MASTER'S THESIS

Distributed Trust Evaluation in Ad-Hoc Networks

by Georgios Theodorakopoulos
Advisor:

CSHCN MS 2004-2
(ISR MS 2004-2)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

ABSTRACT

Title of Thesis: DISTRIBUTED TRUST EVALUATION IN AD-HOC NETWORKS

Georgios E. Theodorakopoulos,
Master of Science, 2004

Thesis directed by: Professor John S. Baras
Department of Electrical and Computer Engineering

An important concept in network security is trust, interpreted as a relation among entities that participate in various protocols. Trust relations are based on evidence related to the previous interactions of entities within a protocol. In this work, we are focusing on the evaluation process of trust evidence in Ad Hoc Networks. Because of the dynamic nature of Ad Hoc Networks, trust evidence may be uncertain and incomplete. Also, no pre-established infrastructure can be assumed. The process is formulated as a path problem on a directed graph, where nodes represent entities, and edges represent trust relations. We show that two nodes can establish an indirect trust relation without previous direct interaction. The results are robust in the presence of attackers. We give intuitive requirements for any trust evaluation algorithm. The performance of the scheme is evaluated on various topologies.

DISTRIBUTED TRUST EVALUATION IN AD-HOC NETWORKS

by

Georgios E. Theodorakopoulos

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2004

Advisory Commmittee:

Professor John S. Baras, Chair
Professor Virgil D. Gligor
Professor Richard J. La

DEDICATION

To my family, friends, and teachers.

ACKNOWLEDGEMENTS

I am grateful to my advisor Dr. John Baras for his continual guidance, and support. I would like to thank Dr. Virgil Gligor and Dr. Richard La for agreeing to serve on my committee and for reviewing my thesis.

Sincere thanks to my colleagues in the SEIL lab, and the CSHCN. Special acknowledgements are due to my officemates, Ulas Kozat, Gun Akkor, Burcak Keskin-Kozat, and Ayan Roy-Chowdhury. I had the most fruitful discussions with Tao Jiang, and the Wireless Information Assurance research group. Althia Kirlew was the major contributor in keeping administrative details to an absolute minimum. I cannot thank her enough for that.

Of the many friends I was lucky to make in College Park, I would like to single out Brent Anderson and Damianos Karakos for helping me adjust in an environment with unique challenges.

This work was supported by the U.S. Army Research Office under Award No. DAAD19-01-1-0494.

TABLE OF CONTENTS

Chapter 1

Introduction

An important concept in network security is the notion of trust, interpreted as a set of relations among entities that participate in various protocols [17]. Trust relations are based on the previous behavior of an entity within a protocol. They are determined by rules that evaluate, in a meaningful way, the evidence generated by this previous behavior. What is meaningful depends on the specific protocol (application), and on the entity that calculates the validity of the trust relation. The application determines the exact semantics of trust, and the entity determines how the trust relation will be used in the ensuing steps of the protocol.

For example, suppose that entity A wants to determine the public key that entity B owns. A and B have had no previous interactions, hence no trust relation, so A has to contact entities that have some evidence about B's key. Relevant pieces of evidence in this case are certificates binding B's key to B's identity. Also, the trustworthiness of the entities that issued these certificates should be taken into account. If A has had previous interactions with these issuing entities then their public keys as well as their trustworthiness will be known to A. Otherwise, the same steps will have to be repeated for the issuing entities, recursively. Finally, A will evaluate the whole body of evidence and establish a trust relation with B. In this case, the trust relation will be : "A does (or does not) believe that B's key is K_B".

The specification of admissible types of evidence, the generation, distribution, discovery and evaluation of trust evidence are collectively called Trust Establishment.

As first pointed out in [17], there are significant differences in the Trust Establishment process according to the type of network we are considering. Specifically, the characteristics peculiar to Ad-Hoc Networks are explored and contrasted against those of the Internet.

In this work, we are focusing on the evaluation process of trust evidence in Ad-Hoc Networks. We will be using the terms "trust evaluation", "trust computation", and "trust inference" interchangeably. This process is formulated as a path problem on a weighted, directed graph, where nodes represent users, and edges represent trust relations. Each node has direct relations only towards his one-hop neighbors, so all user interactions are local. The aim is for a node to establish an indirect relation with a node that is far away; this is achieved by using the direct trust relations that intermediate nodes have with each other. This locality requirement is a distinguishing feature of the work reported here.

We are imposing the following two main constraints on our scheme, based on the characteristics of the networks that we are dealing with:

- *There is no preestablished infrastructure.*

  The computation process cannot rely on, e.g., a Trusted Third Party. There is no Public Key Infrastructure, Certification Authorities, or Registration Authorities with elevated privileges.

- *Evidence is uncertain and incomplete.*

  Evidence is generated by the users on-the-fly, without lengthy processes. So, it is uncertain. Furthermore, in the presence of adversaries, we cannot assume

that all friendly nodes will be reachable: the malicious users may have rendered a small or big part of the network unreachable.

We require that the results are as accurate as possible, yet robust in the presence of attackers. It is desirable to, for instance, identify all allied nodes, but it is even more desirable that no adversary is misidentified as good. We use a general framework for path problems on graphs as a mathematical basis for our proposed scheme, and also give intuitive requirements that any trust evaluation algorithm should have under that framework. We evaluate the performance of the scheme with simulations on various topologies.

1.1 Organization

This thesis is organized in five chapters. In the Introduction, the current chapter, the trust evaluation problem is placed into context, and the aims for our approach are set. The second chapter describes and comments on related work that has been done in the field of trust computation. The main ideas are exposed, and representative examples are given. The third chapter explains our approach, proposes a mathematical framework for trust computation, and describes intuitive properties that any scheme under this framework should have. In the fourth chapter, our proposed scheme is used for actual computation scenarios, and the results are discussed. The fifth chapter concludes the thesis and suggests future directions for improvement.

Chapter 2

Related Work

In this chapter we are examining previous work that is relevant to the evaluation part of the Trust Establishment process. The main aim is to expose and comment on the ideas presented in that work. After this taxonomy, representative examples are given, that serve to illustrate the salient points.

2.1 Taxonomy

2.1.1 System Model

The most commonly used model is a labeled, directed graph. Nodes represent entities, and edges represent binary trust relations. These relations can be (for an edge $i \rightarrow j$): a public key certificate (issued by i for j's key), the likelihood that the corresponding public key certificate is valid, the trustworthiness of j as estimated by i, etc.

2.1.2 Centralized vs decentralized trust

By centralized trust we refer to the situation where a globally trusted party calculates trust values for every node in the system. All users of the system ask this trusted party to give them information about other users. The situation described has two important implications: First, every user depends on the trustworthiness of this single party, thus turning it into a single point of failure. Second, it is reasonable to assume that different users are expected to have different opinions about

the same target; this fact is suppressed here.

The decentralized version of the trust problem corresponds to each user being the "center of his own world". That is, users are responsible for calculating their own trust values for any target they want. This "bottom-up" approach is the one that has been most widely implemented and put into use, as a part of PGP [50] for public key certification.

Note that the distinction just mentioned refers to the semantics of trust. The actual algorithm used for the computation of trust is a separate issue: all data may be gathered at a single user, where the algorithm will be executed; or the computation may be done in a distributed fashion, throughout the network; or the algorithm may even be localized, in the sense that each node only interacts with his local neighborhood, without expecting any explicit cooperation from nodes further away.

2.1.3 Proactive vs reactive computation

This is an issue more closely related to the communication efficiency of the actual implementation. The same arguments as in routing algorithms apply: Proactive trust computation uses more bandwidth for maintaining the trust relationships accurate. So, the trust decision can be reached without delay. On the other hand, reactive methods calculate trust values only when explicitly needed. The choice depends largely on the specific circumstances of the application and the network. For example, if local trust values change much more often than a trust decision needs to be made, then a proactive computation is not favored: The bandwidth used to

keep trust values up to date will be wasted, since most of the computed information will be obsolete before it is used.

2.1.4 Extensional vs intensional (scalar vs group) metrics

One possible criterion to classify uncertainty methods is whether the uncertainty is dealt with *extensionally* or *intensionally*. In extensional systems, the uncertainty of a formula is computed as a function of the uncertainties of its subformulas. In intensional systems, uncertainty is attached to "state of affairs" or "possible worlds". In other words, we can either aggregate partial results in intermediate nodes (in-network computation), or we can collect all data (opinions and trust topology) at the initiator of the trust query and compute a function that depends on all details of the whole graph.

As pointed out by Maurer [36], there seems to be a trade-off between computational efficiency and semantic correctness. Extensional systems are more efficient, whereas intensional ones are more correct. The notion of semantic correctness seems to be related to the attack resistance of a metric, since Levien ([28]) claimed that scalar metrics (as he called extensional systems) are vulnerable to single-node attacks (see next section).

2.1.5 Attack resistance (node/edge attacks)

Levien ([29]) suggested a criterion for measuring the resistance of a trust metric to attackers. First, he distinguished between two types of attacks: node attacks, and edge attacks. Node attacks amount to a certain node being impersonated. So,

the attacker can issue any number of arbitrary opinions (public key certificates in Levien's case) from the compromised node about any other node.

Edge attacks are more constrained: Only one false opinion can be created per each attack. In other words, an attack of this type is equivalent to inserting a false edge in the trust graph. Obviously, a node attack is the more powerful of the two, since it permits the insertion of an arbitrary number of false edges.

The attack resistance of a metric can be gauged by the number of node or edge attacks, or both, that are needed before the metric can be manipulated beyond some threshold. For instance, in [42] Reiter and Stubblebine show that a single misbehaving entity (a 1-node attack) can cause the metric proposed in [3] to return an arbitrary result.

Here an important clarification has to be made: there are trust graphs that are "weaker" than others. When, for example, there exists only a single, long path between the source and the destination, then any decent metric is expected to give a low trust value. So, the attack resistance of a metric is normally judged by its performance in these "weak" graphs. This line of thinking also hints at why intensional systems (group metrics) perform better than extensional: They take into account the whole graph, so they can identify graph "weaknesses" more accurately.

2.1.6 Negative and positive evidence (certificate revocation)

It is desirable to include both positive and negative evidence in the trust model. The model is then more accurate and flexible. It corresponds better to real-life situations, where interactions between two parties can lead to either satisfaction or

complaints. When a node is compromised (e.g. its private key is stolen) the public key certificates for this node should be revoked. So, revocation can be seen as a special case of negative trust evidence.

On the other hand, the introduction of negative evidence complicates the model. Specifically, an attacker can try to deface good nodes by issuing false negative evidence about them. If, as a countermeasure to that, issuing negative evidence is penalized, good nodes may refrain from reporting real malicious behavior for fear of being penalized.

2.1.7 What layer should trust be implemented in?

An important issue that is often glossed over is the layer at which the trust protocol will operate. That is, the services required by the protocol and the services it offers should be made clear, especially its relationship to other security components. As pointed out in [7], some secure routing protocols assume that security associations between protocol entities can be established with the use of a trust establishment algorithm, e.g. by discovering a public key certificate chain between two entities. However, in order to offer its services, the trust establishment algorithm may often assume that routing can be done in a secure way. This creates a circular dependency that should be broken if the system as a whole is to operate as expected.

2.2 Representative Examples

Some of the following examples have been cast in the public key certification framework, whereas others are more general. However, they can all be viewed as trust

evaluation metrics, insofar as they compute a trust "value" for a statement like "Is this public key certificate valid?".

2.2.1 Decentralized Trust Management

Blaze, Feigenbaum, and Lacy [6] seem to have been the first to introduced the term "Trust Management", and identified it as a separate component of security services in networks. They designed and implemented the PolicyMaker trust management system, which provided a unified framework for describing policies (rules), credentials (trust evidence), and trust relationships. Also, this system was locally controlled, since it did not rely on a centralized authority to evaluate the credentials: Each user had the freedom and the responsibility to reach his own decisions.

The main issues in this and related work (KeyNote [5], SPKI/SDSI [13], Delegation Logic [30], Trust Policy Language [22], also [46]) are: the language in which the credentials and the policies will be described; the compliance-checking algorithm that checks if the credentials satisfy the policy rules; and the algorithm for the discovery of the credentials in the first place (remember, credentials can be stored throughout the network). Note that a graph is often used for depicting the credentials issued by an entity i for an entity j, and the edges of the graph are labeled according to the parameters of the credential.

2.2.2 PGP trust metric

In PGP [50], a distinction is made between the *validity* of a public key and its *trust level*. Bob's key is valid for Alice, if Alice believes that it really belongs to Bob. The

trust level of Bob's key corresponds to how carefully Bob authenticates keys before issuing certificates for them. The trust levels of the keys known to Alice are assigned in any way Alice wants. PGP only determines the validity of a key according to how many keys have signed it, and what the trust levels of the signing keys are. The default rules for computing the validity of keys are described next, but a user is free to change them.

Trust level of signing key	Validity rule
unknown	Certificates signed with unknown keys are ignored.
untrusted	Certificates signed with untrusted keys are ignored.
marginally trusted	Key is valid if 2 or more marginally trusted keys have signed it.
fully trusted	Key is valid if 1 or more fully trusted keys have signed it.

2.2.3 Probabilistic

Maurer's metric ([36],[27]) assigns weights $w_{ij} \in [0, 1]$ to edges. These weights correspond to i's opinion about the trustworthiness of the certificate issued for j's public key, i.e. to what degree i believes that the {public key – owner ID} binding implied by the edge $i \rightarrow j$ has been properly certified. The weights are then treated as link survival probabilities. The metric calculates the probability that at least one path survives that leads from the entity evaluating the metric to the entity involved in the certificate in question.

2.2.4 Path Independence

Reiter and Stubblebine ([41],[42]) introduced the concept of path independence for entity authentication. They argued that multiple independent paths are a safer way to authenticate Bob than either the reachability or the bounded reachability metric. Their proposal was a Bounded Length Independent Paths metric which returns a set of node-disjoint paths from Alice to Bob which are shorter than K hops. Since the computation of this metric is an NP-complete problem for $K \geq 5$ they gave approximation algorithms. Note that if we drop the bounded length constraint, the problem becomes polynomial.

In a subsequent paper the same people suggested a different metric, based on network flow techniques. The model being the same, weights were added on the edges indicating the amount of money that the issuer will pay to anyone who is misled because of the certificate. Being misled means falsely authenticating the certified entity or incurring losses because the certified entity misbehaves. Treating the edge weights as capacities, the metric calculates the maximum flow from Alice to Bob. This is the minimum amount of money for which Alice is insured in the case of her being misled by Bob's key. Note that if all edges are assigned unit capacities, this metric calculates the number of edge-disjoint paths from Alice to Bob.

2.2.5 Flow based

Levien's metric ([29]) is also network flow based. After assigning edge capacities the metric treats trust as a commodity that flows from Alice to Bob. Alice has unit

quantity of trust and tries to send it to Bob. The metric calculates how much of this unit quantity reaches Bob. By suitably assigning capacities, the metric is made more resistant to attacks. However, some assumptions in this work are not realistic, e.g. that all nodes have the same indegree d.

2.2.6 Subjective Logic

Jøsang ([24]) has developed an algebra for assessing trust relations, and he has applied it to certification chains. To a statement like "The key is authentic" he is assigning a triplet (called *opinion*) $(b, d, u) \in [0, 1]^3 : b + d + u = 1$, where b, d, and u designate belief, disbelief, and uncertainty respectively. Belief (disbelief) in a statement increases when supporting (contradicting) evidence appears. Uncertainty is caused by the lack of evidence to support either belief or disbelief. When uncertainty is zero, these triplets are interpreted as a traditional probability metric. An opinion is qualified by the user who issues it, and by the statement it refers to: $\omega_Y^X = \{b_Y^X, d_Y^X, u_Y^X\}$ is user X's opinion about Y. Y can be a user, in which case ω_Y^X is X's opinion about the quality of Y's recommendations, or Y can be a statement such as "The key is authentic".

2.2.7 Local Interaction

Trust computation based on interactions with one-hop physical neighbors is a typical case for extensional systems. In [8], for instance, first-hand observations are exchanged between neighboring nodes. Assume i receives from j evidence about k. First of all, i adjusts his opinion for j, based on how close j's evidence is to i's

previous opinion about k. If it is not closer than some threshold, the new evidence is discarded, and i lowers his opinion about j. Otherwise, i increases his trust for j and the new evidence is merged with i's existing opinion for k. The whole process is based on local message exchange.

In [33], a group Q of users is selected, and they are asked to give their opinion about a certain target node. The end result is a weighted average of their opinions and any preexisting opinion that the initiator node may have. One possible selection for the group Q is the one-hop neighbors of the initiator.

In the EigenTrust algorithm [26], nodes exchange vectors of personal observations (called *local trust values*) with their one-hop neighbors. Node i's local trust value for node j is denoted by c_{ij}. These trust values are normalized ($\forall i : \sum_j c_{ij} = 1$). Each node i calculates global trust values t_{ij} for all other nodes j by the following iterative computation: $t_{ij}^{(n+1)} = \sum_k c_{ik} t_{kj}^{(n)}$, where $t_{kj}^{(0)} = c_{kj}$. If $C = [c_{ij}]$ is the local trust value matrix (row i holds node i's local trust values), then the above iteration essentially solves the following system of linear equations for T:

$$T = CT$$

where $T = [t_{ij}]$ contains the global trust values.

Under some assumptions for C, all rows of T are identical: All nodes i have the same opinion about any particular node j. The assumptions for C are that it is irreducible and aperiodic. If C is viewed as the transition probability matrix of a Markov chain, then each of T's rows is the steady state probability distribution,

and also the left principal eigenvector of C.

Chapter 3

Our Approach

3.1 System Model

We view the trust inference problem as a generalized shortest path problem on a weighted directed graph $G(V, E)$ (*trust graph*). The vertices of the graph are the users/entities in the network. A weighted edge from vertex i to vertex j corresponds to the *opinion* that entity i, also referred to as the *issuer*, has about entity j, also referred to as the *target*. The weight function is $l(i, j) : V \times V \longrightarrow S$, where S is the opinion space.

Each opinion consists of two numbers: the *trust* value, and the *confidence* value. The former corresponds to the issuer's estimate of the target's trustworthiness. For example, a high trust value may mean that the target is one of the good guys, or that the target is able to give high quality location information, or that a digital certificate issued for the target's public key is believed to be correct. On the other hand, the confidence value corresponds to the accuracy of the trust value assignment. A high confidence value means that the target has passed a large number of tests that the issuer has set, or that the issuer has interacted with the target for a long time, and no evidence for malicious behavior has appeared. Since opinions with a high confidence value are more useful in making trust decisions, the confidence value is also referred to as the *quality* of the opinion. The space of opinions can be visualized as a rectangle (ZERO_TRUST, MAX_TRUST)×(ZERO_CONF, MAX_CONF) in the Cartesian plane (Figure 3.1, for $S = [0, 1] \times [0, 1]$).

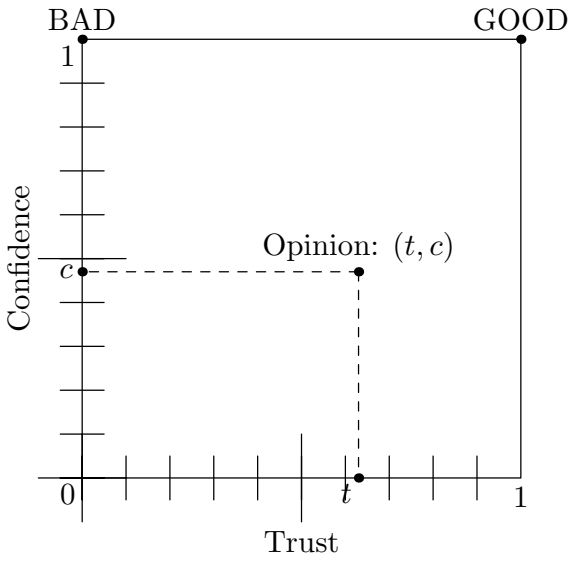

Figure 3.1: Opinion space

Both the trust and the confidence values are assigned by the issuer, in accordance with his own criteria. This means that a node that tends to sign public key certificates without too much consideration will often give high trust and high confidence values. The opposite holds true for a strict entity. When two such entities interact, it is important for the stricter entity to assign a low enough trust value to the less strict one. Otherwise, the less strict entity may lead the stricter one to undesirable trust decisions. This situation is easier to picture in the context of Certification Authorities and public key certification. There, a CA A will only give a high trust value to B, if B's policy for issuing certificates is at least as strict as A's and has the same durability characteristics [17].

Also, it is assumed that nodes assign their opinions based on local observations. For example, each node may be equipped with a mechanism that monitors neighbors for evidence of malicious behavior, as in [35]. Alternatively, two users may come in

close contact and visually identify each other, or exchange public keys, as suggested in [12]. In any case, the input to the system is local: however, extant pieces of evidence based on, e.g., previous interactions with no longer neighboring nodes can also be taken into account for the final decision. This would come into play when two nodes that have met in the past need now to make a trust decision for each other. Of course, the confidence value for such evidence would diminish over time. One consequence of the locality of evidence gathering is that the trust graph initially overlaps with the physical topology graph: The nodes are obviously the same, and the edges are also the same if the trust weights are not taken into account. As nodes move, opinions for old neighbors are preserved, so the trust graph will have more edges than the topology graph. However, as time goes by, these old opinions fade away, and so do the corresponding edges.

In the framework described, two versions of the trust inference problem can be formalized. The first is finding the trust-confidence value that a source node A should assign to a destination node B, based on the intermediate nodes' trust-confidence values. Viewed as a generalized shortest path problem, it amounts to finding the generalized distance between nodes A and B. The second version is finding the most trusted path between nodes A and B. That is, find a sequence of nodes $\langle v_0 = A, v_1, \ldots, v_k = B \rangle : (v_i, v_{i+1}) \in E, 0 \leq i \leq k-1$ that has the highest aggregate trust value among all trust paths starting at A and ending at B. A high level view of the system is shown in Figure 3.2.

Both problems are important: finding a target's trust value is needed before deciding whether to grant him access to one's files, or whether to disclose sensitive

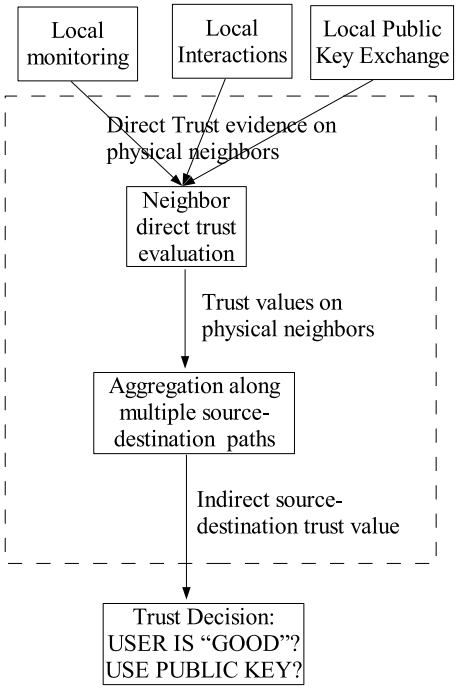

Figure 3.2: System operation

information, or what kind of orders he is allowed to give (in a military scenario, for instance). With this approach, a node will be able to rely on other nodes' past experiences and not just his own, which might be insufficient. The second problem is more relevant when it comes to actually communicating with a target node. The target node being trustworthy is one thing, but finding a trusted path of nodes is needed, so that traffic is routed through them. Note that this does not necessarily reduce to the previous problem of finding the *trust distance* between the nodes, as is the case for the usual shortest path problem in a graph. In the trust case, we will usually utilize multiple trust paths to find the trust distance from the source to the destination, since that will increase the evidence on which the source bases its final estimate. Consequently, there may be more than one paths contributing to this estimate.

The core of our approach is the two operators that are used to combine opinions: One operator (denoted $\otimes$) combines opinions along a path, i.e. A's opinion for B is combined with B's opinion for C into one indirect opinion that A should have for C, based on B's recommendation. The other operator (denoted $\oplus$) combines opinions across paths, i.e. A's indirect opinion for X through path p_1 is combined with A's indirect opinion for X through path p_2 into one aggregate opinion that reconciles both. Then, these operators can be used in a general framework for solving path problems in graphs, provided they satisfy certain mathematical properties, i.e. form an algebraic structure called a semiring. More details on this general framework are in section 3.2. Two existing trust computation algorithms (PGP [50] and EigenTrust [26]) are modeled as operations on two particular semirings. Note that

our approach differs from PGP in that it allows the user to infer trust values for unknown users/keys. That is, not all trust values have to be directly assigned by the user making the computations. The operators are discussed in greater depth in section 3.3.

3.2 Semirings

For a more complete survey of the issues briefly exposed here, see [43].

3.2.1 Definitions

A *semiring* is an algebraic structure $(S, \oplus, \otimes)$, where S is a set, and $\oplus, \otimes$ are binary operators with the following properties $(a, b, c \in S)$:

- $\oplus$ is commutative, assosiative, with a neutral element $\textcircled{0} \in S$:

$$\begin{aligned} a \oplus b &= b \oplus a \\ (a \oplus b) \oplus c &= a \oplus (b \oplus c) \\ a \oplus \textcircled{0} &= a \end{aligned}$$

- $\otimes$ is associative, with a neutral element $\textcircled{1} \in S$, and $\textcircled{0}$ as an absorbing element:

$$\begin{aligned} (a \otimes b) \otimes c &= a \otimes (b \otimes c) \\ a \otimes \textcircled{1} = \textcircled{1} \otimes a &= a \\ a \otimes \textcircled{0} = \textcircled{0} \otimes a &= \textcircled{0} \end{aligned}$$

- $\otimes$ distributes over $\oplus$:

$$\begin{aligned}(a \oplus b) \otimes c &= (a \otimes c) \oplus (a \otimes c) \\ a \otimes (b \oplus c) &= (a \otimes b) \oplus (a \otimes c)\end{aligned}$$

A semiring $(S, \oplus, \otimes)$ with a partial order relation $\preceq$ that is monotone with respect to both operators is called an *ordered semiring* $(S, \oplus, \otimes, \preceq)$:

$$a \preceq b \text{ and } a' \preceq b' \Longrightarrow a \oplus a' \preceq b \oplus b' \text{ and } a \otimes a' \preceq b \otimes b'$$

An ordered semiring $(S, \oplus, \otimes, \preceq)$ is ordered by the *difference relation* if:

$$\forall a, b \in S : (a \preceq b \iff \exists z \in S : a \oplus z = b)$$

A semiring is called idempotent when the following holds:

$$\forall a \in S : a \oplus a = a$$

3.2.2 Semirings for path problems

In the context of the generalized shortest path problem in a weighted graph, $\otimes$ is the operator used to calculate the weight of a path based on the weights of the path's edges:

$$p = (v_0, v_1, \ldots, v_k), \quad w(p) = w(v_0, v_1) \otimes w(v_1, v_2) \otimes \ldots \otimes w(v_{k-1}, v_k)$$

The $\oplus$ operator is used to compute the shortest path weight d_{ij} as a function of all paths from the source i to the destination j:

$$d_{ij} = \bigoplus_{\substack{p \text{ is a path} \\ \text{from } i \text{ to } j}} w(p)$$

In the familiar context of edge weights being transmission delays, the semiring used is $(\Re_+ \cup \{\infty\}, \min, +)$, i.e. $\oplus$ is min, and $\otimes$ is $+$: The total delay of a path is equal to the sum of all constituent edge delays, whereas the shortest path is the one with minimum delay among all paths. Also, ⓪ is ∞, and ① is 0. On the other hand, if edge weights are link capacities, then the maximum bottleneck capacity path is found by the semiring $(\Re_+ \cup \{\infty\}, \max, \min)$, with ⓪ $\equiv 0$, ① $\equiv \infty$. The transitive closure of a graph uses the Boolean semiring: $(\{0, 1\}, \vee, \wedge)$, where all edge weights are equal to 1. This answers the problem of path existence.

Note that the $\oplus$ operator may pick a single path weight (as is the case with max and min) or it may explicitly combine information from all paths (addition or multiplication).

3.2.3 Semirings for systems of linear equations

An equivalent way to describe the previous shortest path problem is by way of a system of equations that the shortest path weights and the edge weights should satisfy. If a_{ij} is the weight of the edge (i, j), with ⓪ being the weight of non-existent edges, and x_{ij} is the shortest path weight from i to j, then the following equation has to hold (assume there exist n nodes):

$$x_{ij} = \bigoplus_{k=1}^{n} (a_{ik} \otimes x_{kj})$$

For example, when edge weights are transmission delays, this equation becomes:

$$x_{ij} = \min_{1 \le k \le n} (a_{ik} + x_{kj})$$

Note, also, that if $\oplus$ and $\otimes$ are the usual addition and multiplication, respectively, then the first of the above equations becomes exactly matrix multiplication.

$$x_{ij} = \sum_{k=1}^{n} a_{ik} x_{kj} \quad \Leftrightarrow \quad X = AX, X = [x_{ij}]_{n \times n}, A = [a_{ij}]_{n \times n}$$

We will use this fact in a later section to model an existing trust computation algorithm.

3.2.4 Semirings in previous work on trust

Semirings have not been used in the context of trust management, with one exception [4]. In this paper, the aim is to combine access control policies, and come up with one that maximally satisfies the imposed constraints. The problem is seen as a Semiring-based Constraint Satisfaction Problem, in which the constraints are defined over a semiring.

In order to show the modeling power of this framework, we now model PGP's web of trust computations [50] as a semiring. Remember that PGP computes the validity of an alleged key-to-user binding, as seen from the point of view of a particular user, henceforth called the source. The input to the computation algorithm consists of three things: The source node, the graph of certificates issued by users for each other, and the trust values for each user as assigned by the source. Note that the validity of all key-to-user bindings has to be verified, since only certificates signed by valid keys are taken into account, and any certificate may influence the validity of a key-to-user binding.

The validity of the key-to-user binding for user i will be deduced from the

vector $d_i \in \mathbb{N}^k$, where k is the number of different trust levels defined by PGP. It seems that k is 4 ("unknown", "untrusted", "marginally trusted", "fully trusted"), but some include a fifth level : "ultimately trusted". Our analysis is independent of the exact value of k. The vector d_i will hold the number of valid certificates for user i that have been signed by users of each trust level. For example, $d_i = (0, 1, 2, 3)$ means that one "untrusted", two "marginally trusted", and three "fully trusted" users have issued certificates for user i's public key. In addition, all six of these certificates are signed by valid keys, i.e. keys for which the key-to-user binding has been verified.

In order to verify the actual validity of the binding, we will use the function $\texttt{val} : \mathbb{N}^k \rightarrow \mathbb{V}$, where $\mathbb{V}$ is the space of admissible results. For simplicity, we will be assuming that $\mathbb{V} =$ {"invalid", "valid"}, although values such as "marginally valid" have also been proposed. The output of `val` for a specific input is determined by thresholds such as: "A key-to-user binding is valid if at least two "marginally trusted" users have issued a certificate for it". These thresholds are incorporated in `val` and will be transparent to our analysis. Finally, for computation simplicity we will be assuming that $\mathbb{V} = \{0, 1\}$, where "invalid"$= 0$, and "valid"$= 1$.

The edge weights $w_{ij} \in \mathbb{N}^k, 1 \leq i, j \leq n$, where n is the number of users, correspond to the certificate from i about j's alleged public key. A weight can only have one of $k + 1$ possible values. Either it consists only of 0s, or of exactly $k - 1$ 0s and one 1. An all-zero weight means that there is no certificate from i about j's key. An 1 in the position that corresponds to trust level t means that the source has assigned trust level t to i, and i has issued a certificate for j.

The $\otimes$ operator is defined as follows ($a, b \in \mathbb{N}^k$):

$$a \otimes b = \texttt{val}(a)b \in \mathbb{N}^k$$

The $\oplus$ operator is defined exactly as vector addition in $\mathbb{N}^k$.

Verification of the semiring properties

For $\otimes$, the absorbing element is $\textcircled{0} = (0, \ldots, 0) \in \mathbb{N}^k$, and the neutral element is $\textcircled{1} = \{x \in \mathbb{N}^k : \texttt{val}(x) = 1\}$. That is, all such vectors are mapped to $\textcircled{1}$; for our purposes, they are equivalent. It is trivial to prove that $\textcircled{0}$ is a neutral element for $\oplus$.

The $\otimes$ operator is associative:

$$a \otimes (b \otimes c) = a \otimes (\texttt{val}(b)c) = \texttt{val}(a)\texttt{val}(b)c$$

$$(a \otimes b) \otimes c = (\texttt{val}(a)b) \otimes c = \texttt{val}(\texttt{val}(a)b)c$$

and these two are equal because $\texttt{val}(\textcircled{0})$=0.

The $\oplus$ operator is commutative and associative, because it is vector addition.

The $\otimes$ operator distributes over $\oplus$:

$$a \otimes (b \oplus c) = \texttt{val}(a)(b + c)$$

$$(a \otimes b) \oplus (a \otimes c) = \texttt{val}(a)b + \texttt{val}(a)c$$

The following computation algorithm uses the above semiring to compute the validity or otherwise of all keys in the certificate graph G. The source node is s and the function w maps edges to edge weights.

PGP-SEMIRING-CALCULATION(G, w, s)

```
1  for i ← 1 to |V|
2        do d[i] ← ⓪
3  d[s] ← ①
4  S ← {s}
5  while S ≠ ∅
6        do u ← DEQUEUE(S)
7           for each v ∈ Neighbors[u], such that val(d[v]) = 0
8                 do
9                    d[v] ← d[v] ⊕ (d[u] ⊗ w(u, v))
10                   if val(d[v]) = 1
11                      then ENQUEUE(S, v)
```

The computation starts at the source s, and progressively computes the validity of all keys reachable from s in the certificate graph. The queue S contains all valid keys for which the outgoing edges (certificates signed with these keys) have not been examined yet. When a key is extracted from S, its certificates to other keys are examined, and their d-vectors are updated. Only certificates to so-far-invalid keys are examined, since adding a certificate to the d-vector of a key already shown to be valid is redundant. If a so-far-invalid key obtains enough certificates to become valid, it is added to the queue for future examination. Each key is enqueued at most once (when it becomes valid), and all keys in the queue are eventually dequeued. Ergo, the algorithm terminates. After termination, all valid keys have been discovered.

Note that if s is only interested in the validity of a particular key-to-user binding, then the algorithm can stop earlier: as soon as its validity is determined, or after all certificates for that key have been examined.

We can also model the EigenTrust algorithm [26] as a semiring. Using the system of linear equations interpretation of a semiring, the EigenTrust algorithm solves the following matrix equation for T:

$$T = CT \quad \Leftrightarrow \quad t_{ij} = \sum_{k=1}^{n} c_{ik} t_{kj}$$

where the semiring operators are the usual addition and multiplication.

3.3 Trust Semiring

3.3.1 Intuitive Requirements

Based on intuitive concepts about trust establishment, we can expect the binary operators to have certain properties in addition to those required by the semiring structure.

Since an opinion should deteriorate along a path, we require the following for the $\otimes$ operator ($a, b \in S$):

$$a \otimes b \preceq a, b$$

where $\preceq$ is the difference relation defined in Section 3.2. Note that the total opinion along a path is "limited" by the source's opinion for the first node in the path.

The element ⓪ (neutral element for $\oplus$, absorbing for $\otimes$) is the set of opinions (t, ZERO_CONF), for any $t \in [0, 1]$, which, in essence, corresponds to non-existent trust relations between nodes. The motivation is that if a ⓪ is encountered along

a path, then the whole path "through" this opinion should have zero confidence. Also, such opinions should be ignored in $\oplus$-sums.

The element ① (neutral element for $\otimes$) is the "best" opinion that can be assigned to a node: (MAX_TRUST, MAX_CONF). This can be seen as the opinion of a node about itself. Also, it is the desirable point of convergence of the opinions of all good nodes about all other good nodes in the classification example. If encountered along a path, ① effectively contracts the corresponding edge and identifies the nodes at its endpoints for the purposes of the aggregation.

Regarding aggregation across paths with the $\oplus$ operator, we generally expect that opinion quality will improve, since we have multiple opinions. If the opinions disagree, the more confident one will weigh heavier. In a fashion similar to the $\otimes$ operator, we require that the $\oplus$ operator satisfies ($a, b \in S$):

$$a \oplus b \succeq a, b$$

3.3.2 Path semiring

In this semiring, the opinion space is $S = [0,1] \times [0,1]$ Our choice for the $\otimes$ and $\oplus$ operators is as follows (Figure 3.3):

$$(t_{ik}, c_{ik}) \otimes (t_{kj}, c_{kj}) = (t_{ik}t_{kj}, c_{ik}c_{kj}) \tag{3.1}$$

$$(t_{ij}^{p_1}, c_{ij}^{p_1}) \oplus (t_{ij}^{p_2}, c_{ij}^{p_2}) = \begin{cases} (t_{ij}^{p_1}, c_{ij}^{p_1}) & \text{if } c_{ij}^{p_1} > c_{ij}^{p_2} \\ (t_{ij}^{p_2}, c_{ij}^{p_2}) & \text{if } c_{ij}^{p_1} < c_{ij}^{p_2} \\ (\max(t_{ij}^{p_1}, t_{ij}^{p_2}), c_{ij}) & \text{if } c_{ij}^{p_1} = c_{ij}^{p_2} = c_{ij} \end{cases}, \tag{3.2}$$

where $(t_{ij}^{p_1}, c_{ij}^{p_1})$ is the opinion that i has formed about j along the path p_1.

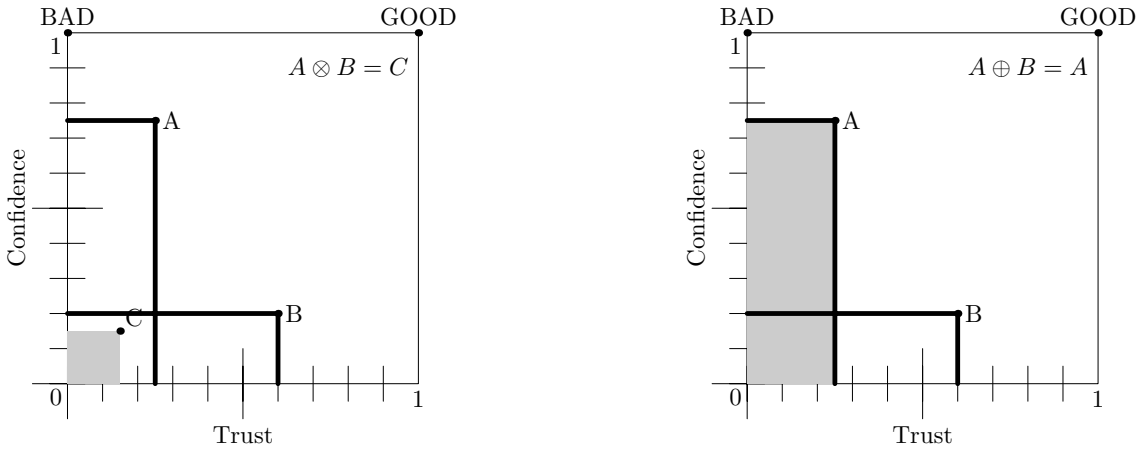

Figure 3.3: $\otimes$ and $\oplus$ operators for the Path semiring

Since both the trust and the confidence values are in the $[0, 1]$ interval, they both decrease when aggregated along a path. When opinions are aggregated across paths, the one with the highest confidence prevails. If the two opinions have equal confidences but different trust values, we pick the one with the highest trust value. We could have also picked the lowest trust value; the choice depends on the desired semantics of the application.

This semiring essentially computes the trust distance along the most confident trust path to the destination. An important feature is that this distance is computed along a single path, since the $\oplus$ operator picks exactly one path. Other paths are ignored, so not all available information is being taken into account. One of the advantages is that if the trust value turns out to be high, then a trusted path to the destination has also been discovered. Also, fewer messages are exchanged for information gathering.

Verification of the semiring properties

The neutral elements for this semiring are: $\textcircled{0} = (t, 0)$, for any t, and $\textcircled{1} = (1, 1)$. We can verify this by direct substitution.

The $\otimes$ operator is both associative and commutative, since the underlying multiplication operator is. The $\oplus$ operator also has both of these properties, since it picks the opinion with the highest confidence. So, it is essentially equivalent to a max operation. The distributivity of $\otimes$ over $\oplus$ is proven as follows (We can assume, without loss of generality, that $c_{kj}^{p_1} > c_{kj}^{p_2}$, or $c_{kj}^{p_1} = c_{kj}^{p_2}$ and $t_{kj}^{p_1} > t_{kj}^{p_2}$):

$$\begin{aligned}
(t_{ik}, c_{ik}) \otimes \left((t_{kj}^{p_1}, c_{kj}^{p_1}) \oplus (t_{kj}^{p_2}, c_{kj}^{p_2})\right) &= (t_{ik}, c_{ik}) \otimes (t_{kj}^{p_1}, c_{kj}^{p_1}) \\
&= (t_{ik} t_{kj}^{p_1}, c_{ik} c_{kj}^{p_1}) \\
\left((t_{ik}, c_{ik}) \otimes (t_{kj}^{p_1}, c_{kj}^{p_1})\right) \oplus \left((t_{ik}, c_{ik}) \otimes (t_{kj}^{p_2}, c_{kj}^{p_2})\right) &= (t_{ik} t_{kj}^{p_1}, c_{ik} c_{kj}^{p_1}) \oplus (t_{ik} t_{kj}^{p_2}, c_{ik} c_{kj}^{p_2}) \\
&= (t_{ik} t_{kj}^{p_1}, c_{ik} c_{kj}^{p_1})
\end{aligned}$$

So, we have proven that:

$$(t_{ik}, c_{ik}) \otimes \left((t_{kj}^{p_1}, c_{kj}^{p_1}) \oplus (t_{kj}^{p_2}, c_{kj}^{p_2})\right) = \left((t_{ik}, c_{ik}) \otimes (t_{kj}^{p_1}, c_{kj}^{p_1})\right) \oplus \left((t_{ik}, c_{ik}) \otimes (t_{kj}^{p_2}, c_{kj}^{p_2})\right)$$

3.3.3 Distance semiring

Our second choice is a semiring based on the *Expectation semiring* defined by Eisner in [15], and used for speech/language processing:

$$\begin{aligned}
(a_1, b_1) \otimes (a_2, b_2) &= (a_1 b_2 + a_2 b_1, b_1 b_2) \\
(a_1, b_1) \oplus (a_2, b_2) &= (a_1 + a_2, b_1 + b_2)
\end{aligned}$$

The opinion space is $S = [0, \infty] \times [0, 1]$. Before using this semiring, the pair (trust, confidence)=(t, c) is mapped to the weight $(c/t, c)$. The motivation for this mapping becomes clear when we describe its effect on the results of the operators. The binary operators are then applied to this weight, and the result is mapped back to a (trust, confidence) pair.

$$
\begin{aligned}
(t_{ik}, c_{ik}) \otimes (t_{kj}, c_{kj}) &\rightarrow \left(\frac{c_{ik}}{t_{ik}}, c_{ik}\right) \otimes \left(\frac{c_{kj}}{t_{kj}}, c_{kj}\right) = \left(\frac{c_{ik}c_{kj}}{t_{ik}} + \frac{c_{ik}c_{kj}}{t_{kj}}, c_{ik}c_{kj}\right) \\
&\rightarrow \left(\frac{1}{\frac{1}{t_{ik}} + \frac{1}{t_{kj}}}, c_{ik}c_{kj}\right) \\
\left(t_{ij}^{p_1}, c_{ij}^{p_1}\right) \oplus \left(t_{ij}^{p_2}, c_{ij}^{p_2}\right) &\rightarrow \left(\frac{c_{ij}^{p_1}}{t_{ij}^{p_1}}, c_{ij}^{p_1}\right) \oplus \left(\frac{c_{ij}^{p_2}}{t_{ij}^{p_2}}, c_{ij}^{p_2}\right) = \left(\frac{c_{ij}^{p_1}}{t_{ij}^{p_1}} + \frac{c_{ij}^{p_2}}{t_{ij}^{p_2}}, c_{ij}^{p_1} + c_{ij}^{p_2}\right) \\
&\rightarrow \left(\frac{c_{ij}^{p_1} + c_{ij}^{p_2}}{\frac{c_{ij}^{p_1}}{t_{ij}^{p_1}} + \frac{c_{ij}^{p_2}}{t_{ij}^{p_2}}}, c_{ij}^{p_1} + c_{ij}^{p_2}\right)
\end{aligned}
$$

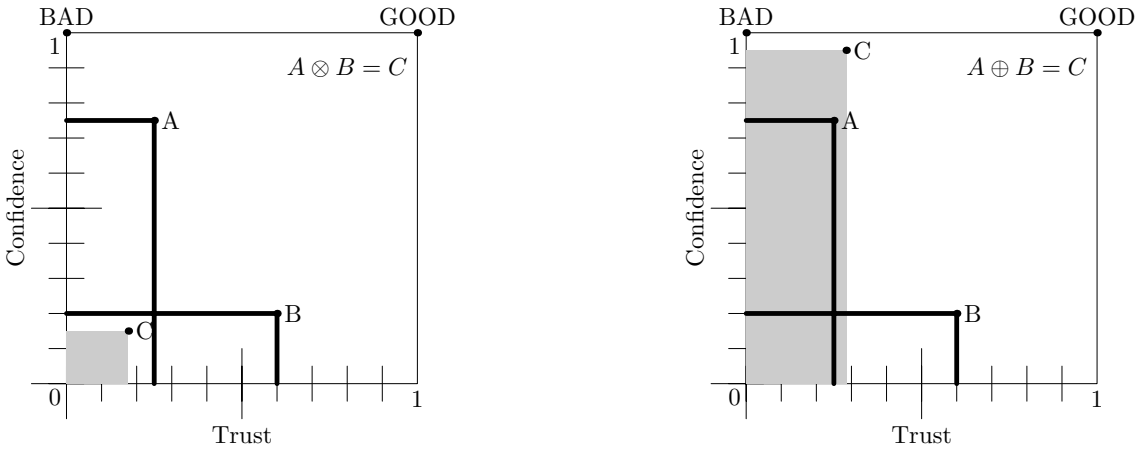

Figure 3.4: $\otimes$ and $\oplus$ operators for the Distance semiring

So, when aggregating along a path, both the trust and the confidence decrease. The component trust values are combined like parallel resistors. We can see here the effect of the mapping: Two resistors in parallel offer lower resistance than either of them in isolation. Also, a zero trust value in either opinion will result in a zero trust value in the resulting opinion (absorbing element), while a trust value equal to

infinity will cause the corresponding opinion to disappear from the result (neutral element). On the other hand, the component confidence values are between 0 and 1, and they are multiplied, so the resulting confidence value is smaller than both.

When aggregating across paths, the total trust value is the weighted harmonic average of the components, with weights according to their confidence values. So, the result is between the two component values, but closer to the more confident one. Again we can see the effect of the mapping: The weighted harmonic average outcome is a direct result of the inverse mapping. Note, also, the behavior caused by extreme (zero or infinity) trust values: A zero trust value dominates the result (unless its corresponding confidence is zero); a trust value equal to infinity results in an increase in the trust value given by the other opinion. In order for the resulting trust value to be the maximum possible, both opinions have to assign the maximum. So, in general, we can say that this operator is conservative. A zero confidence value (neutral element) causes the corresponding opinion to disappear from the result.

Verification of the semiring properties

The neutral elements are: $\textcircled{0} = (0, 0)$ and $\textcircled{1} = (0, 1)$, which we can verify by direct substitution.

Because of their symmetry, both operators are commutative. The $\oplus$ operator is trivially associative, and here is the proof for the associativity of $\otimes$:

$$\begin{aligned} ((a_1, b_1) \otimes (a_2, b_2)) \otimes (a_3, b_3) &= (a_1 b_2 + a_2 b_1, b_1 b_2) \otimes (a_3, b_3) \\ &= (a_1 b_2 b_3 + a_2 b_1 b_3 + a_3 b_1 b_2, b_1 b_2 b_3) \end{aligned}$$

$$
\begin{aligned}
(a_1, b_1) \otimes ((a_2, b_2) \otimes (a_3, b_3)) &= (a_1, b_1) \otimes (a_2 b_3 + a_3 b_2, b_2 b_3) \\
&= (a_1 b_2 b_3 + a_2 b_1 b_3 + a_3 b_1 b_2, b_1 b_2 b_3)
\end{aligned}
$$

We now prove that $\otimes$ distributes over $\oplus$:

$$
\begin{aligned}
(a_1, b_1) \otimes ((a_2, b_2) \oplus (a_3, b_3)) &= (a_1, b_1) \otimes (a_2 + a_3, b_2 + b_3) \\
&= (a_1(b_2 + b_3) + b_1(a_2 + a_3), b_1(b_2 + b_3)) \\
((a_1, b_1) \otimes (a_2, b_2)) \oplus ((a_1, b_1) \otimes (a_3, b_3)) &= (a_1 b_2 + a_2 b_1, b_1 b_2) \oplus (a_1 b_3 + a_3 b_1, b_1 b_3) \\
&= (a_1 b_2 + a_2 b_1 + a_1 b_3 + a_3 b_1, b_1 b_2 + b_1 b_3)) \\
&= (a_1(b_2 + b_3) + b_1(a_2 + a_3), b_1(b_2 + b_3))
\end{aligned}
$$

3.3.4 Computation algorithm

The following algorithm, due to Mohri [38], computes the $\oplus$-sum of all path weights from a designated node s to all other nodes in the trust graph $G = (V, E)$.

GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE(G, s)

```
1  for i ← 1 to |V|
2        do d[i] ← r[i] ← ⓪
3  d[s] ← r[s] ← ①
4  S ← {s}
5  while S ≠ ∅
6        do q ← head(S)
7           DEQUEUE(S)
8           r' ← r[q]
9           r[q] ← ⓪
10          for each v ∈ Neighbors[q]
11                do if d[v] ≠ d[v] ⊕ (r' ⊗ w[(q, v)])
12                   then d[v] ← d[v] ⊕ (r' ⊗ w[(q, v)])
13                        r[v] ← r[v] ⊕ (r' ⊗ w[(q, v)])
14                        if v ∉ S
15                           then ENQUEUE(S, v)
16 d[s] ← ①
```

This is an extension to Dijkstra's algorithm [14]. S is a queue that contains the vertices to be examined next for their contribution to the shortest path weights. The vector $d[i], i \in V$ holds the current estimate of the shortest distance from s to i. The vector $r[i], i \in V$ holds the total weight added to $d[i]$ since the last time i was extracted from S. This is needed for non-idempotent semirings, such as the second one proposed.

Our computation algorithm is based on Mohri's, but with three adjustments which are needed when considering the problem from the perspective of trust. Lines 11-13 of the algorithm will be referred to as "node q votes for node v".

First of all, some nodes may be prevented from voting. Only if a node's trust value exceeds a predefined trust threshold, is the node allowed to vote. This is motivated from the common sense observation that only good nodes should participate in the computation, and bad nodes should be barred. Note that there is no restriction on the corresponding confidence. This will initially lead to bad nodes being allowed to vote, but after some point they will be excluded since good nodes will acquire evidence for their maliciousness.

Second, no node is allowed to vote for the source (s). Since it is s that initiates the computation, it does not make sense to compute s's opinion for itself.

Third, no cyclic paths are taken into account. If that were the case, we would be allowing a node to influence the opinion about itself, which is undesirable. Unfortunately, there is no clear way to discard any single edge-opinion of the cycle. So, the approach taken is to discard any edges that would form a cycle if accepted. As a result, the order in which the voters are chosen in line 6 is important. We argue that it makes sense to choose the node for which the confidence is highest.

Note that these adjustments introduce characteristics from the Path semiring into the Distance semiring. For example, the node with the maximum confidence gets to vote first. Moreover, some paths are pruned which means that fewer messages are exchanged, thus saving bandwidth, but also some of the existing information is not taken into account. In general, this combination of the two semirings seems to

be a good tradeoff between the two.

Chapter 4

Evaluation and Experimental Results

In this chapter, we are describing the scenarios that were examined in the simulations. The results obtained are discussed, and explained in terms of the parameters and properties of the algorithms.

4.1 Good and Bad Nodes

We assume that some nodes are Good, and some are Bad. Good nodes adjust their direct opinions (opinions for their neighbors) according to some predefined rules (explained in Section 4.2). Bad nodes, however, always have the best opinion $(1, 1)$ for their neighboring Bad nodes, and the worst opinion $(0, 1)$ for their neighboring Good nodes.

We expect that the opinions of a Good node for all other nodes would evolve as in Figure 4.1. That is, all Good and all Bad nodes will be identified as Good and Bad, respectively.

4.2 Simulation details

When the network is "born", the nodes are partitioned into Good and Bad. We pick a Good node, which will be computing indirect opinions to all other nodes. Initial direct opinions are all set to $(0.5, 0.1)$, i.e. medium trust and low confidence. The trust threshold, which decides which nodes are allowed to vote, is empirically set to 0.3. Time is discrete and is measured in rounds. At each round, two things happen. First, the direct opinions of each node for his neighbors approach the correct

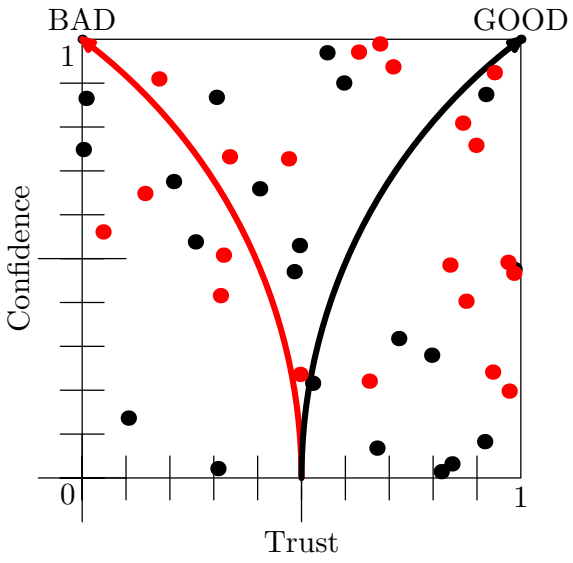

Figure 4.1: Opinion convergence. Opinions for good nodes are black, opinions for bad nodes are red.

opinion, which is $(0, 1)$ for the bad neighbors, and $(1, 1)$ for the good neighbors. Second, the designated good node calculates his indirect opinions for all other nodes. These indirect opinions are the experimental results shown in Section 4.3. Also, the confidence for some indirect opinions may be too low (within $\epsilon = 0.01$ of zero), so these nodes are not assigned any opinion.

The most important evaluation metric is whether the nodes are correctly classified as good and bad. In other words, we want the opinions for all bad nodes to be close to $(0, 1)$ and the opinions for all good nodes close to $(1, 1)$. Moreover, we want this to happen as soon as possible, i.e. before all direct opinions converge to the correct ones, since the users in the real network may be forced to make an early trust decision. Furthermore, a failsafe is desirable: If trust evidence is insufficient, we prefer not to make any decision about a node, rather than make a wrong one.

Of course, we have to evaluate the robustness of each of the above mentioned metrics as the proportion of bad nodes increases. We also measure the effect of different trust topologies. Namely, three topologies are selected: *Grid*, *Random*, and *Small World*. The *Grid* and *Random* topologies can be seen as two extremes of a spectrum. On the one hand, the Grid is completely symmetric and deterministic: We are using a 10x10 square for 100 nodes. Each node, except the perimeter nodes, has exactly 8 neighbors. On the other hand, the Random topology was constructed so that the average degree is again 8, but this symmetry is completely probabilistic. Each edge has the same probability of existing, according to the Erdös-Rényi model [16]. The Small World topology [45] is between these two extremes, in the sense that there are a few nodes that have a high degree, and all the rest have much fewer neighbors. In this case, too, the average degree is 8. The Small World topology for trust has also been used in [23].

4.3 Results

In this section we present the results obtained from the simulations. For each of the three topologies (Grid, Random, Small World), the percentage of bad nodes is increased from 10% to 50% to 90%. The figures show the opinions of the source node (s) for every other node after the computations of rounds $10, 20, ..., 90, 95, 99$. The nodes originally designated as Good appear in black, whereas the Bad ones appear in red. The aim is, first and foremost, for the black nodes to be separated from the red ones. Also, the black nodes should be as close as possible to the upper right corner (GOOD corner, corresponding to the $(1, 1)$ opinion), and the red nodes

to the upper left corner (BAD corner, $(0, 1)$ opinion).

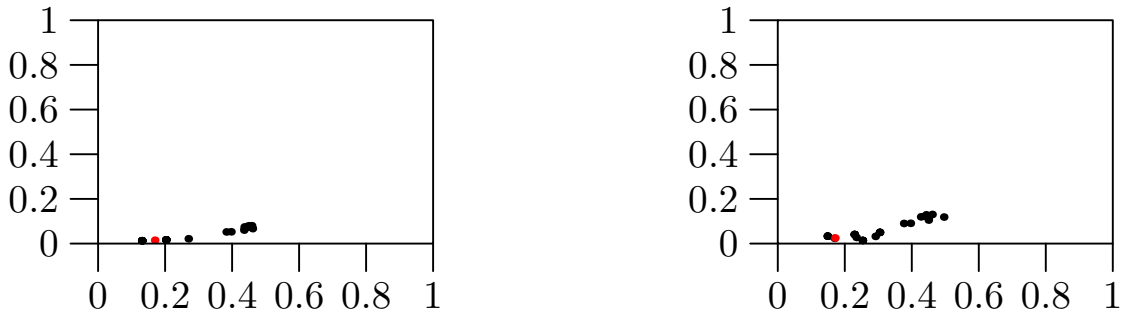

Figure 4.2: Grid:bad1:Rounds(10-20)

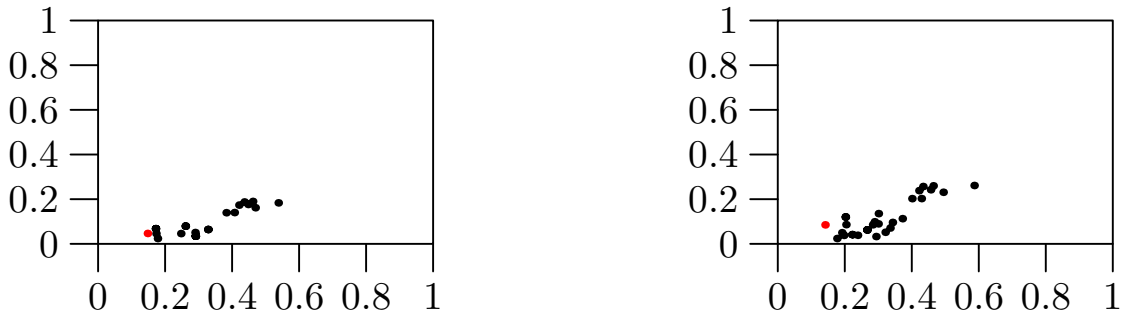

Figure 4.3: Grid:bad1:Rounds(30-40)

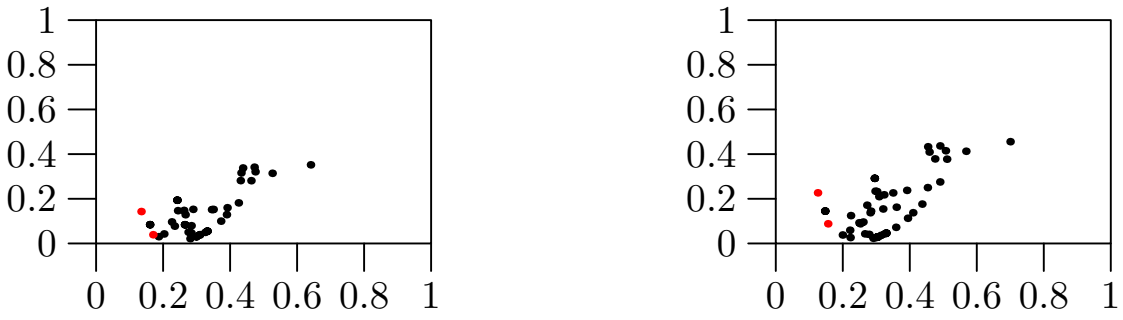

Figure 4.4: Grid:bad1:Rounds(50-60)

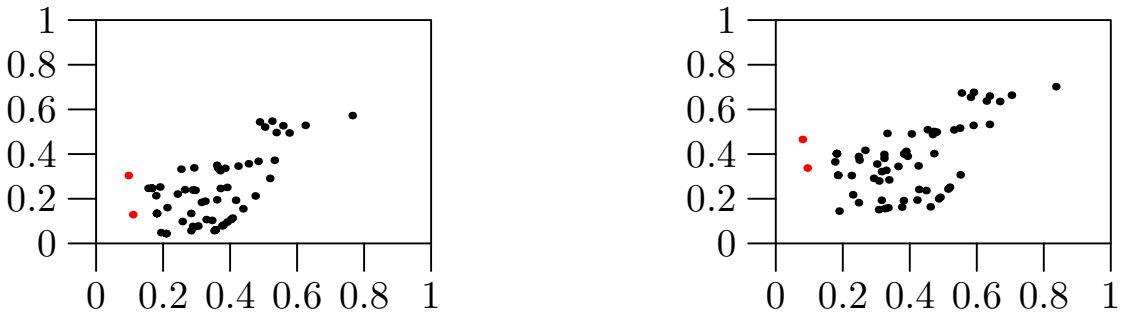

Figure 4.5: Grid:bad1:Rounds(70-80)

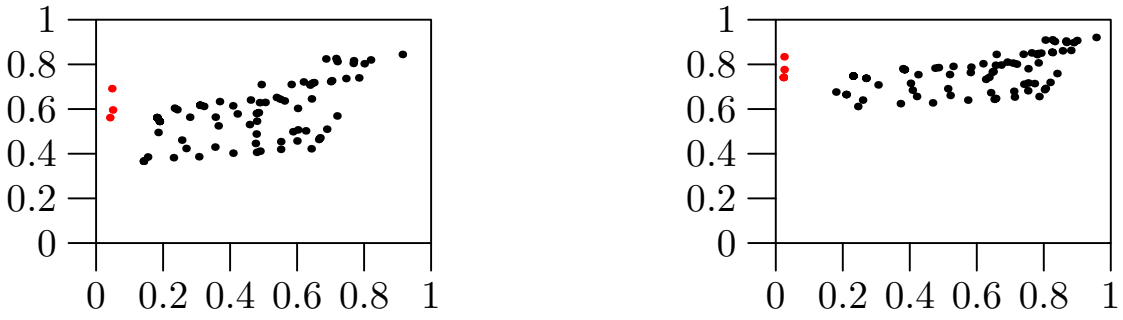

Figure 4.6: Grid:bad1:Rounds(90-95)

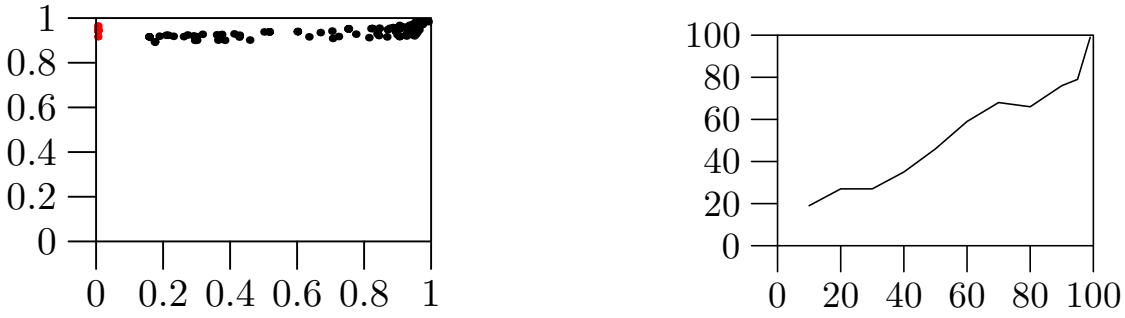

Figure 4.7: Grid:bad1:Round(99)AndClassifiedNodes

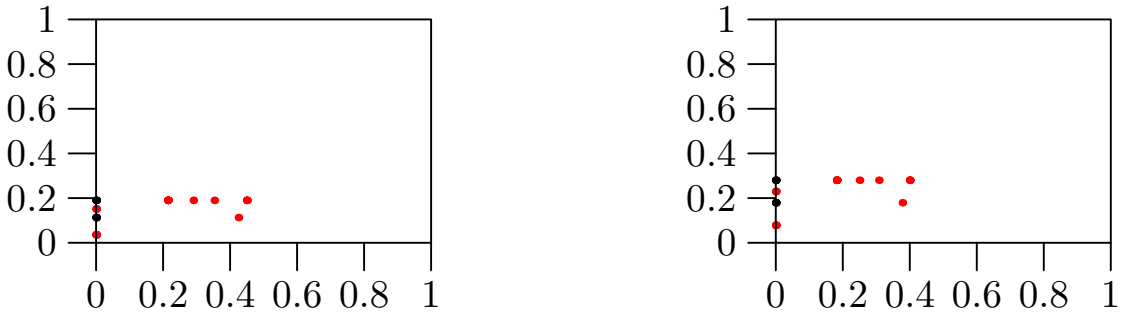

Figure 4.8: Grid:bad5:Rounds(10-20)

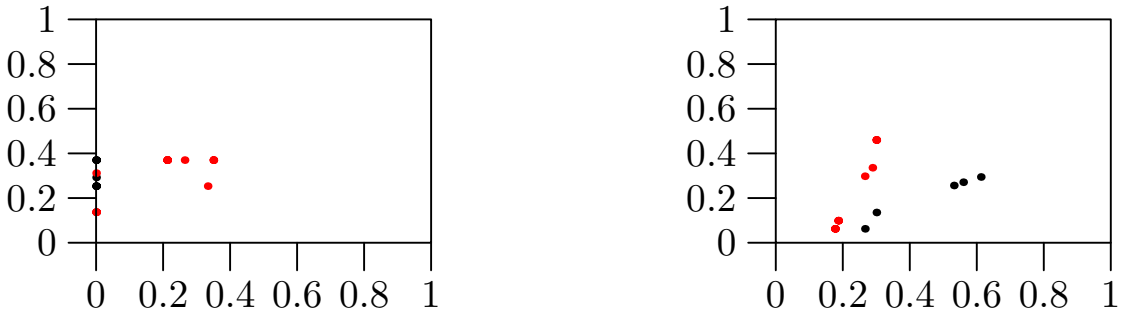

Figure 4.9: Grid:bad5:Rounds(30-40)

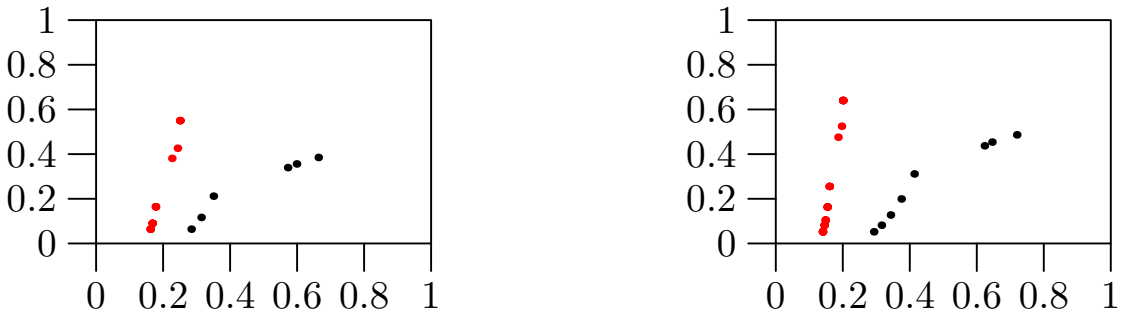

Figure 4.10: Grid:bad5:Rounds(50-60)

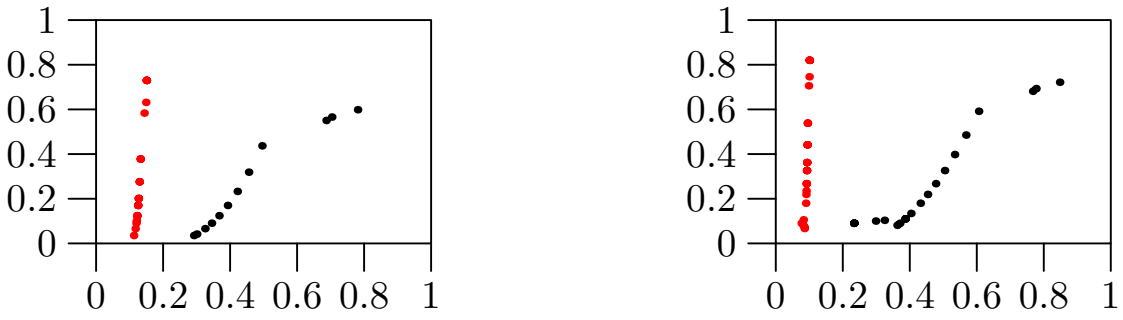

Figure 4.11: Grid:bad5:Rounds(70-80)

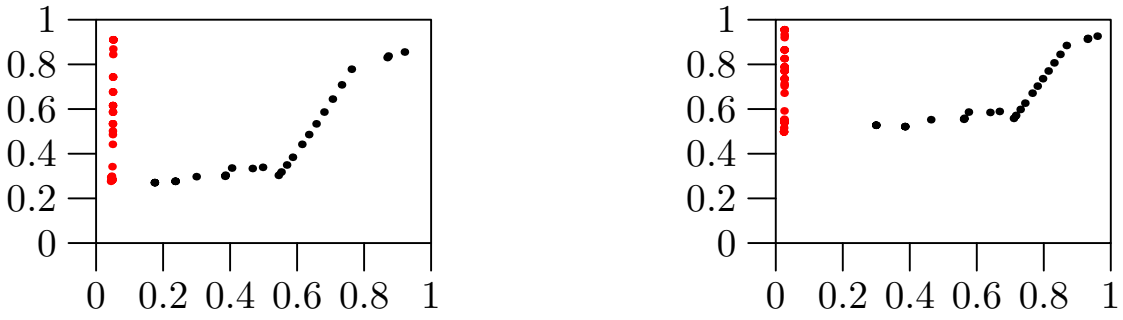

Figure 4.12: Grid:bad5:Rounds(90-95)

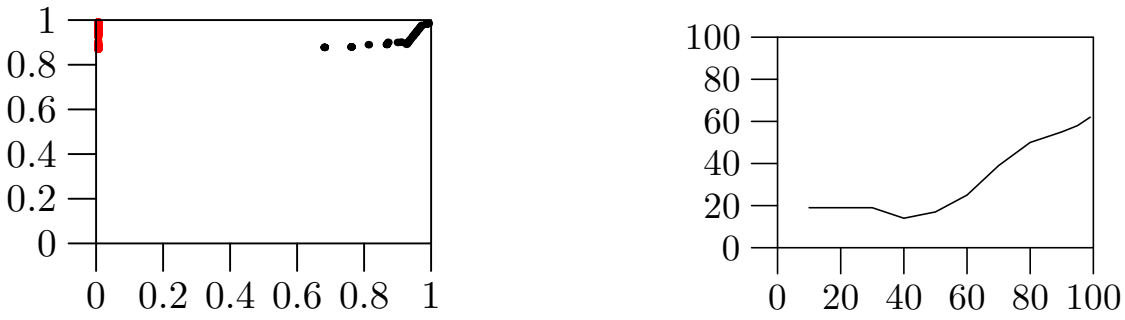

Figure 4.13: Grid:bad5:Round(99)AndClassifiedNodes

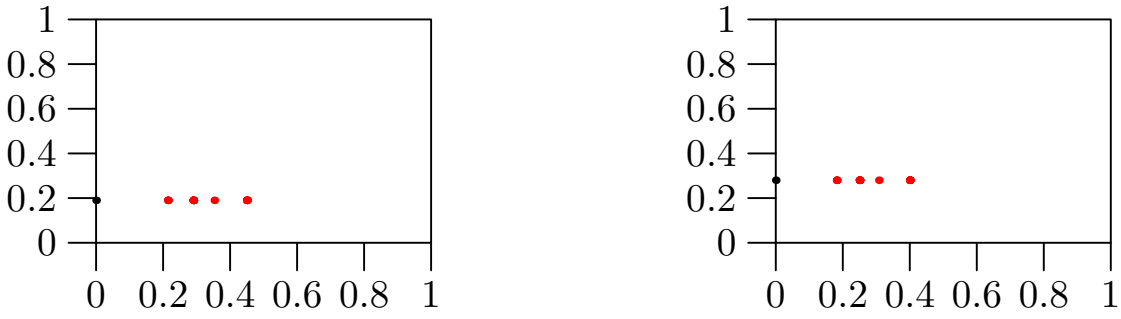

Figure 4.14: Grid:bad9:Rounds(10-20)

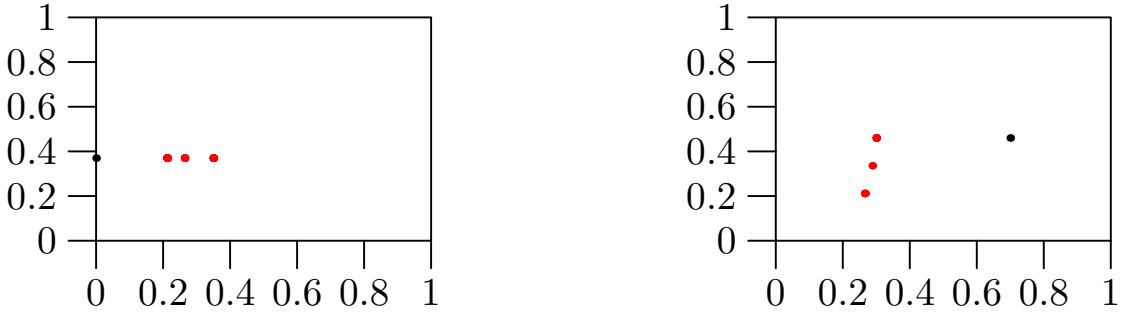

Figure 4.15: Grid:bad9:Rounds(30-40)

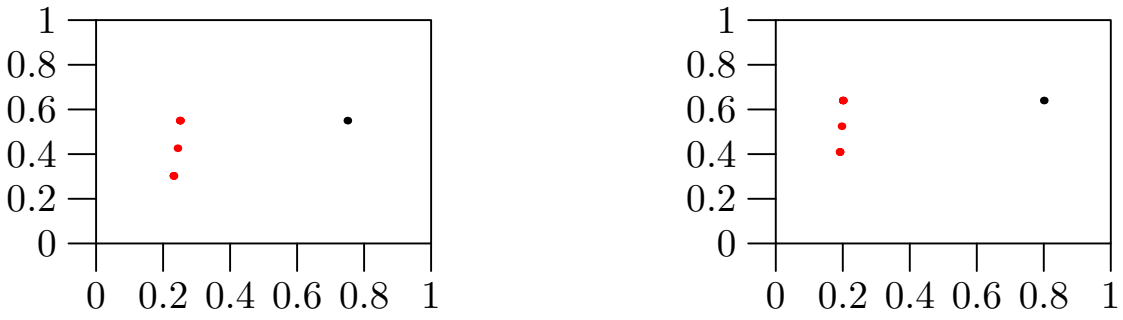

Figure 4.16: Grid:bad9:Rounds(50-60)

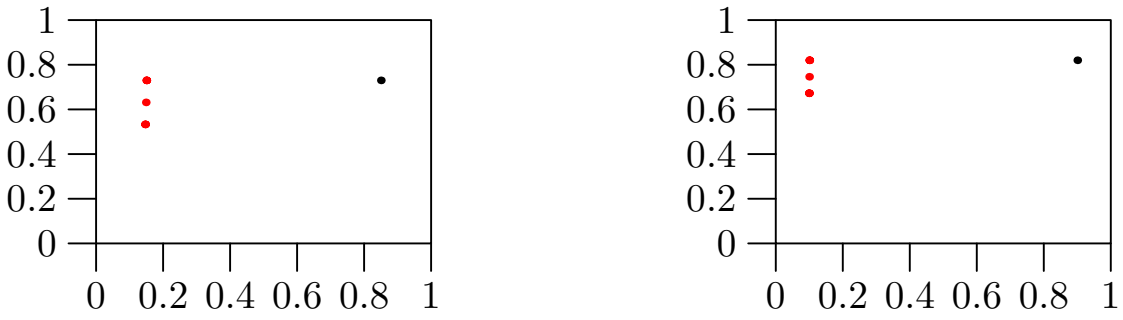

Figure 4.17: Grid:bad9:Rounds(70-80)

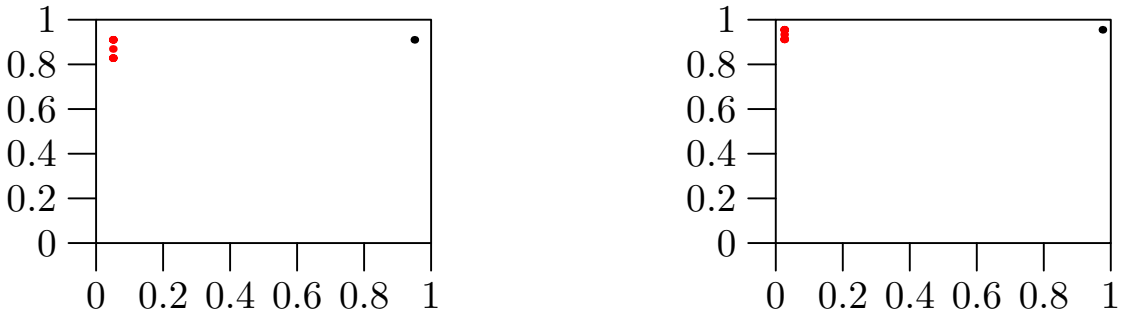

Figure 4.18: Grid:bad9:Rounds(90-95)

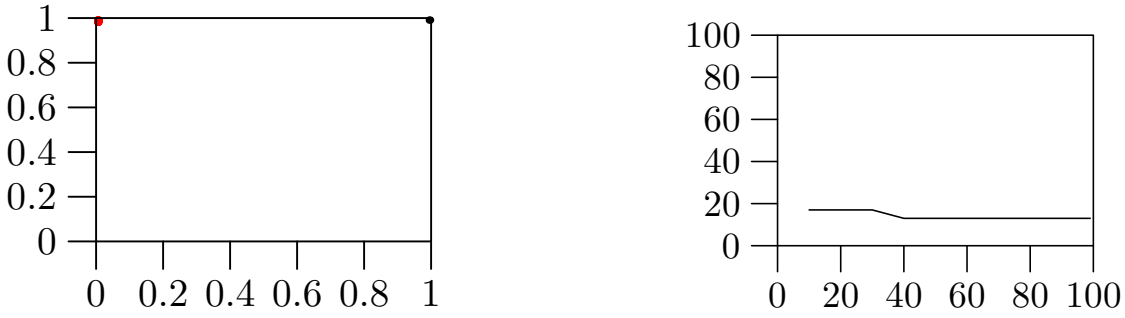

Figure 4.19: Grid:bad9:Round(99)AndClassifiedNodes

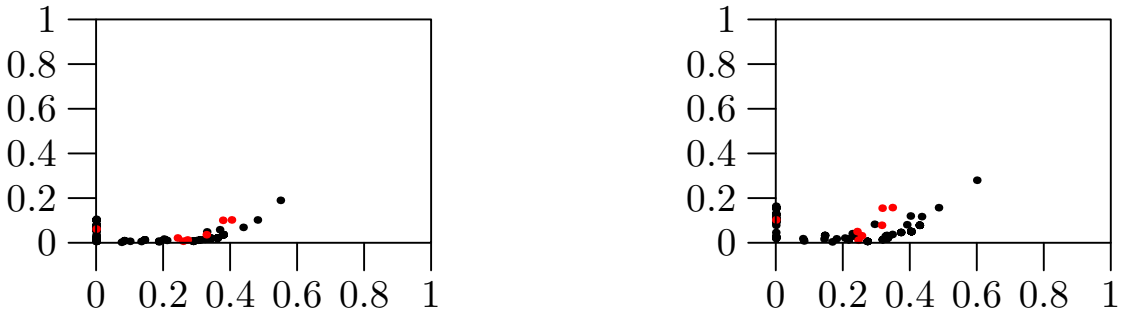

Figure 4.20: SmallWorld:bad1:Rounds(10-20)

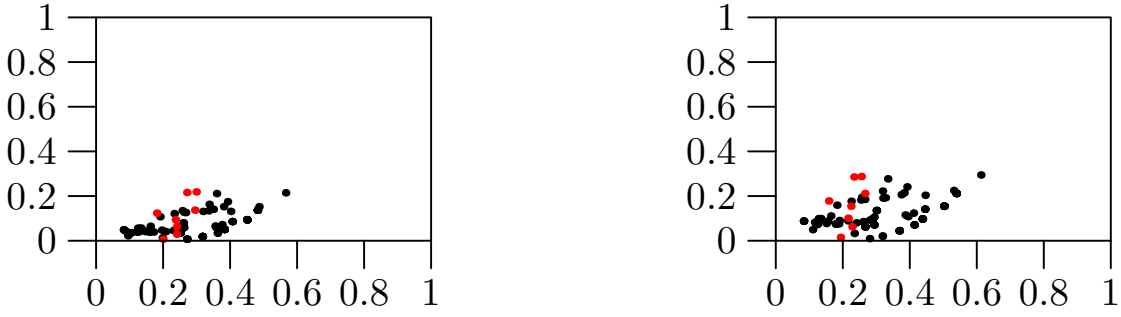

Figure 4.21: SmallWorld:bad1:Rounds(30-40)

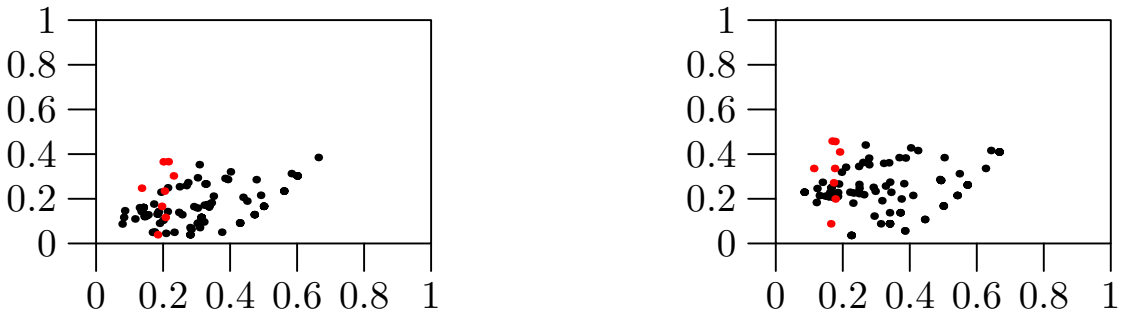

Figure 4.22: SmallWorld:bad1:Rounds(50-60)

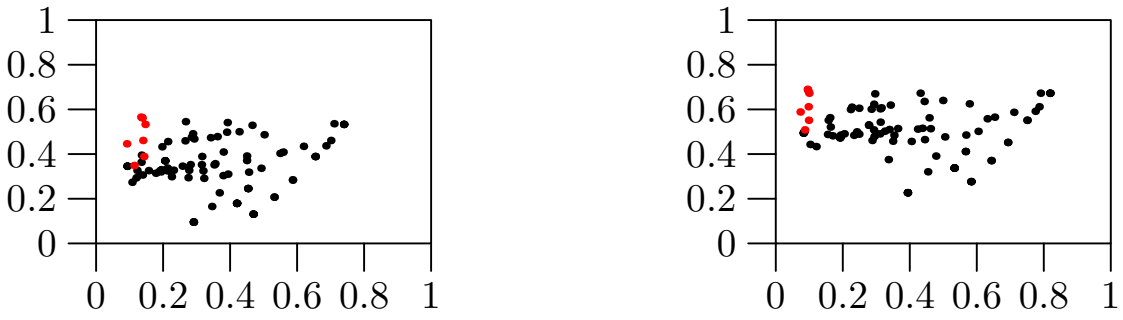

Figure 4.23: SmallWorld:bad1:Rounds(70-80)

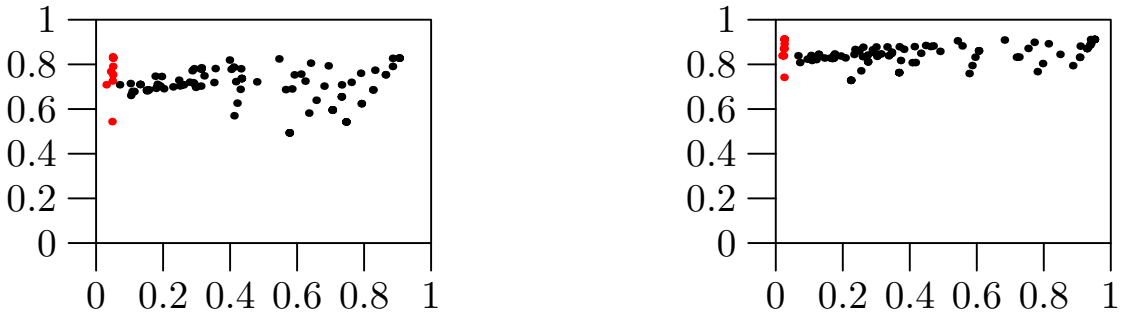

Figure 4.24: SmallWorld:bad1:Rounds(90-95)

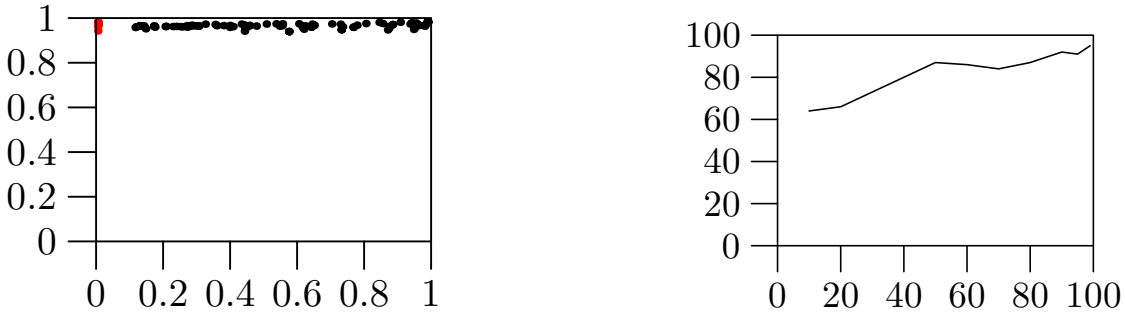

Figure 4.25: SmallWorld:bad1:Round(99)AndClassifiedNodes

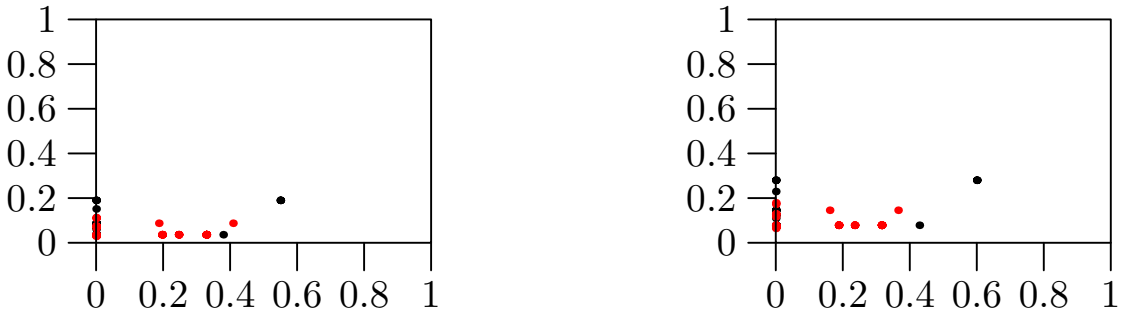

Figure 4.26: SmallWorld:bad5:Rounds(10-20)

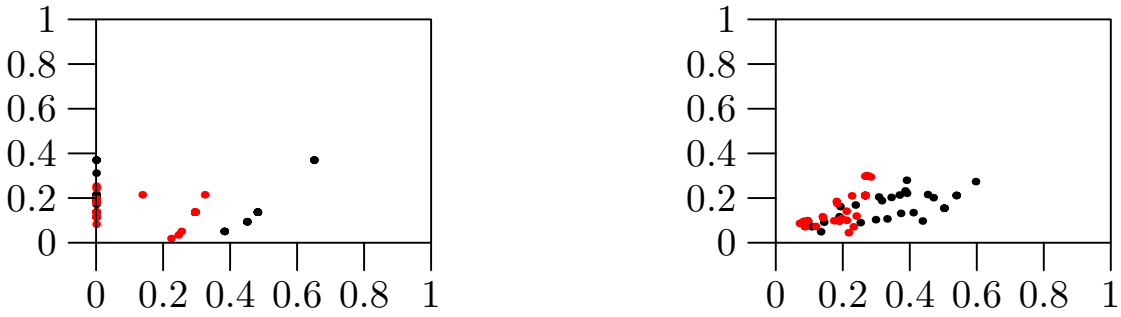

Figure 4.27: SmallWorld:bad5:Rounds(30-40)

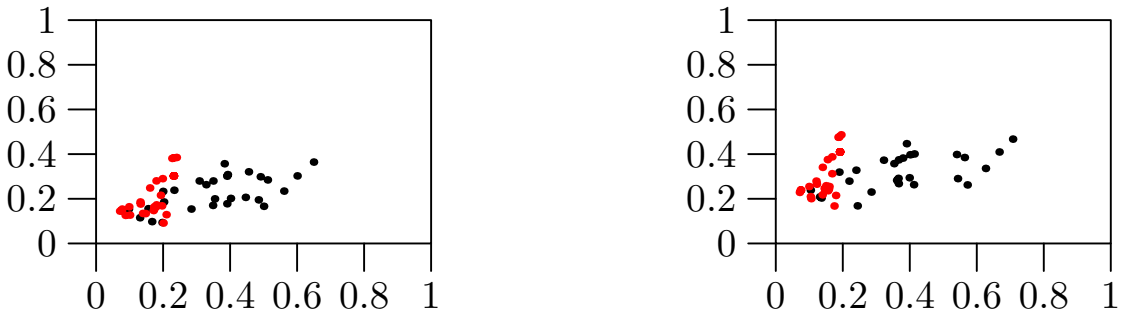

Figure 4.28: SmallWorld:bad5:Rounds(50-60)

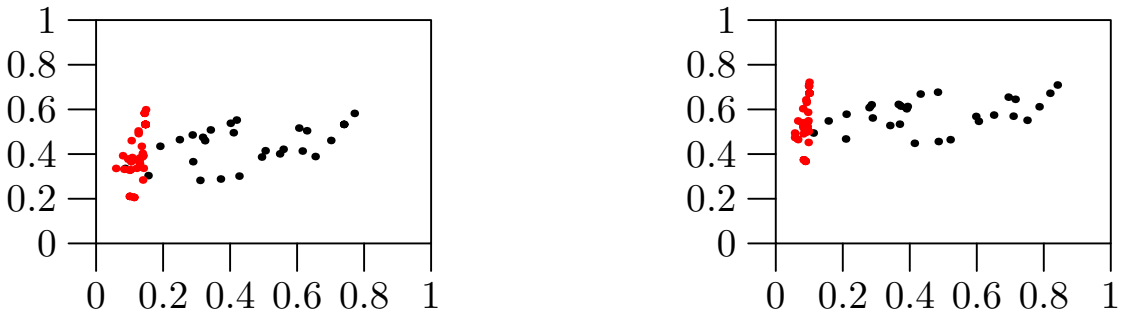

Figure 4.29: SmallWorld:bad5:Rounds(70-80)

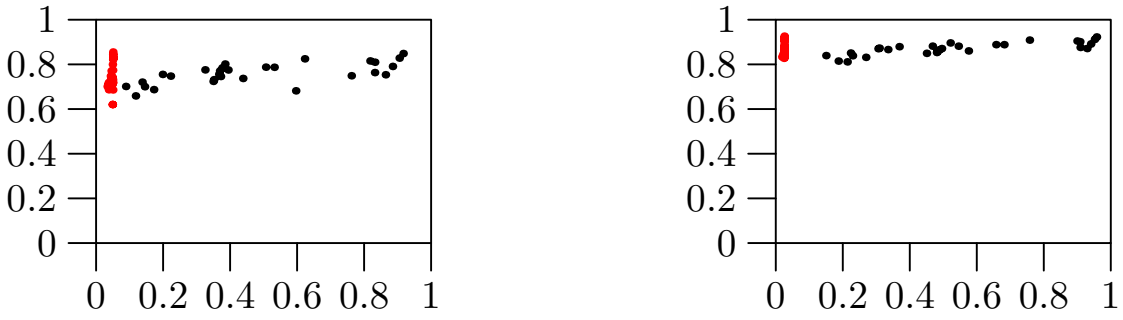

Figure 4.30: SmallWorld:bad5:Rounds(90-95)

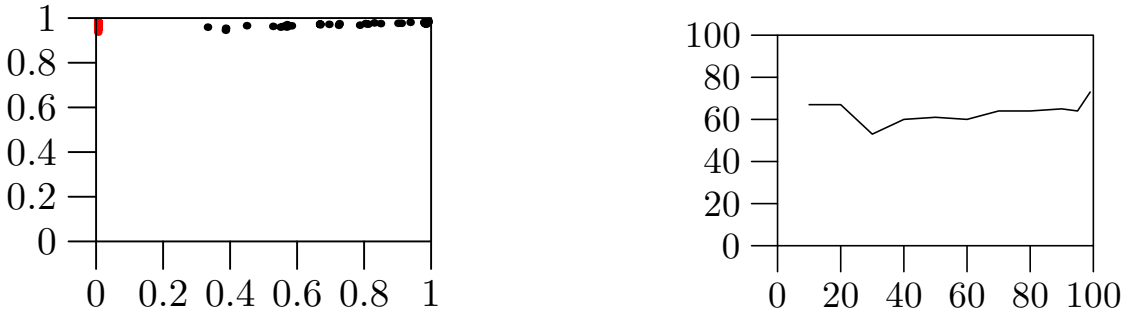

Figure 4.31: SmallWorld:bad5:Round(99)AndClassifiedNodes

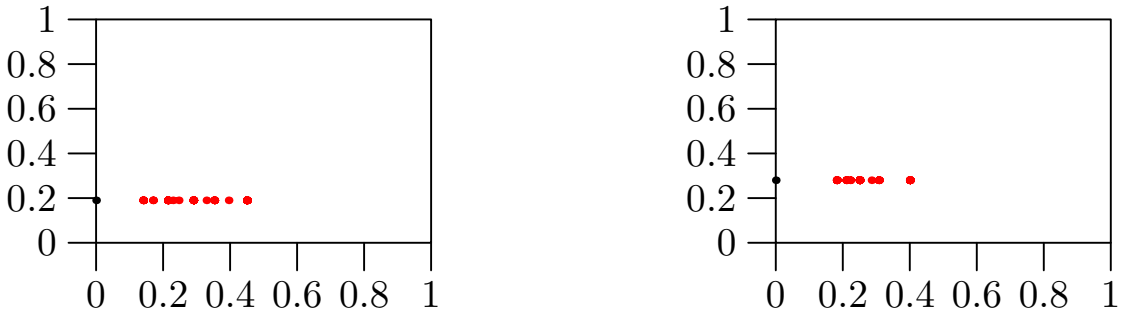

Figure 4.32: SmallWorld:bad9:Rounds(10-20)

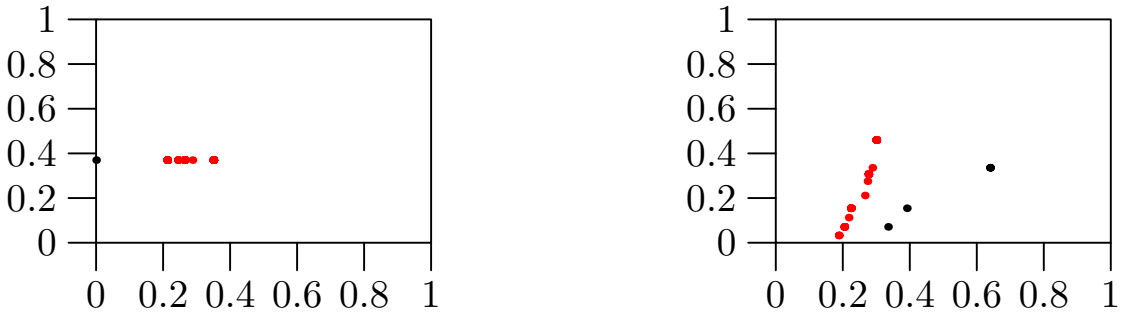

Figure 4.33: SmallWorld:bad9:Rounds(30-40)

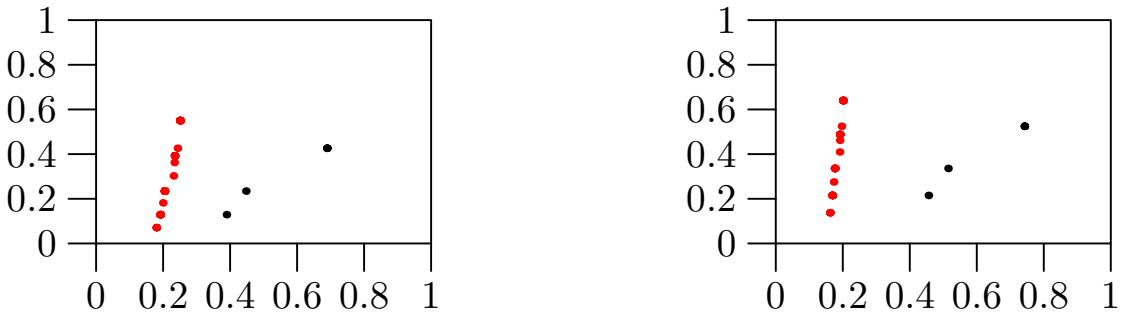

Figure 4.34: SmallWorld:bad9:Rounds(50-60)

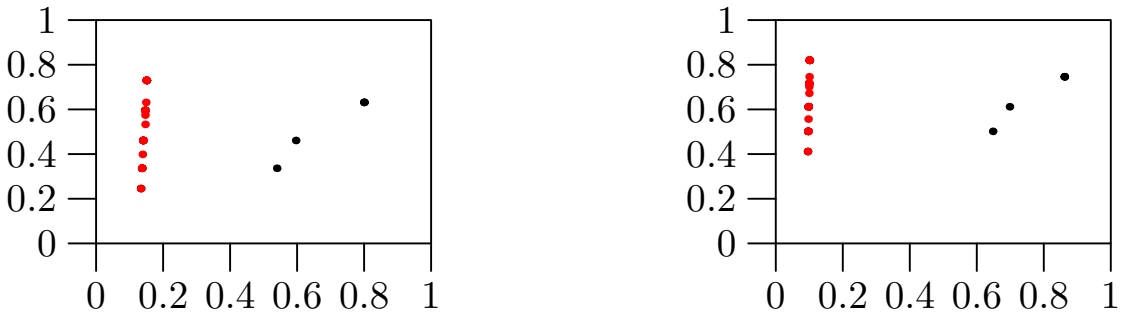

Figure 4.35: SmallWorld:bad9:Rounds(70-80)

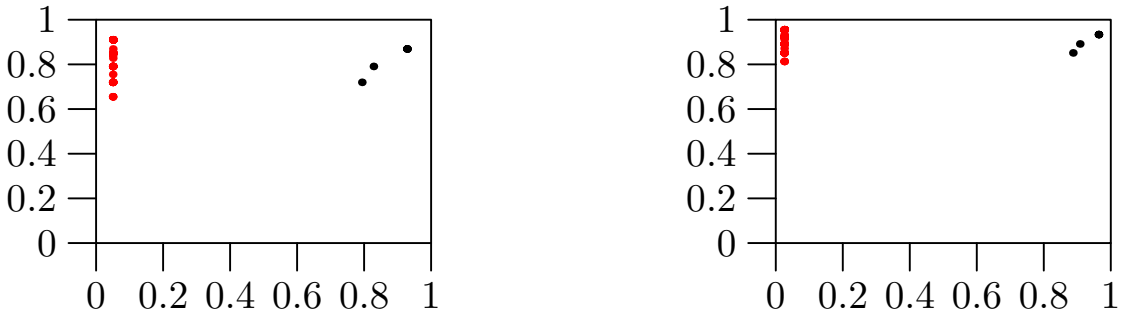

Figure 4.36: SmallWorld:bad9:Rounds(90-95)

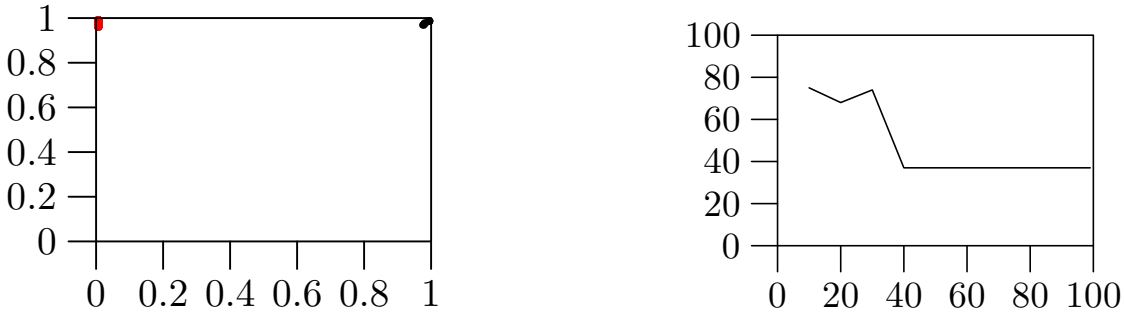

Figure 4.37: SmallWorld:bad9:Round(99)AndClassifiedNodes

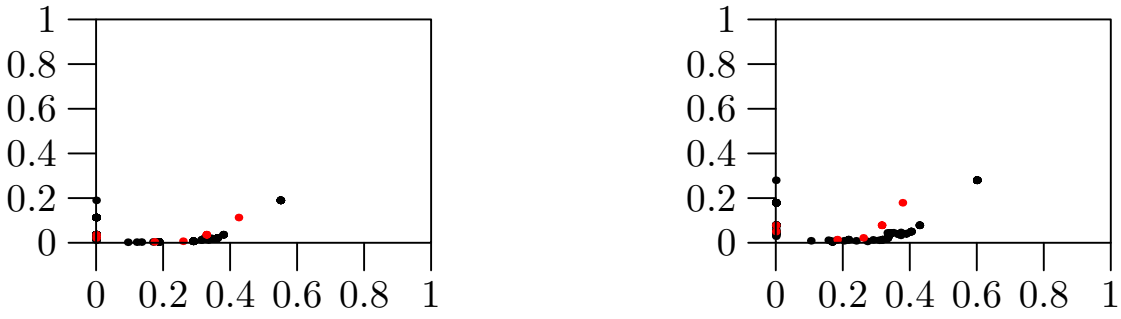

Figure 4.38: Random:bad1:Rounds(10-20)

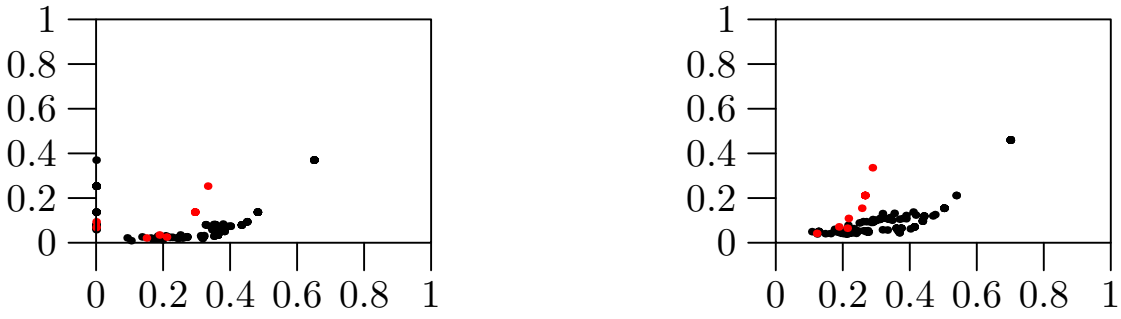

Figure 4.39: Random:bad1:Rounds(30-40)

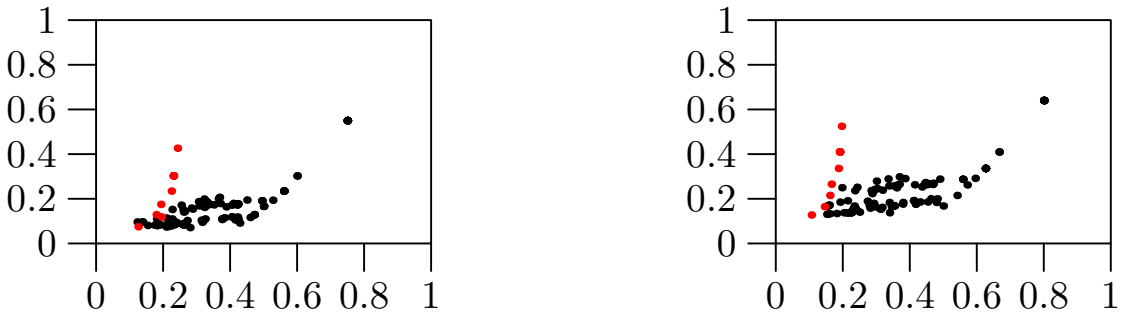

Figure 4.40: Random:bad1:Rounds(50-60)

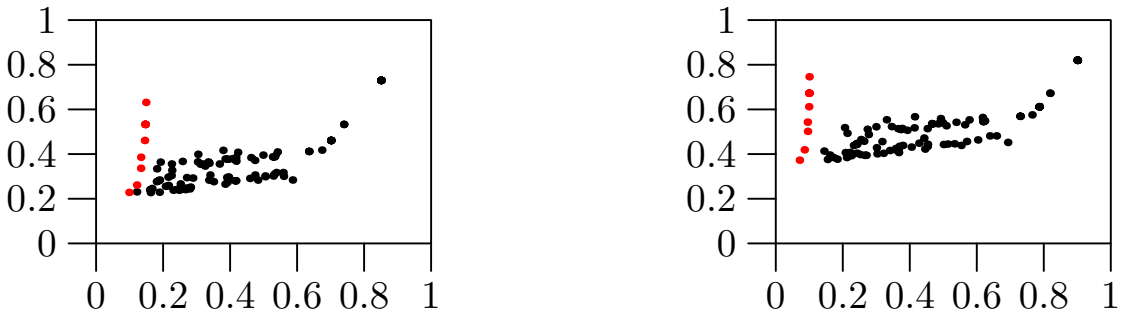

Figure 4.41: Random:bad1:Rounds(70-80)

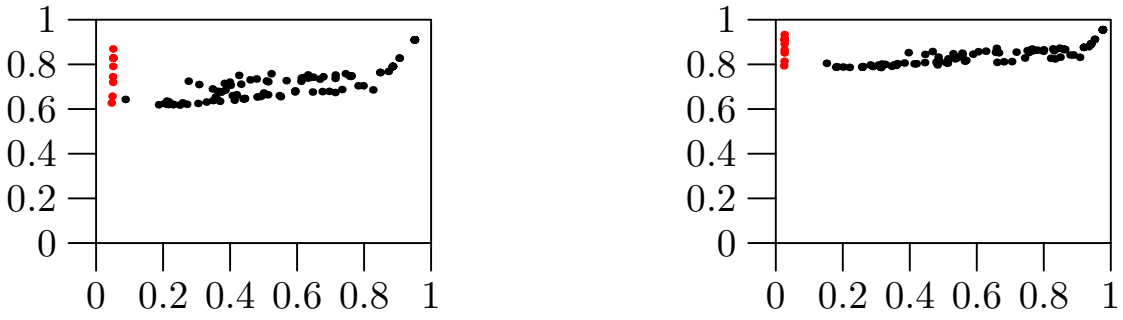

Figure 4.42: Random:bad1:Rounds(90-95)

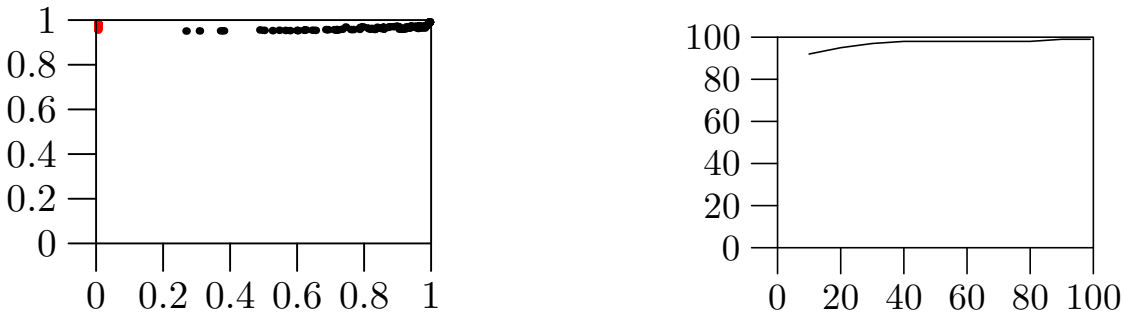

Figure 4.43: Random:bad1:Round(99)AndClassifiedNodes

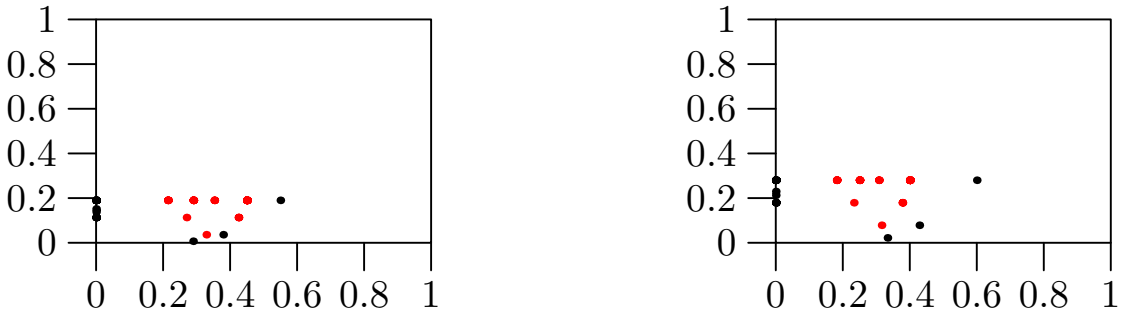

Figure 4.44: Random:bad5:Rounds(10-20)

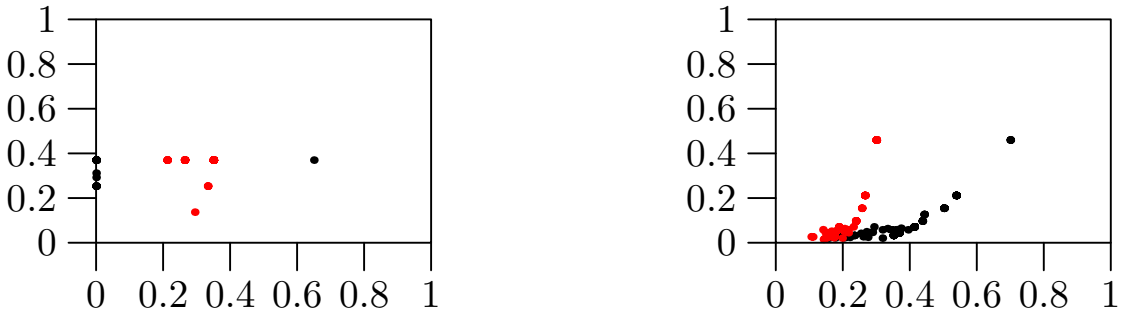

Figure 4.45: Random:bad5:Rounds(30-40)

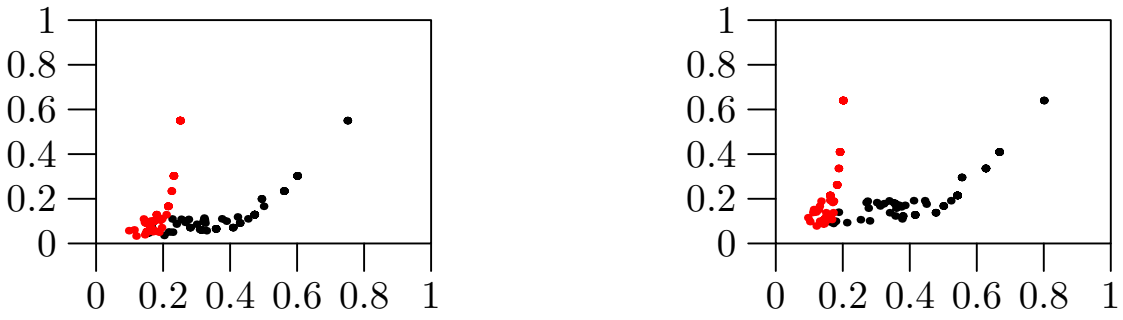

Figure 4.46: Random:bad5:Rounds(50-60)

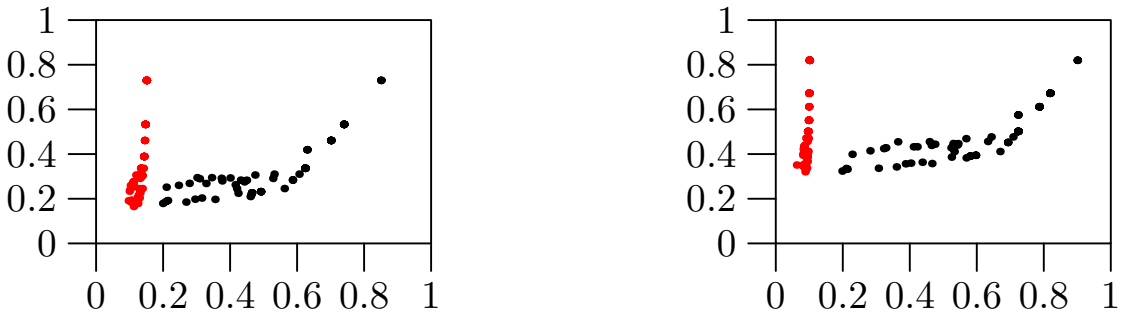

Figure 4.47: Random:bad5:Rounds(70-80)

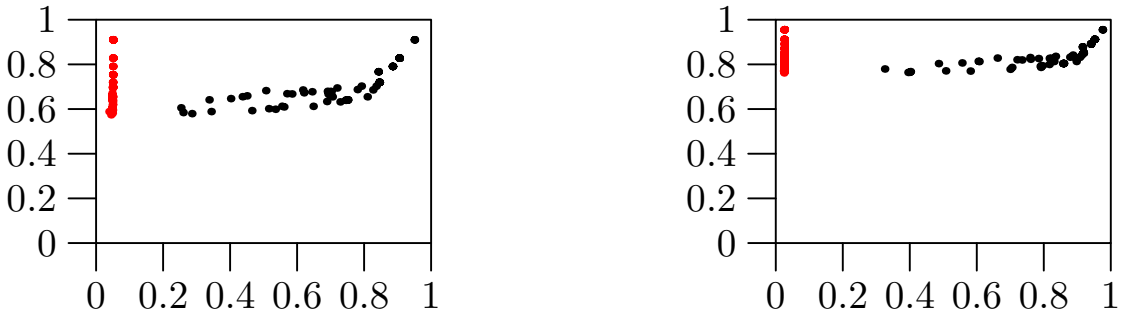

Figure 4.48: Random:bad5:Rounds(90-95)

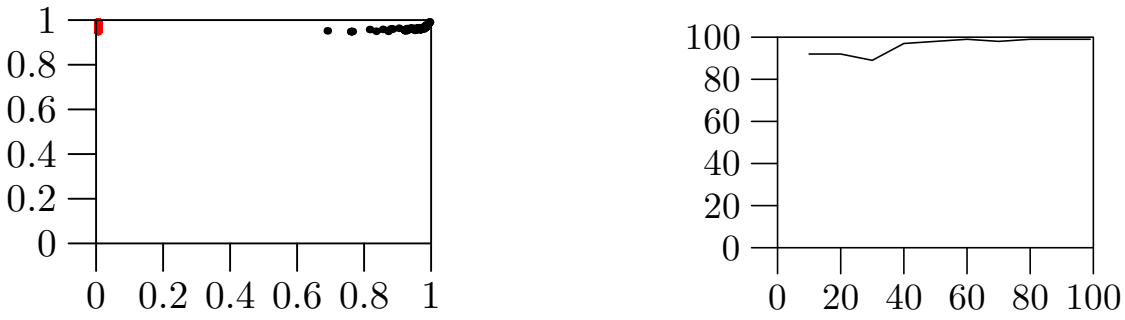

Figure 4.49: Random:bad5:Round(99)AndClassifiedNodes

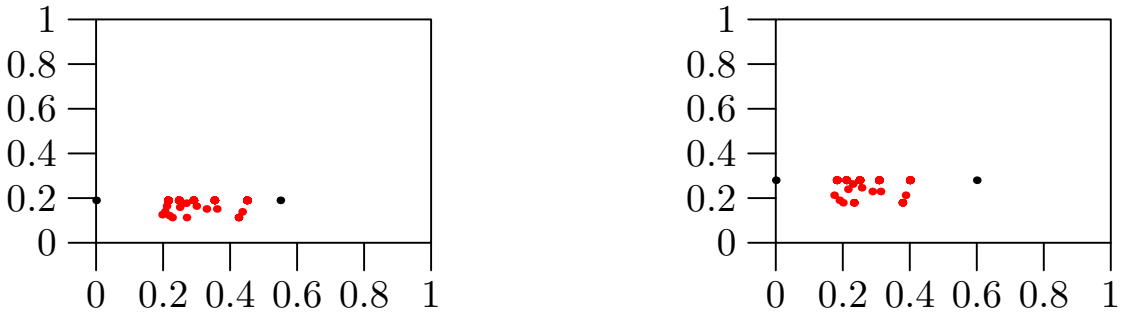

Figure 4.50: Random:bad9:Rounds(10-20)

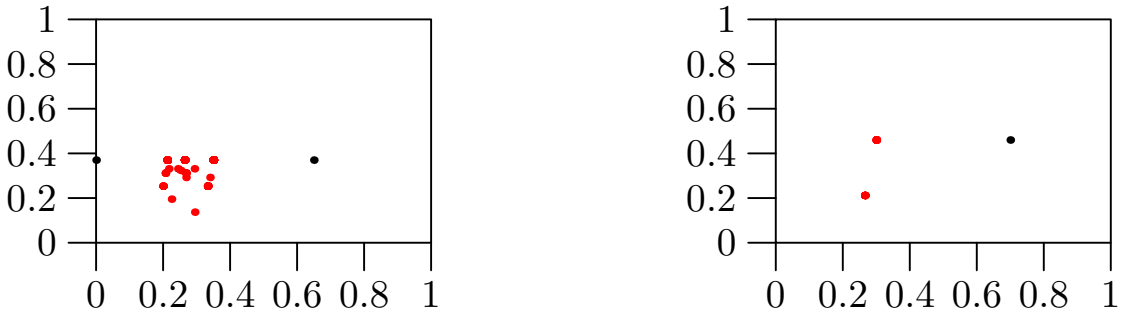

Figure 4.51: Random:bad9:Rounds(30-40)

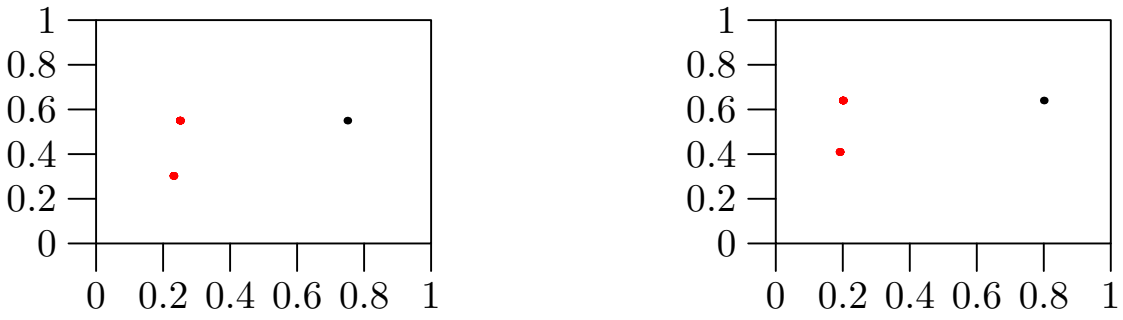

Figure 4.52: Random:bad9:Rounds(50-60)

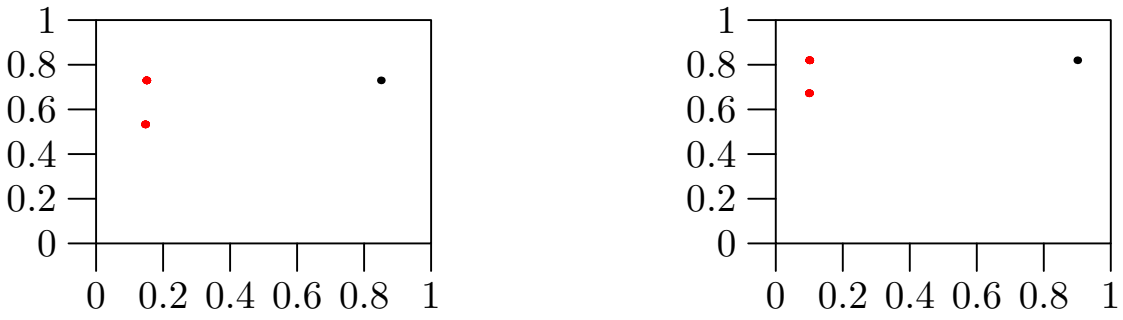

Figure 4.53: Random:bad9:Rounds(70-80)

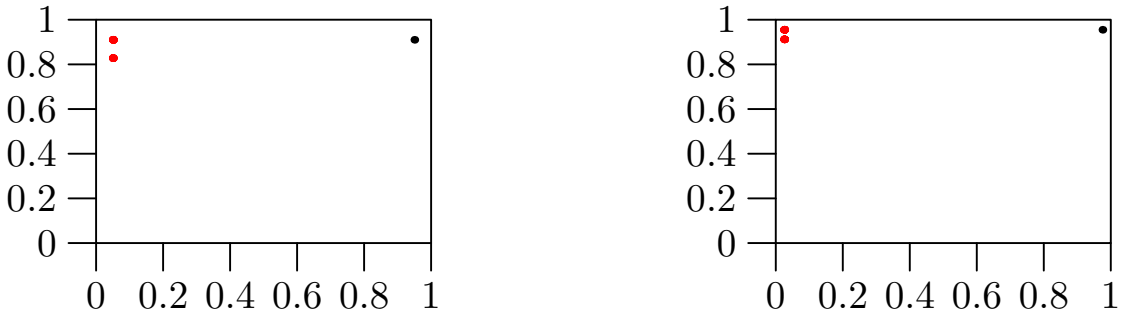

Figure 4.54: Random:bad9:Rounds(90-95)

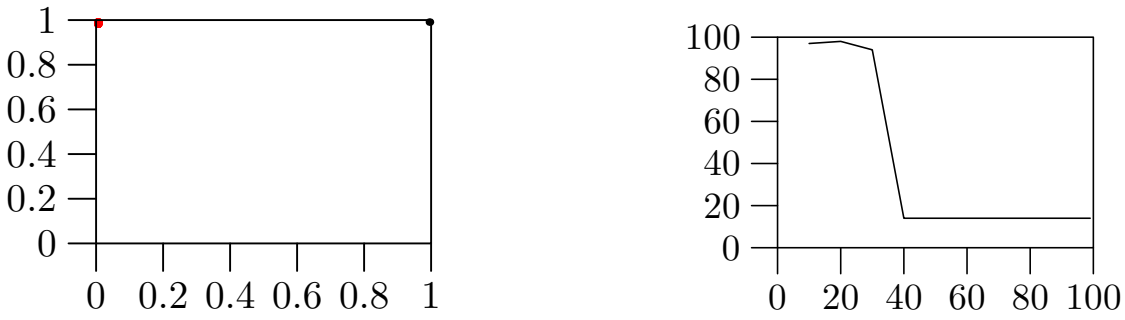

Figure 4.55: Random:bad9:Round(99)AndClassifiedNodes

We can observe some general trends in the diagrams shown. First of all, in the early rounds Good and Bad nodes are intermixed: there is no clear separating line. Even more, Bad nodes seem to be given better opinions than Good nodes, which is clearly undesirable. The explanation for this is based on two aspects of the scheme; namely, the trust threshold and the Bad nodes' way of assigning direct opinions. Initially, Bad nodes are allowed to vote, since the trust threshold (0.3) is lower than the initial default trust value (0.5), i.e. they have not been "discovered" yet. So, their $(0, 1)$ opinions for Good nodes are taken into account and the result is that Good nodes appear to be bad. Also, Bad nodes give $(1, 1)$ opinions to each other, hence reinforcing each other.

The situation in later rounds improves. The Good nodes move towards the upper right corner, the Bad ones towards the upper left. There is also a clear separating line between the two groups of nodes. For an actual implementation a practical guideline could be derived from the above observation, i.e. to be especially careful when making important trust decisions in early rounds. The trust computation may be based on too little raw evidence (direct opinions) to be relied upon. In all cases, however, the Good and Bad nodes are separated eventually (in the last rounds). This serves as a sanity check for the algorithm.

As the percentage of Bad nodes increases, we can see that the separation is still successful sooner or later, but the main observation is that the number of classified nodes is decreasing, especially for the Grid topology. Classified nodes are those for

which the evidence was sufficient, i.e. the confidence of the source's opinion for them was more than $\epsilon = 0.01$. The following graphs show the number of nodes classified in each topology, for different percentages of Bad nodes, after every round of computation. The general effect of Bad nodes on the number of classified nodes is that, after they are discovered, they block the trust paths they are on since they are not allowed to vote. So, nodes that are further away from the source than these Bad nodes can be reached by fewer paths. They may even be completely isolated. In any case, the confidence in the source's opinion for them is decreased, so some of them cannot be classified.

The Random topology performs best, because it is less affected by Bad nodes. This topology has a relatively short average path length between the source (s) and all other nodes, so confidence values for opinions are not too low. At the same time, it does not rely on information provided by any single node or small set of nodes. The links are random, so every node is reached through different paths.

The average path length from the source is the main defect of the Grid topology, since for certain nodes it may be large. If this is coupled with Bad nodes blocking some of the paths, the confidence values for nodes that are away from the source is dropping considerably. The more bad nodes, the more pronounced this effect is. So, the Grid topology performs worst of all.

As far as the Small World topology is concerned, the path length is short, since there are some highly connected nodes. So, it performs better than the Grid topology. However, it is exactly these highly connected nodes that degrade the performance of the computation when they are Bad. The reason is, again, that they

block many paths and affect opinions for most nodes. If the majority of these highly connected nodes are Bad, few trust paths will be able to be established.

The 90% bad node case is interesting to examine specifically. First, there is a sudden drop in the number of classified nodes between rounds 30 and 40. This is so, because at this point the opinions for Bad nodes acquire trust values that are lower than the trust threshold, so they become ineligible to vote.

Second, and more intriguing, is that the Random topology becomes equivalent to the Grid topology, and the Small World topology performs better than both. The explanation is that almost all nodes are Bad, so only nodes one or (rarely) two hops away from the source can be classified. This is true for all topologies. But the Grid nodes have exactly 8 neighbors, and all Random nodes have approximately 8 neighbors, too. So, the number of classified nodes turns out to be around 20. On the other hand, in the Small World topology the source node is one of the highly connected nodes (19 neighbors, when the average degree is 8). So, all of the 19 neighbors, and some of the nodes that are two hops away are classified for a total of about 40 nodes. A practical guideline for the Small World topology would then be that highly connected nodes should be protected, better prepared to withstand attacks, or, in general, less vulnerable.

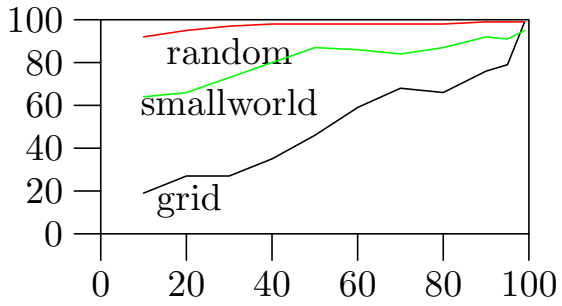

Figure 4.56: Node classification, 10% bad nodes

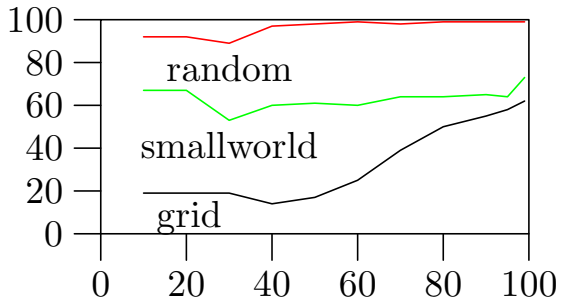

Figure 4.57: Node classification, 50% bad nodes

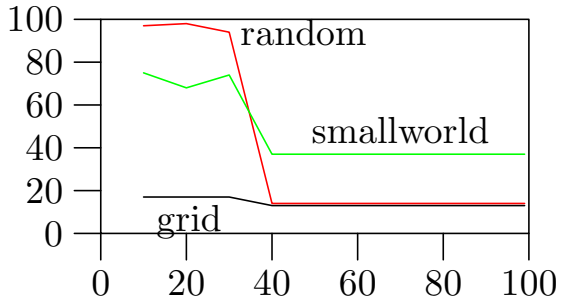

Figure 4.58: Node classification, 90% bad nodes

Chapter 5

Conclusion and Future Work

In this chapter we derive conclusions based on the goals we set for this work, as mentioned in the Introduction, and discuss in what degree these goals were met by our proposed scheme.

5.1 Conclusion

We have presented a scheme for evaluating trust evidence in Ad-Hoc networks. Our scheme is entirely based on information originating at the users of the network. No centralized infrastructure is required, although the presence of one can certainly be utilized. Also, users need not have personal, direct experience with every other user in the network in order to compute an opinion about them. They can base their opinion on second-hand evidence provided by intermediate nodes, thus benefitting from other nodes' experiences. Of course, we are taking into account the fact that second-hand (or third, or fourth...) evidence is not as valuable as direct experience. In this sense, our approach extends PGP, since PGP only uses directly assigned trust values.

At each round of computation, the source node computes opinions for all nodes. This means that information acquired at a single round can be stored and subsequently used for many trust decisions. If there is not enough evidence to determine an opinion, then no opinion is formed. So, when malicious nodes are present in the network they cannot fool the system into accepting a malicious node as benevolent. A failsafe state exists that ensures graceful degradation as the number

of adversaries increases. The trust topology also has significant influence on the performance of the algorithm. We have seen that if any node can be malicious with the same probability, the Random topology performs better. On the other hand, if the highly connected nodes of the Small World topology are Good, the algorithm fares better at the crucial cases of malicious node preponderance.

5.2 Future Work

In future work, we plan to implement more elaborate models for the attackers' behavior, and for the measures taken against nodes that are being assigned low trust values (i.e., detected to be bad). So, the attackers will be facing a tradeoff between the amount of damage they can inflict, and the possibility of being, for instance, isolated from the rest network. Suitable strategies will be developed for Good as well as Bad nodes.

[1] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *Proceedings of the 1997 New Security Paradigms Workshop*, pages 48–60, September 1997.

[2] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM 2001)*, 2001.

[3] Thomas Beth, Malte Borcherding, and Birgit Klein. Valuation of trust in open networks. In *Proceedings of the European Symposium on Research in Computer Security, ESORICS94*, pages 3–18, 1994.

[4] Vijay Bharadwaj and John Baras. Dynamic adaptation of access control policies. In *Proceedings of MILCOM 2003*, Boston, MA, October 2003.

[5] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The KeyNote trust management system. RFC 2704, September 1999.

[6] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[7] Rakesh Babu Bobba, Laurent Eschenauer, Virgil Gligor, and William Arbaugh. Bootstrapping security associations for routing in mobile ad-hoc networks. In *Proceedings of IEEE Globecom 2003*, San Francisco, CA, December 2003.

[8] S. Buchegger and J. Y. Le Boudec. The effect of rumor spreading in reputation systems for mobile ad-hoc networks. In *Proceedings of WiOpt '03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Sophia-Antipolis, France, March 2003.

[9] S. Buchegger and J. Y. Le Boudec. A robust reputation system for mobile ad hoc networks. Technical Report IC/2003/50, EPFL-DI-ICA, July 2003.

[10] Srdjan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux. Small worlds in security systems: an analysis of the pgp certificate graph, 2002.

[11] Srdjan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux. SECTOR: Secure tracking of node encounters in multi-hop wireless networks. In V. Swarup and S. Setia, editors, *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2003)*, Fairfax, VA, October 2003.

[12] Srdjan Čapkun, Jean-Pierre Hubaux, and Levente Buttyán. Mobility helps security in ad hoc networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2003)*, Annapolis, MD, June 2003.

[13] Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[14] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[15] Jason Eisner. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, July 2002.

[16] Paul Erdös and Alfréd Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[17] Laurent Eschenauer, Virgil D. Gligor, and John Baras. On trust establishment in mobile ad-hoc networks. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *10th International Security Protocols Workshop, Cambridge, UK, April 2002*, volume 2845 of *Lecture Notes in Computer Science*, pages 47–66. Springer-Verlag, 2004.

[18] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 1(2), 2002.

[19] V.D. Gligor, S.W. Luan, and J.N. Pato. On inter-realm authentication in large distributed systems. In *Proceedings of the 1992 IEEE Symposium on Research in Security and Privacy*, May 1992.

[20] Virgil D. Gligor, Himanshu Khurana, Radostina K. Koleva, Vijay G. Bharadwaj, and John S. Baras. On the negotiation of access control policies. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *9th International Security Protocols Workshop, Cambridge, UK, April 2001*,

volume 2467 of *Lecture Notes in Computer Science*, pages 188–201. Springer-Verlag, 2002.

[21] Rolf Haenni. Web of trust: Applying probabilistic argumentation to public-key cryptography.

[22] Amir Herzberg, Yosi Mass, Joris Michaeli, Dalit Naor, and Yiftach Ravid. Access control meets public key infrastructure, or: Assigning roles to strangers. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 2–14, Berkeley, CA, May 2000.

[23] Jean-Pierre Hubaux, Levente Buttyán, and Srdjan Čapkun. The quest for security in mobile ad hoc networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001)*, 2001.

[24] Audun Jøsang. An algebra for assessing trust in certification chains. In *Proceedings of the Network and Distributed Systems Security (NDSS'99) Symposium*, 1999.

[25] Audun Jøsang, Elisabeth Gray, and Michael Kinateder. Analysing topologies of transitive trust. In *Proceedings of the First International Workshop on Formal Aspects in Security & Trust: FAST2003*, Pisa, Italy, September 2003.

[26] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust algorithm for reputation management in p2p networks. In *WWW2003*, May 2003.

[27] Reto Kohlas and Ueli Maurer. Confidence valuation in a public-key infrastructure based on uncertain evidence. In *Proceedings of Public Key Cryptography 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 93–112. Springer-Verlag, January 2000.

[28] Raph Levien. *Attack-resistant trust metrics (draft).* PhD thesis, UC Berkeley, www.levien.com/thesis/.

[29] Raph Levien and Alex Aiken. Attack-resistant trust metrics for public key certification. In *Proceedings of the 7th USENIX Security Symposium, San Antonio, Texas*, pages 229–242, January 1998.

[30] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. A practically implementable and tractable delegation logic. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 27–42. IEEE Computer Society Press, May 2000.

[31] Ninghui Li, William H. Winsborough, and John C. Mitchell. Distributed credential chain discovery in trust management. In *Proceedings of CCS01, Philadelphia, Pennsylvania, USA.*, pages 156–165, November 2001.

[32] Sergio Marti, Prasanna Ganesan, and Hector Garcia-Molina. Sprout: P2P routing with social networks. Technical report, Stanford University, January 2004.

[33] Sergio Marti and Hector Garcia-Molina. Limited reputation sharing in p2p systems.

[34] Sergio Marti and Hector Garcia-Molina. Examining metrics for peer-to-peer reputation systems. Technical report, Stanford University, July 2003.

[35] Sergio Marti, T.J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad-hoc networks. In *Proceedings of MOBICOM 2000*, pages 255–265, 2000.

[36] Ueli Maurer. Modelling a public-key infrastructure. In E. Bertino, editor, *Proc. 1996 European Symposium on Research in Computer Security (ESORICS' 96)*, volume 1146 of *Lecture Notes in Computer Science*, pages 325–350. Springer-Verlag, 1996.

[37] Seapahn Meguerdichian, Sasa Slijepcevic, Vahag Karayan, and Miodrag Potkonjak. Localized algorithms in wireless ad-hoc networks: Location discovery and sensor exposure. In *Proceedings of ACM MobiHoc 2001*, October 2001.

[38] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *J. Autom. Lang. Comb.*, 7(3):321–350, 2002.

[39] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, San Antonio, TX, January 2002.

[40] Bartosz Przydatek, Dawn Song, and Adrian Perrig. SIA: Secure information aggregation in sensor networks. In *SenSys '03*, Los Angeles, CA, November 2003.

[41] Michael K. Reiter and Stuart G. Stubblebine. Resilient authentication using path independence. *IEEE Trans. Comput.*, 47(12):1351–1362, December 1998.

[42] Michael K. Reiter and Stuart G. Stubblebine. Authentication metric analysis and design. *ACM Trans. Inf. Syst. Secur.*, 2(2):138–158, May 1999.

[43] Günter Rote. Path problems in graphs. *Computing Supplementum*, 7:155–189, 1990.

[44] Anand Srinivas and Eytan Modiano. Minimum energy disjoint path routing in wireless ad-hoc networks. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, pages 122–133. ACM Press, 2003.

[45] D. Watts and S. Strogatz. Collective dynamics of "smallworld" networks. *Nature*, 393, 1998.

[46] William H. Winsborough, Kent E. Seamons, and Vicki E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, January 2000.

[47] Li Xiong and Ling Liu. Building trust in decentralized peer-to-peer electronic communities. In *Fifth International Conference on Electronic Commerce Research (ICECR-5)*, Canada, 2002.

[48] Raphael Yahalom, Birgit Klein, and Thomas Beth. Trust relationships in secure systems - a distributed authentication perspective. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, 1993.

[49] Raphael Yahalom, Birgit Klein, and Thomas Beth. Trust-based navigation in distributed systems. *Computing Systems*, 7(1):45–73, Winter 1994.

[50] Philip R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.