-dimensional trust. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 489–496, New York, NY, USA, 2005. ACM Press.

[23] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[24] Chung-Wei Hang, Yonghong Wang, and Munindar P. Singh. An adaptive probabilistic trust model and its evaluation. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1485–1488, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[25] Maaike Harbers, Rineke Verbrugge, Carles Sierra, and J. Debenham. The examination of an information-based approach to trust. In P. Noriega and J. Padget, editors, *International Workshop on Coordination, Organization, Institutions and Norms (COIN)*, pages 101–112, Durham University, Durham, 2008.

[26] A. Harmon. Amazon glitch unmasks war of reviewers. *The New York Times*, 14, 2004.

[27] AK Jain, MN Murty, and PJ Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[28] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[29] Audun Jøsang and Roslan Ismail. The beta reputation system. 15th Bled Electronic Commerce Conference e-Reality: Constructing the e-Economy, June 2002.

[30] R. Jurca and B. Faltings. An incentive compatible reputation mechanism, 2003.

[31] Radu Jurca and Boi Faltings. Collusion-resistant, incentive-compatible feedback payments. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 200–209, New York, NY, USA, 2007. ACM.

[32] G.A. Kaminka, M. Fidanboylu, A. Chang, and M.M. Veloso. Learning the sequential coordinated behavior of teams from observations. *Lecture notes in computer science*, pages 111–125, 2003.

[33] Jonathan M. Katz. Hold 'em, fold 'em, cheat 'em. *Slate.com*, (http://www.slate.com/id/2112213), 2005.

[34] H.A. Kautz. A formal theory of plan recognition and its implementation. *Reasoning about plans*, 125, 1991.

[35] Nancy Keates. Deconstructing tripadvisor. June 2007.

[36] Reid Kerr and Robin Cohen. Modeling trust using transactional, numerical units. In *PST '06: Proceedings of the Conference on Privacy, Security and Trust*, Markham, Ontario, Canada, October 2006.

[37] Reid Kerr and Robin Cohen. Trunits: A monetary approach to modeling trust in electronic marketplaces. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06) Workshop on Trust in Agent Societies*, Hakodate, Japan, 2006.

[38] Reid Kerr and Robin Cohen. Towards provably secure trust and reputation systems in e-marketplaces. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–3, New York, NY, USA, 2007. ACM.

[39] Reid Kerr and Robin Cohen. Towards provably secure trust and reputation systems in e-marketplaces. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, Honolulu, Hawaii, USA, 2007.

[40] Reid Kerr and Robin Cohen. An experimental testbed for evaluation of trust and reputation systems. In *Proceedings of the Third IFIP WG 11.11 International Conference on Trust Management (IFIPTM'09)*, West Lafayette, Indiana, USA, 2009.

[41] Reid Kerr and Robin Cohen. Smart cheaters do prosper: Defeating trust and reputation systems. In *Proceedings of AAMAS'09*, Budapest, Hungary, 2009.

[42] Reid Kerr and Robin Cohen. TREET: The Trust and Reputation Experimentation and Evaluation Testbed. *Electronic Commerce Research*, 10(3):271–290, 2010.

[43] Reid Kerr and Robin Cohen. Trust as a tradable commodity: A foundation for safe electronic marketplaces. *Computational Intelligence*, 26(2):160–182, 2010.

[44] M. Klusch and A. Gerber. Issues of dynamic coalition formation among rational agents, 2002.

[45] B.N. Levine, C. Shields, and N.B. Margolin. A survey of solutions to the sybil attack. *University of Massachusetts Amherst, Amherst, MA*, 2006.

[46] S. Marsh. Formalising trust as a computational concept, 1994.

[47] Bhaskar Mehta and Thomas Hofmann. Ieee data eng. bull.; a survey of attack-resistant collaborative filtering algorithms. 31(2):14–22, 2008.

[48] Bhaskar Mehta, Thomas Hofmann, and Peter Fankhauser. Lies and propaganda: detecting spam users in collaborative filtering. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 14–21, New York, NY, USA, 2007. ACM.

[49] Bhaskar Mehta and Wolfgang Nejdl. Attack resistant collaborative filtering. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 75–82, New York, NY, USA, 2008. ACM.

[50] Bhaskar Mehta and Wolfgang Nejdl. Unsupervised strategies for shilling detection and robust collaborative filtering. *User Modeling and User-Adapted Interaction*, 19(1-2):65–97, 2009.

[51] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):26113, 2004.

[52] Zeinab Noorian, Stephen Marsh, and Michael Fleming. Multi-layer cognitive filtering by behavioral modeling. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 871–878, 2011.

[53] M.J. Osborne and A. Rubinstein. *A course in game theory*. The MIT press, 1994.

[54] Girish Keshav Palshikar and Manoj M. Apte. Collusion set detection using graph clustering. *Data Min. Knowl. Discov.*, 16(2):135–164, 2008.

[55] C. Piro, C. Shields, and B.N. Levine. Detecting the sybil attack in mobile ad hoc networks. In *Securecomm and Workshops, 2006*, pages 1–11. IEEE, 2006.

[56] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[57] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM New York, NY, USA, 1994.

[58] Paul Resnick and Rahul Sami. The influence limiter: provably manipulation-resistant recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 25–32. ACM, 2007.

[59] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[60] Raquel Ros, Manuela Veloso, Ramon López de Mántaras, Carles Sierra, and Josep Lluis Arcos. Retrieving and reusing game plays for robot soccer. In *Advances in Case-Based Reasoning. 8th European Conference on Case-Based Reasoning (ECCBR-06), Fethiye, Turkey, September 4-7, 2006*, volume 4106 of *Lecture Notes in Artificial Intelligence*, pages 47–61. Springer, 2006.

[61] P Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.

[62] Hosam Rowaihy, William Enck, Patrick McDaniel, and Thomas La Porta. Limiting sybil attacks in structured p2p networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2596–2600. IEEE, 2007.

[63] Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. *Artif. Intell. Rev.*, 24(1):33–60, 2005.

[64] Gerard Salton and Michael J McGill. Introduction to moderm information retrieval, 1983.

[65] JJ Sandvig, B. Mobasher, and R. Burke. Robustness of collaborative recommendation based on association rule mining. In *Proceedings of the 2007 ACM conference on Recommender systems*, page 112. ACM, 2007.

[66] O. Shehory and S. Kraus. Feasible formation of coalitions among autonomous agents in nonsuperadditive environments. *Computational Intelligence*, 15(3):218–251, 1999.

[67] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[68] Gita Sukthankar and Katia Sycara. Robust recognition of physical team behaviors using spatio-temporal models. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 638–645, New York, NY, USA, 2006. ACM.

[69] Gita Sukthankar and Katia Sycara. Policy recognition for multi-player tactical scenarios. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[70] Gita Sukthankar and Katia Sycara. Robust and efficient plan recognition for dynamic multi-agent teams. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1383–1388, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[71] Gita Sukthankar and Katia Sycara. Activity recognition for dynamic multi-agent teams. *ACM Transactions in Intelligent Systems Technology*, 3(1):18:1–18:24, oct 2011.

[72] M. Tambe. Tracking dynamic team activity. In *Proceedings of the National Conference on Artificial Intelligence*, pages 80–87, 1996.

[73] W. T. Teacy, Jigar Patel, Nicholas R. Jennings, and Michael Luck. Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2):183–198, 2006.

[74] W. T. L. Teacy, T. D. Huynh, R. K. Dash, N. R. Jennings, M. Luck, and J. Patel. The ART of IAM: The winning strategy for the 2006 competition. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'07) Workshop on Trust in Agent Societies*, Honolulu, Hawaii, USA, 2007.

[75] Thomas Tran and Robin Cohen. A learning algorithm for buying and selling agents in electronic marketplaces. In *AI '02: Proceedings of the 15th Conference of the*

Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, pages 31–43, London, UK, 2002. Springer-Verlag.

[76] Thomas Tran and Robin Cohen. Improving user satisfaction in agent-based electronic marketplaces by reputation modelling and adjustable product quality. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 828–835, Washington, DC, USA, 2004. IEEE Computer Society.

[77] W. Wei, F. Xu, C. Tan, and Q. Li. Sybildefender: A defense mechanism for sybil attacks in large social networks. 2013.

[78] Gerhard Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1999.

[79] Andrew Whitby, Audun Josang, and Jadwiga Indulska. Filtering out unfair ratings in bayesian reputation systems. In *Proceedings of the 7th Int Workshop on Trust in Agent Societies*, 2004.

[80] G. Wu, D. Greene, B. Smyth, and P. Cunningham. Distortion as a validation criterion in the identification of suspicious reviews. 2010.

[81] L. Xu, S. Chainan, H. Takizawa, and H. Kobayashi. Resisting sybil attack by social network and network clustering. In *Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on*, pages 15–21. IEEE, 2010.

[82] Junichi Yamamoto and Katia Sycara. A stable and efficient buyer coalition formation scheme for e-marketplaces. In *Proceedings of the fifth international conference on Autonomous agents*, AGENTS '01, pages 576–583, New York, NY, USA, 2001. ACM.

[83] Bin Yu and Munindar P. Singh. Distributed reputation management for electronic commerce. *Computational Intelligence*, 18(4):535–549, 2002.

[84] Haifeng Yu, Phillip B Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3–17. IEEE, 2008.

[85] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):267–278, August 2006.

[86] Giorgos Zacharia and Pattie Maes. Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14(9):881–907, 2000.

[87] Giorgos Zacharia, Alexandros Moukas, and Pattie Maes. Collaborative reputation mechanisms in electronic marketplaces. In *HICSS '99: Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 8*, page 8026, Washington, DC, USA, 1999. IEEE Computer Society.

[88] Jie Zhang and Robin Cohen. Evaluating the trustworthiness of advice about seller agents in e-marketplaces: A personalized approach. *Electronic Commerce Research and Applications*, 7(3):330–340, 2008.