Learning to Trust on the Move

Neal Lathia

Abstract Computational trust has been developed as a novel means of coping with uncertainty within collaborative communities of interacting peers. The idea now offers enourmous potential for use in pervasive mobile environments; however, to date there is little agreement about what *computational trust* itself means, and what the limitations that emerge from its use are. In this work, we project the idea of computational trust into machine learning terms, showing that trust is a metaphor that helps system designers reason about and exploit the intended deployment scenario to achieve their goals. Viewing a trust model as a strategy to confront a learning problem thus allows us to explore the effect that constraints, such as mobility and user participation, will have on the quantity of information available to learn from; in this work, we demonstrate this idea with a set of experiments on the Reality Mining Dataset. The results highlight that the most successful trust models will be based on strong contextual information about the environment they are to be deployed in.

1 Introduction

Computational trust has appeared as a important approach that can be used to address situations in networked systems where knowledge is incomplete or uncertain. Take, for example, the case of a customer deciding whether to buy an item on offer or accept a service from an online seller. This customer is faced with a situation where the information regarding the transaction is *asymmetric*: she does not know what the seller will do once she has committed to the transaction. In order to mitigate the risk of the customer losing her money, she can reason about how *trustworthy* the seller is, based on her and other customer's previous experiences.

Neal Lathia
Department of Computer Science, University College London, London WC1E 6BT, UK
e-mail: n.lathia@cs.ucl.ac.uk

Computational trust models aim at automating this reasoning, by imitating human interactions, in situations where complete knowledge is not available to all participants. These environments include distributed security protocols and collaborative groups; a number of examples are explored below. In this work we place a particular emphasis on trust models deployed in mobile environments, which operate between a set of intermittently connected (and co-located) *peers*, where the full set of peers forms a *community* of participants. We therefore are not focusing on trust issues that arise between users and their devices, or trust that lies on the interface level [1]. However, there is little consensus within broader trust management research as to the definition or purpose of trust itself within a computational environment; furthermore, little has been explored as to the suitability of computational trust to different potential application scenarios.

In this work we thus propose to refine the notion of computational trust, by reshaping it into a machine learning framework, in order to extrapolate the applicability and limitations of this idea when applied to mobile environments. Based on the characteristics of the refined definition, we show with a set of experiments that the utility of a trust model will be highly dependent on the particular features (such as mobility patterns) of the context where it is deployed, and therefore in depth knowledge of the application scenario is required before being able to design a successful trust model.

2 Trust: A Learning Problem

A significant point of debate within trust management research has been defining what *computational trust* itself means. The idea was first formalised in Marsh's PhD thesis [2]; however, one of the earliest and most often cited definitions of trust is that provided by Gambetta [3, 4]:

> "trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before [we] can monitor such action (or independently of his capacity of ever to be able to monitor it) and in a context in which it affects [our] own action."

As the comprehensive review in [5] discusses, this definition explores trust from the perspective of *reliability* between peers. Other definitions turn towards *decision making*, by describing trust as a factor that supports making the "correct" decision when incomplete or imperfect information is available and the response of the other peer is uncertain. Further definitions generalise away from subjective interactions towards the broader concept of *reputation*. These definitions propose to construct trust information based on multiple observations or interactions with a peer by different sources; they aim at capturing information by means of a collective effort. The idea of trust and reputation, however, are not strongly coupled [6], and each one can certainly be implemented without the other.

All of the proposed definitions describe trust based on the interaction between two entities, which may be mobile pervasive devices, or networked machines, by

pushing definitions that are applicable to human interactions down to computational scenarios. However, the difficulty in resolving the suitability of one definition over another is that trust between humans is a largely *qualitative characteristic* of relationships and, although its features can be extracted into definitions, there would be incredible difficulty to reduce them to quantitative, enumerable values as is done in computational trust. Expressing the worth, potential gain, or uncertainty of a future interaction as a single, aggregated numerical value (or an equivalent set of values contained within a trust table) based on previous interactions hardly seems to mimic the trust-based decisions made by humans; in fact, they seem to reduce the complexity of trust into simple numerical comparisons.

While the definitions of trust may vary, there seems to be a growing consensus on the emergent qualities of a trust model. These include the subjective and context-dependent nature of the models: peers will learn different trust values based on their own experiences (and who they receive recommendations from), and the way they learn will depend on the application scenario, which will often require adaptive, continuous updates of trust values as interactions grow.

So what is computational trust? The devices themselves are, to date, not fully recreating or enacting the complicated, unknown internal reasonings on trust that humans are doing; however, the *system designers and researchers* are making use of these concepts to build secure and collaborative systems. In other words, computational trust is a *metaphor*, or paradigm, describing how the interactions within the system should be played out: it captures the assumptions that we have of the data that is being manipulated, the range of actions that are possible on this data, the conditions we envisage should be satisfied for the system to be functioning correctly, and offers a language for describing such interactions. To that extent, the most prominent assumptions of a trust management scenario are:

- **Predictability**: learning to trust peers is only viable when their behaviour exhibits a certain amount of routine, or predictability. If peer behaviour was completely random, peers would not be reliable at all and trust models would not help in any application scenario.
- **Learning and Confidence**: Since peer behaviour is predictable, it can also be learned based on a number of observations or interactions. As the number of interactions increases, so does the number of examples that we have had an opportunity to learn from, and therefore so does our *confidence* in what we have learned [7].
- **Knowledge Propagation**: Peers can exchange what they have learned, in order to avoid requiring each pair of users to learn about each other from scratch [3]. This feature is complicated by the fact that what one peer learns about the next may not be suitable for a third; knowledge propagation requires more work than simply disseminating data over the community.
- **Behavioural Enforcement**: The utility of implementing trust models is in the potential they offer to change peer's behaviour over time, by operating as *incentives* for participants to behave appropriately [8].

This perspective of computational trust equates the task of deciding who to trust to an instance of a subjective semi-supervised learning problem [9]. A traditional example of an supervised problem is classifying data points into one many groups; deciding who to trust can also be viewed as learning to classify peers into the correct group (trustworthy; not trustworthy; semi-trustworthy, etc), and thus blurs the lines between trust management and collaborative filtering [10]. Each potential interaction between peers is a data point, and the evaluation of the interaction once it has taken place is the label that peers need to be able to predict. However, trust is a *semi*-supervised learning environment, since peers can only learn from the outcomes of interactions that took place. Deciding to not interact with another peer (perhaps because it is not currently trusted) is the equivalent of a data point with no label; thus the problem is semi-supervised. We can view trust as learning to predict what others will do, and reacting accordingly: learning is the keyword and central notion to computational trust. Lastly, the appropriate classification scheme will vary from one peer to the next: trust is *subjective*.

However, system designers are also implementing trust models in order to construct cooperative or secure environments. In other words, they have goals in mind when developing the system, which may, for example, include boosting cooperative behaviour by punishing free-riders [11]. To that extent, designing a computational trust model is also an exercise in algorithmic mechanism design [12]; we aim to build systems such that all are encouraged to perform in the manner preferred by the system designer. This perspective, by grounding the objectives of computational trust in learning and mechanism design, revokes the need to firmly define trust in itself, while highlighting that the principles extracted from human interactions can be used to guide the development and construction of secure or collaborative systems.

3 The Quantity and Quality of Information

Now that we have refined trust into a metaphor for learning interaction behaviour in different scenarios, we can see that the context-specific characteristic of trust models also gains significant importance. This is due to the fact that the wide range of application scenarios will have an equally wide range of diverse features, which will influence the ability we have to deploy a learning algorithm. These features can be categorised into four groups:

- **System Objectives**: Trust models have been deployed for a wide variety of reasons, ranging from mobile content filtering and service selection [13] to performing cooperative tasks (such as routing [14]), or implementing distributed security policies [15]. Each goal assumes differences in what the correct behaviour that needs to be learned is and the extent to which that is possible.
- **Resource Availability**: Trust models also vary in computational complexity and resource requirement. For example, reputation-based models may require a central, accessible server, while fully decentralised methods running on a mobile phone need to be lightweight in order to not overload the restricted processing

power of the device [16]. The limited resources challenge researchers to design learning methods that do not only satisfy a goal, but can do so under very specific conditions.

- **Behavioural Homogeneity**: The value added to a system by a trust model will also depend on how diverse the potential behaviours of the participants are. If all are uncooperative (or cooperative), there is little to be gained from using a trust model to guide interactions; the value of computational trust is inversely proportional to the level of predictability of each peer. This idea is closely tied to any policies that govern peer behaviour within the community.
- **Mobility Patterns**: Deploying trust models in mobile scenarios will further be affected by the mobility patterns that emerge from the application domain, by changing the amount of confidence that is possible in the learned trust values (and therefore what kind of trust models are applicable).

In other words, the quantity and quality of information available for peers to learn from depends on both the application scenario of the trust model and the design of the model itself. The first two of the above constraints are largerly dependent on the system developers, and somewhat equate to functional requirements imposed on the system. They will often be inter-related with the latter two, since resource deployment may be decided based on constraints brought on by factors like mobility, but will most significantly affect the *quantity* of information available for learning. The second two, on the other hand, play a different role: these limiting factors are emergent properties of system and are therefore much more difficult to anticipate. Knowing a priori, for example, how people will move within a particular context is not immediately available information, even though mobility models have been studied in depth [17]. The uncertainty relating to these factors is aggravated by the *temporal* nature of trust-based systems, since the amount of knowledge that is available will vary as interactions take place. These constraints will affect the *quality* of the information available to learn from. Both impose important limitations to trust models; in this work we focus on the constraints emerging from varying amounts of information.

4 Learning With Limited Information

The above factors will influence the applicability of trust models to a particular context: from this it is possible to see that in certain cases trust models will either not be very useful or may have to focus on particular aspects of the model more than others. For example, if the movement of users leads to a majority of only one-off interactions (and there are no *familiar strangers* [18]), then the focus of the trust model turns away from how to evolve repeat interactions towards how to deal with initial encounters (trust bootstrapping [16]), and how to correctly spread information to other peers (trust dissemination and propagation). Propagation itself is a limiting factor; propagating trust information over the entire network of peers may not be possible, or, more importantly, reliable.

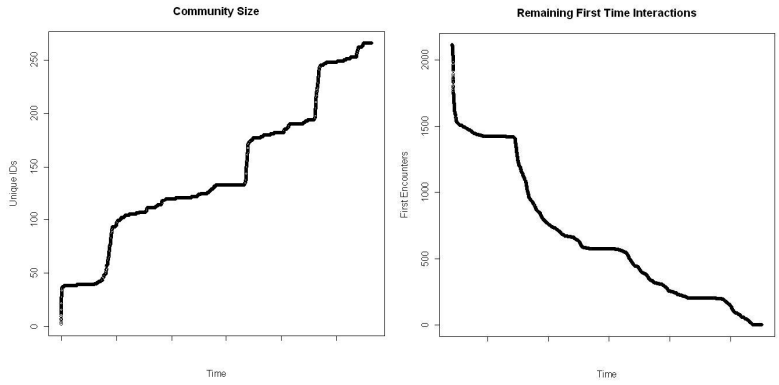

Fig. 1 Community Growth and Unique Interactions in Reality Mining Dataset

In essence, the application scenarios where trust models can be deployed will offer peers varying access to information, and will thus affect how suitably behavioural rules can be learned. Selecting different trust models will further constrain the ability that peers have to learn about each other, by affecting the extent to which interaction examples are available to each peer. Given a particular dataset, and a trust model (or learning strategy), it is possible to determine an upper bound to the amount of information available to each peer at each interaction.

To demonstrate the ideas outlined above, we performed a set of experiments using the Reality Mining Dataset [19]. The dataset contains both a list of bluetooth contacts, describing when users a and b are both connected and disconnected from each other, and social network information, enumerating the users that know each other. The social network dataset contains 77 unique identifiers; however, due to varying mobility patterns, a total of 264 unique identifiers appear in the contact dataset. These additional contacts represent connectivity with people who were not participating in the experiment, but were nonetheless co-located with those who were.

In the following experiment we considered the full contact dataset, without removing non-participant users. In fact, it is often the case that trust models are deployed in order to handle interactions with strangers, and thus removing them from our dataset would simply be observing how much information can be gained from a tightly-knit social network. The dataset's characteristics have already been extensively studied [20]; however, in this experiment we are interested in observing how different trust model strategies bound the ability each peer has learn over time. Figure 1 shows how the dataset evolves over time. The plot on the left depicts the number of unique identifiers that appear over time, as peers come into contact with others. An equivalent view of this temporal progression is the plot on the right, which depicts the number of first time interactions that are yet to happen. This perspective gives insight into how quickly groups are forming; if this curve fell quickly to zero, no more first-time interactions would occur and the dataset would be com-

prised of a number of isolated cliques that do not interact with others outside of their own group.

In this work we consider three generic strategies used in trust models:

- **Only Me (OM):** in this case, peers learn to trust others based only on their own experiences, and do not share this information with any others.
- **Neighbourhood Recommendations (NR):** peers accept recommendations about others from their neighbours [3]. The amount of experiences they have to learn about a peer p is the sum of their own and the number of experiences that have been had with p by their neighbours. In other words, if peer a meets b, b will let a know about all of its historical experiences, thus allowing a to gain from the number of experiences it has had.
- **Reputation Server (RS):** In this case, a central reputation server is available; peer a's knowledge of p will be based on *all* interactions that p has had in the past.

It is important to note that, as a first step, we aim only to quantify the amount of information available to learn trust from, rather than analysing the suitability or relevance of the information itself. It will often be the case that only a subset of the full range of examples will be useful to a peer in order to allow it to make the correct subjective decision about another peer. The *added value* of a recommendation is insofar unexplored, and we leave it as a matter of future work.

In this work we only consider the above three basic strategies. We do not consider more complex models that, for example, require multi-hop dissemination and propagation of trust information. However, the strategies we selected are representative of more complicated models; it is quite clear that the amount of information available from 2-hop dissemination will be bounded on either side by the OM and RS strategies.

5 Information Availability in the Reality Mining Dataset

The first experiment we performed examines the access that peers who behave differently will have to experiences to infer trust from. To highlight the results, we present an analysis comparing two users, who we have dubbed **Peer A** and **Peer B**:

- **Peer A**: this is the user who meets the highest number of other peers, meeting 101 different hosts in over $1,000$ connections. This host is therefore highly mobile (in terms of the number of contacts it meets), and will need to learn trust values for the highest number of other community members.
- **Peer B**: this user has the lowest proportion of unique contacts to number of overall connections. In 101 connections, this peer only meets 2 different hosts. In fact, this peer is most likely not one of the original experiment participants, but is a frequent "familiar stranger" to two other hosts. This peer is therefore particularly stationary, and only needs to know about and trust the two peers it frequently meets.

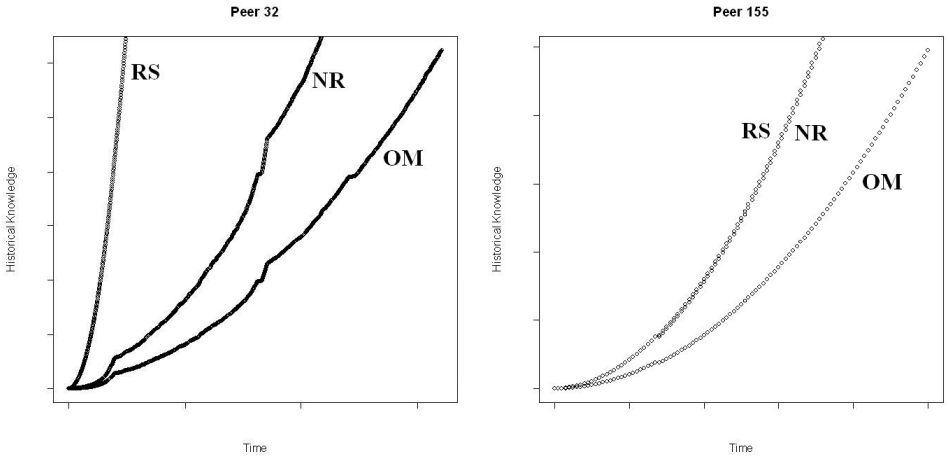

Fig. 2 Available information for users who meet a lot (left) or very few (right) contacts

Figure 2 shows the amount of information that is available to the peers about their neighbours over time, using the above three strategies. Information is quantified as follows: at each new connection, we sum the currently available historical interactions with all those that can be collected from potential recommendations. Historical encounters may thus be counted more than once; this reflects that once they have occurred they are available to be learned from multiple times. The plot on the left is relative to the peer A, while the plot on the right relates to peer B. In both cases, the "only me" strategy offers the least amount of information, and can be significantly improved by receiving recommendations from neighbours. The reputation server offers all available information, and is thus a strict upper bound to the number of observations each peer can learn from. However, the difference between the two peers is highlighted by the gain that a reputation server offers over one-hop recommendations; in the case of the high-contact peer, a reputation server delivers a tremendous increase in information, while there is little to no information to be gained with the centralised solution for the user with low churn.

The two users have a different number of contacts; amongst the first 101 connections, peer A meets 30 other hosts, while peer B only meets the 2 that it will continue meeting throughout the dataset. Figure 3 shows how the different strategies compare for the first 101 contacts made by each user. Overlaying the amount of information that each peer has available shows that the same strategy will offer varying amounts of information to each peer, and that the "only me" strategy offers the same amount of information to the low-mobility peer as the reputation server offers the peer that meets a lot of contacts, and needs to know about a broader range of hosts. It also becomes apparent that, due to the high rate of change in the contacts that peer A meets, both the OM and NR strategies offer minimal access to information; in fact, it is near impossible for this peer to learn anything since it is

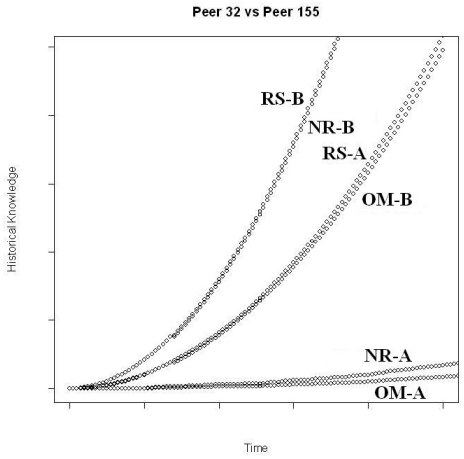

Fig. 3 Comparing the two peers

continuously faced and surrounded by new, unknown neighbours who have a very brief history of interactions within the system.

6 Learning With Uncooperative Users

While the first set of images were aimed at visualising the effect of application scenario characteristics (such as mobility) and trust model strategies on the availability of information that peers can use to learn from, the second set looks at the influence of behavioural homogeneity on access to information. In fact, the first set of images assumed a relatively homogeneous community. For example, when the neighbour recommendation protocol was adopted, all neighbours provided information on request. In other words, there was no lack of cooperation amongst the peers. How does information access vary when different proportions of the community are not cooperative?

To get an initial idea of the changes brought on by cooperation we consider a very simplistic behavioural model; peers will either be fully cooperative or not cooperative at all. This model is likely to be oversimplistic; however, measurements of the behaviour of nodes in realistic deployments is lacking, and most researchers are forced to make assumptions about the distribution of behaviour in the underlying community. As the proportion of uncooperative peers reaches the size of the community, the only available information can be drawn from each peer's own experiences, while if full cooperation is enforced, we can expect similar results to the strategies above. Otherwise, we can count how many experiences can be accessed. Figure 4 shows the effect of cooperation on peer A when seeking recommendations from neighbours; to simulate cooperation with our very basic model, we simply

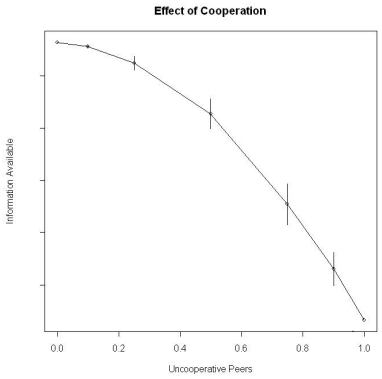

Fig. 4 Effect of cooperation on information availability

randomly selected varying proportions of peer A's neighbours to be uncooperative. Given a set of cooperative peers, we counted how many historical experiences peer A had access to regarding the other peers it encountered, and then repeated the entire random-selection and counting process 100 times. The figure shows that even when a majority of the community is non-cooperative, there is a lot to be gained from the few who will share their experiences over the "only me" strategy.

7 Discussion

In the above analysis we only included very basic trust model strategies, in order to observe the influence these strategies have on the availability of information, or training instances, to learn trust from. We did not, for example, consider the case of multi-hop trust propagation. Propagating trust over multiple hops adds a new layer of complexity, especially when recommendation information is being aggregated into trust values. Care has to be taken, therefore, to not gain too much confidence from recommendations received. For example, if peer a learns about neighbour p using recommendations from both b and c, where c's recommended trust is based on a direct experience and b's recommended trust is based on a recommendation that it received previously (from c), then although there were two recommendations there has only been one direct experience with p: the snowballing effect of propagated information may (but will not necessarily) lead to overfitting on the training examples.

On the other hand, the interesting point to note with regards to the two peers that we isolated for analysis above is that if full history recommendations are sought from any other within a two-hop radius, then both the low-mobility and high-contact peers have access to marginally less information than would be available from a full reputation server. In other words, the amount of historical experiences these peers have access to grows explosively as the number of hops increases.

Observing a correlation between community structure, behaviour, trust model strategy and the ability that peers have to learn from the system with the basic strategies we included here needs to be extended to include more comprehensive (and realistic) behavioural models, and, more importantly, how peers can maximise the utility they gain from the information made available to them.

This work has aimed at exploring two questions, the first addressing the definition of computational trust. Trust management research aims at developing models of interaction based on how humans interact; however, by converting a qualitative characteristic of human relationships into quantitative, measurable values, computational trust significantly departs from the notion of trust in human relationships . The resulting disparity between definitions of trust defined in sociology and the use of computational trust in environments like pervasive networks implies that the devices themselves are not reasoning about trust, but have been designed by researchers who are: computational trust is a metaphor used by developers that assumes devices can learn about each other, share their knowledge, and enforce behaviours in a community. Even in scenarios where humans do intervene, such as manually inputting trust values for other users in a recommender system [21], similar notions hold; in the cited context, the manual inputs are used to override the weakness that trust-based systems have when no one knows how much to trust others.

Once trust is defined as a learning problem it becomes apparent that the ability any device will have to learn will depend on how much information it has available to learn from. If too little information is available, the trust values are unreliable, just like learning algorithms that have not had sufficient training will not be able to fit appropriately on the data. On the other hand, too much information may lead to overfitting, and leaves hosts prone to attack and to making incorrect decisions. In this work we have explored how both trust model strategies and emergent properties of the application scenario, like mobility, will influence the amount of examples available to hosts implementing a trust model. Since community structure has a strong effect on the ability each peer has to learn from the others, leveraging in more social network information may offer further insight into how appropriate a trust model will be for a particular scenario. In fact, this kind of information has already been used as a defense mechanism [22] against sybil attacks, where a malicious peer gain an unfair advantage over others my using a large number of fake identities. The next step is to include what was disregarded here, that is, the *quality* of information, and the extent that extra information adds value to the trust that is learned, in order to boost the predictive power that trust values hold about peer behaviour.

References

1. J. Riegelsberger, M.A. Sasse, and J.D. McCarthy. The mechanics of trust: A framework for research and design. *International Journal of Human-Computer Studies*, 62:381–422, March 2005.
2. S. Marsh. Formalising trust as a computational concept. *PhD Thesis, Department of Mathematics and Computer Science University of Stirling UK*, 1994.

3. A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the* 33^{rd} *International Conference on System Sciences*, Hawaii, USA, 2000.
4. D. Gambetta. Can we trust trust? *Trust: Making and Breaking Cooperative Relations*, pages 213–238, 1990.
5. A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.
6. D. Quercia, S. Hailes, and L. Capra. B-trust: Bayesian trust framework for pervasive computing. In *Proceedings of the* 4^{th} *International Conference on Trust Management. LNCS*, pages 298–312, Pisa, Italy, May 2006.
7. S. D. Ramchurn, C. Sierra, L. Godo, and N.R. Jennings. A computational trust model for multi-agent interactions based on confidence and reputation. In *6th International Workshop of Deception, Fraud and Trust in Agent Societies*, Melbourne, Australia, 2003.
8. M. Ahmed and S. Hailes. A game theoretic analysis of the utility of reputation management. *Technical Report, Department of Computer Science, University College London*, January 2008.
9. S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 1995.
10. N. Lathia, S. Hailes, and L. Capra. Trust-based collaborative filtering. In *Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM)*, Trondheim, Norway, 2008.
11. D. Quercia, M. Lad, S. Hailes, L. Capra, and S. Bhatti. Strudel: Supporting trust in the dynamic establishment of peering coalitions. In *Proceedings of the* 21^{st} *ACM Symposium on Applied Computing*, pages 1870–1874, Dijon, France, April 2006.
12. N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Behavior*, 35:166–196, 2001.
13. L. McNamara, C. Mascolo, and L. Capra. Trust and Mobility Aware Service Provision for Pervasive Computing. In *Pervasive 2006 Workshop Proceedings: 1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures (RSPSI)*, pages 603–610. Springer-Verlag, May 2006.
14. S. Lee, R. Sherwood, and B. Bhattacharjee. Cooperative peer groups in nice. In *IEEE Infocom*, April 2003.
15. M. Carbone, M. Nielsen, and V. Sassone. A formal model for trust in dynamic networks. In *Int. Conference on Software Engineering and Formal Methods (SEFM)*, pages 54–63, Brisbane, Australia, September 2003.
16. D. Quercia, S. Hailes, and L. Capra. Lightweight distributed trust propagation. In *Proceedings of the* 7^{th} *IEEE International Conference on Data Mining*, Omaha, US, October 2007.
17. M. Musolesi and C. Mascolo. Designing mobility models based on social network theory. *Mobile Computing and Communications Review*, 11(3):166–196, July 2007.
18. S. Milgram. *The Familiar Stranger: An Aspect of Urban Anonymity*. Addison-Wesley, 1977.
19. N. Eagle and A.S Pentland. Reality mining: Sensing complex social systems. *Personal Ubiquitous Computing*, 10:255–268, 2006.
20. L. del Prete. Dynamic service composition in mobile environments. *First Year Report, Department of Computer Science, University College London*, June 2007.
21. P. Massa and P. Avesani. Trust-aware recommender systems. In *Proceedings of Recommender Systems (RecSys)*, Minneapolis, USA, October 2007.
22. H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. *SIGCOMM Computer Communications Review*, 36(4):267–278, 2006.