Evolving a Trust Model for Peer-To-Peer Networks Using Genetic Programming

Ugur Eray Tahta[1,2], Ahmet Burak Can[1], and Sevil Sen[1]

[1] Department of Computer Engineering
Hacettepe University, 06800 Ankara/Turkey
[2] ASELSAN, 06370 Ankara/Turkey

Abstract. Peer-to-peer (P2P) systems have attracted significant interest in recent years. In P2P networks, each peer act as both a server or a client. This characteristic makes peers vulnerable to a wide variety of attacks. Having robust trust management is very critical for such open environments to exclude unreliable peers from the system. This paper investigates the use of genetic programming to asses the trustworthiness of peers without a central authority. A trust management model is proposed in which each peer ranks other peers according to local trust values calculated automatically based on the past interactions and recommendations. The experimental results have shown that the model could successfully identify malicious peers without using a central authority or global trust values and, improve the system performance.

1 Introduction

In the last decade, with the fast expansion and improvement of peer-to-peer (P2P) systems, malicious activities have become a major security problem in P2P systems. Due to openness of P2P systems, unreliable users may occupy considerable portions of P2P populations. Trust management in such open environments is an important and difficult research problem. Trust management models generally aim to exclude unreliable peers from P2P systems. However maintaining true trust relationships without a priori knowledge is a very hard problem. It is difficult to distinguish malicious peers from innocent ones with a certainty in such environments. Thus, most of the proposed trust models in the literature offer approximate decision guidelines about peers.

Trust management can be accomplished by a central authority, such as eBay. Participants in eBay can rate each other at the end of auctions and information about auctions is stored in the central server. However, having a central authority conflicts with the nature of P2P systems. Thus peers need to organize themselves to manage and store information about their trust relationships [1–3]. In pure P2P networks like Gnutella [4], peers flood trust queries to the network in order to obtain trust information about others. In such a network, all peers store trust information about neighbors according to the past interactions [2, 5, 6]. Queries enable to collect recommendations about the queried peer and make a decision about it. Some models use distributed hash tables (DHT) to store

trust information [1, 3, 7]. Each peer stores the trust information about other peers determined by a DHT algorithm, which enables efficient access to the information. Thus peers can learn the global trust information about others without flooding queries to the whole network.

Trust management in P2P systems is a difficult problem due to the lack of a central authority and uncertain information collected from peers. P2P trust models should be able to recognize complicated behavioral patterns of malicious peers and make smart decisions to distinguish malicious peers from benign peers using this uncertain information. Using machine learning techniques might be a good choice for such a complex problem. In this paper, we propose a genetic programming (GP) based trust management model. Our model intends to determine characteristics of malicious and benign peers using the features derived from peers. Two kinds of information are collected by peers: interactions and recommendations. Peers store their past interactions with other peers and collect recommendations about peers from their neighbors. These two types of information provide bases for the feature set. A trust model is evolved with these features by using genetic programming in order to measure trustworthiness of peers. Peers do not collect information about all other peers. A peer creates a view with the peers interacted in the past or intended to interact with. Each peer ranks other peers according to the trust values generated by the model which is evolved by using genetic programming, and makes download decisions using these values. Using the generated trust values, malicious peers are excluded from the system.

The paper is organized as follows. Section 2 discusses the related research. Section 3 introduces the proposed trust model. Section 4 presents the simulation environment and gives the experimental results. Section 5 summarizes the conclusion.

2 Related Work

P2P systems offer sharing environments for common resources by improving diversity, prevalence and easy accessibility. On the other hand, these characteristics make them vulnerable to many attacks. P2P systems can be divided into two groups; structured and unstructured [8]. In the unstructured overlay networks, queries are flooded in the network, such as in Gnutella [9]. The structured P2P networks generally utilize DHTs for indexing information on peers selected by the DHT algorithm. For example, Chord system [10] proposes a decentralized network with a distributed lookup primitive on a circular Chord ring. Peers on this ring are charged to store information determined by the Chord's algorithm.

Most of the prominent trust models use the reputation concept and statistical models to make decisions on trustworthiness of peers. Reputation generally relies on peer's past experiences and recommendations from other peers, such as in XRep [11] or P2PRep [12]. EigenRep [3] uses transitivity of trust to calculate trust values. Conner et al. [13] proposed a reputation-based trust management framework supporting synthesis of trust-related feedback from different entities.

In [14], an effective way of calculating reputation has been presented. The model considers several features such as number, age, or frequency of transactions, how frequently a given peer attends a common vendor, and the number of common vendors between the pairs. It aims to investigate the characteristics of transactions executed by malicious peers.

Detecting malicious peer behaviors with the help of machine learning techniques is another promising approach for generating trust management models. Weihua Song et al. [15] uses neural networks and derives trust values from heterogeneous agents based on recommendations. The agents classify recommendations as qualified or unqualified for choosing the providers. In [16], support vector machines are used to reduce the cost of communication with less query and to improve the success rate. In [17] a generic trust framework is proposed by using linear discriminant analysis and decision trees. An agent uses its own previous transactions (with other agents) to build a knowledge base and distinguishes successful transactions from unsuccessful ones.

There are some applications of evolutionary computation techniques to computer and network security in the literature. One of the mostly employed area is intrusion detection in which either genetic programming (GP) or genetic algorithm (GA) is mainly used. The first GP application to intrusion detection is given by Crosbie and Stafford [18]. Since then there are many useful applications to the field. In [19], Abraham and Grosan compare the genetic programming technique with other machine learning methods for intrusion detection [19] and show that genetic programming techniques outperform other techniques and are lightweight. The grammatical evolution technique is successfully employed for intrusion detection on wired networks [20] and on ad hoc networks [21]. Sen and Clark [22] employ multi-objective evolutionary computation (MOEC) techniques in order to show how energy usage and detection ability can be traded off for resource-constrained networks. Moreover, they show the significant potential of evolutionary computation techniques to explore the suitable intrusion detection architecture by taking into account the objectives of cooperative intrusion detection programs. The MOEC techniques are also used to explore how intrusion detection system sensors could be best placed on a network in [23].

Even though there are many applications of evolutionary computation techniques to the intrusion detection problem, as far as we know there is only one application of genetic algorithm in order to detect attackers in P2P domain. A peer profile based trust model proposed by Selvaraj et al. [24] uses genetic algorithm. This model combines peer profiling with an anomaly detection technique. It establishes trust using only local interaction data of the peer. There is a trusted central authority which manages the peer list to secure peers' IDs. Our model have used both interaction data from peer's own experience and recommendation data collected from other peers. Additionally, our model does not depend on a central authority to calculate trust values. This is believed to be a more suitable approach for P2P systems.

3 The Model

The proposed trust model uses genetic programming to make trusting decisions on peers. Genetic Programming (GP) is a common evolutionary computation technique, which is introduced to the machine learning community by Koza [25]. Banzhaf [26] comes up with an assertion that GP could produce more successful results comparing to other machine learning techniques and programs written by people.

In GP, functions (operators, program statements etc.) and terminals (features, constants etc.) build a GP tree. Each GP tree represents an individual. Basically, a group of individuals which are the candidate solutions to the problem are generated by GP in each generation. How well the individuals solve the problem is evaluated by using a fitness function.

3.1 Feature Sets And Operators

Selecting the right feature set is a difficult problem and a key point to obtain successful results in GP and other machine learning techniques [27]. In our model, the information collected from past interactions and recommendations of neighbors form the feature set.

Interaction based features are obtained from the peer's past experiences with other peers. These experiences occur directly between two peers who interacted in the past. Interactions can be any activity specific to the P2P application, such as file sharing, CPU sharing, and storage sharing. Interaction based features are listed in Table 1.

Table 1. Interaction Based Features

Feature	**Symbol**
number of interactions	f1
number of successful interactions	f2
average size of downloaded files	f3
average time difference between last two interactions	f4
average weight	f5
average satisfaction	f6

Satisfaction and weight parameters are calculated as in [28]. Successful interactions are the interactions that the file download is finished successfully. Satisfaction parameter is calculated based on average bandwidth, agreed bandwidth before the interaction, online, and offline period values of the uploader:

$$Satisfaction = \begin{cases} (\frac{AveBw}{AgrBw} + \frac{OnP}{OnP+OffP})/2 & if\ AveBw < AgrBw, \\ (1 + \frac{OnP}{OnP+OffP})/2 & otherwise \end{cases} \quad (1)$$

Weight parameter is calculated based on file size, number of uploaders of the dowloaded file, number of uploaders of the maximum uploaded file:

$$Weight = \begin{cases} (\frac{size}{100MB} + \frac{\#Uploaders}{Uploader_{max}})/2 \; if \; size < 100MB, \\ (1 + \frac{\#Uploaders}{Uploader_{max}})/2 \qquad otherwise \end{cases} \tag{2}$$

The second set of features is recommendation based features. When a peer wants to interact with another peer, it asks its own neighbors about their experiences. The neighbors who have information about the peer requested send their recommendations. These experiences about another peer are called recommendations. A recommendation contains the following information: average number of successful interactions, average satisfaction of interactions, average weight of interactions, and calculated trust value of the queried peer. The recommendation based features are listed in Table 2:

Table 2. Recommendation Based Features

Feature	**Symbol**
number of recommendations	f7
average of neighbours' average number of successful interactions	f8
average of neighbours' average satisfaction values	f9
average of neighbours' average weight values	f10
average of trust values	f11

In our genetic model, we use simple operators to generate a formula for trust calculation. The operators used in our model are addition, subtraction, division, multiplication, inverse, log, square root, and square.

3.2 Fitness Function

The fitness function is one of the important factors affecting the performance of evolutionary computation techniques. The fitness function determines how well a program is able to solve the problem [25, 29]. In the evolved trust model, a fitness function based on the reduction in the number of attacks is used. In other words, if R_{trust} denotes the number of attacks with our trust model and $R_{noTrust}$ denotes the number of attacks without any trust model, then our fitness function is;

$$fitness = R_{trust}/R_{noTrust}. \tag{3}$$

If the generated individuals can mitigate the number of attacks, the value of fitness function decreases and the success of the model increases. Thus, the fitness function is aimed to be minimized in our genetic model. At the end of the evolution, the most successful individual is selected as the solution.

4 Experiments And Analysis

The experiment environment consists of two integrated modules. First one is a file sharing simulation program implemented in Java language to asses the evolved trust model in P2P environments against malicious attacks. The second one is the ECJ 21 toolkit [30] for the GP implementation. It is integrated with the simulation program to train the trust model. In the experiments, the population and generation sizes are chosen as 100 and 300 respectively. The other parameters are equal to the default parameters of the ECJ toolkit.

4.1 Simulation Module

The simulation module is adapted from the program used in [28]. Each simulation takes 50.000 cycles, where each cycle represents 10 minutes of network activity. There are 1000 peers in each simulation. Basically, peers interact with each other for sharing a file and build a reputation according to their behaviors. At the beginning of the simulation, peers are strangers to each other. When a peer uploads a file to another peer, it becomes a neighbor of the peer. A neighbor is preferred over a stranger if they are equally trustworthy.

Peers build an interaction history while downloading and uploading files. If a peer intends to download a file, it gets the list of file providers. Then, it calculates the trust values of these file providers using its own interaction history and recommendations from its neighbors. Trust values are calculated based on the formula generated by the genetic programming module using the features and the fitness function explained in Section 3. If a peer has neighbors in the file provider list, it prefers the one with the highest trust value. Otherwise, it downloads the file from the stranger who has the highest trust value. At the end of a download process, if the file provider uploads a virus infected or an inauthentic file, it is marked as a malicious peer and is never interacted again.

4.2 GP Module

The GP module works in an integrated manner with the simulation module. It trains our trust model against various attacker types and tries to find the best individual in order to evaluate trust values of peers. In the training process, GP creates individuals by using the features and the operators given in Section 3.1. Each individual runs the file sharing simulation from start to finish. Reduction in the number of attacks represents the success of an individual. When the best individual is found, it is tested on various attacker models on the simulation module. The general steps of the GP Module are listed in Algorithm 1.

4.3 The Problem

Generally, a P2P network consists of good peers and malicious peers (attacker). A good peer always gives fair recommendations and uploads authentic files.

Algorithm 1 How Gp Module Works

```
initialize population
while current generation <= maximum generation do
   for all individuals in the current generation do
      execute simulation
      evaluate the fitness function
   end for
   apply genetic operators (selection, crossover, reproduction, mutation, etc.) to the individuals
   create new population
end while
```

However, a malicious peer may upload inauthentic files or give unfair recommendations to harm the system. Reducing the number of inauthentic/infected file uploads and unfair recommendations is the aim of a trust management model.

In our simulation, malicious peers are considered to behave in two different ways: naive and hypocritical. If malicious peers perform unaccompanied attacks and do not aware of other malicious peers, they are called individual attackers. Individual attackers can behave as described below:

- *Naive:* The attacker always uploads virus infected/inauthentic files and gives unfair recommendations to others [31].
- *Hypocritical:* The attacker perform attacks by uploading inauthentic files or giving unfair recommendations with x% probability. Otherwise, it acts like a good peer [3, 5].

If a group of peers know each other and attack to other peers as a team, they are called collaborators. Collaborators always upload authentic files to each other. If a good peer requests a recommendation from a collaborator about another collaborator, the collaborator might give high recommendations unfairly in order to improve the queried collaborator's trust value. The types of attack carried out by collaborators are be described as follows:

- *Naive:* Collaborators always upload virus infected/inauthentic files to good peers and gives unfair recommendations to good peers.
- *Hypocritical:* Collaborators perform attacks by uploading inauthentic files to good peers or giving unfair recommendations with x% probability. Otherwise, it acts like a good peer.

4.4 Experiments

In the experiments, the model is trained for all types of individual attackers firstly. Training is done with a network setup in which 10% of the peers is malicious. The best results of 10 runs is chosen for each attack type. Then, the trained model is tested with 10%, 30% and 50% malicious peers ratio in the networks. During the experiments, the attack probability of hypocritical attackers is chosen as 20% in all interactions. If a peer uploads a virus infected or inauthentic file, it is counted as a file-based attack. Initially, the simulation is executed without the trust model for each network setting in order to figure out

the number of attacks when a trust model does not exist. Then the simulations are run with the evolved trust model. Success of the trust model is assessed by the number of attacks prevented with the model.

Table 3. Success ratio of the trust model against individual attackers for the file-based attacks

	10%	**30%**	**50%**
Naive	83.8	78.9	73.6
Hypocritical	71.8	57.7	47.1

Table 3 shows the success ratio of the evolved trust model against individual attackers according to varying malicious peer populations in the network. The model has a notable success against individual naive attackers. Since identifying a naive attacker is easy after the first interaction, a high percentage of these attacks can be prevented. Our model has a good success ratio for individual hypocritical attackers, which is 71.8% in a network in which 10% of the peers is malicious. In the network in which 50% of the peers is malicious, the trust model could prevent nearly half of the attacks as shown in Table 3. In such extremely malicious networks, this is a good success ratio for hypocritical attackers.

Convergence speed of the trust model is important to identify attacks in a reasonable time. Figure 1 shows the decrement in the number of attacks by naive and hypocritical individual attackers when the evolved trust model is used.

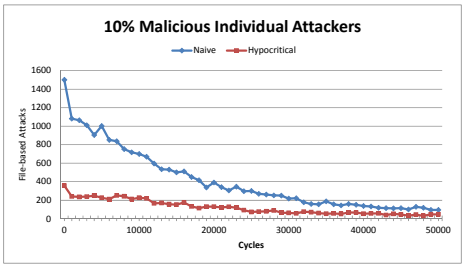

Fig. 1. File-based attacks over time in a network consisting of 10% individual attackers

Unfair recommendations given by malicious peers are considered as recommendation-based attacks. The evolved trust model has also good performance on recommendation-based attacks. Figure 2 shows the decrement in the recommendation-based attacks over time. In the model, if a peer intends to collect recommendations about another peer, it firstly requests recommendations from its trustworthy neighbors. Therefore, unfair recommendation rate is mitigated over time as peers gain more neighbors. However, unfair recommendations do not drop as quickly as file-based

attacks since determining an unfair recommendation is not easy as determining an infected/inauthentic file.

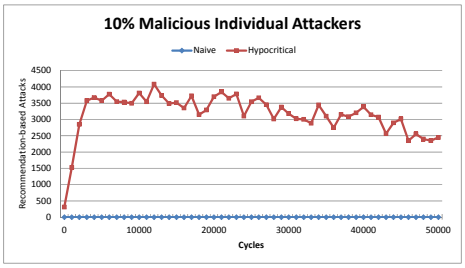

Fig. 2. Recommendation-based attacks over time in a network consisting of 10% individual attackers

The second step of the experiments is done with collaborative attackers. Like individual attackers, at first the model is trained against the collaborators, and then tested on various malicious network setups. Collaborators attack to other peers as a team and give fair recommendations to each other. The attack probability of hypocritical attackers is chosen as 20% in all interactions in the experiments. The team size of collaborators is set to 50 peers.

Table 4. Success ratio of the trust model against collaborators for the file-based attacks

	10%	**30%**	**50%**
Naive	79.3	75.1	71.9
Hypocritical	61.7	46.3	39.5

Table 4 shows the success ratio of the trust model against collaborators in networks consisting of varying malicious peer population. Naive collaborators are identified by good peers after the first interaction. Hence they can not disseminate high recommendations about each other and can not take advantage of collaboration. The success ratio of preventing attacks in naive collaborators is 79.3% in a network in which 10% of the peers is malicious and, this performance drops to only 71.9% even the ratio of malicious peers is increased to 50%. However, hypocritical collaborators are more effective than naive ones. Detection of hypocritical collaborators is more difficult since they perform attacks intermittently. A hypocritical collaborator can disseminate high recommendations about its team mates before being identified by good peers. Since the collaborators help each other in order to evade detection, their identifications become very difficult. However, the trust model could still prevent 61.7% of file-based attacks carried out by hypocritical collaborators in a network in which 10% of peers is malicious.

Figure 3 shows the number of file-based attacks over time in a network consisting of 10% collaborators. The model decreases the number of effective attacks carried out by naive and hypocritical collaborators dramatically.

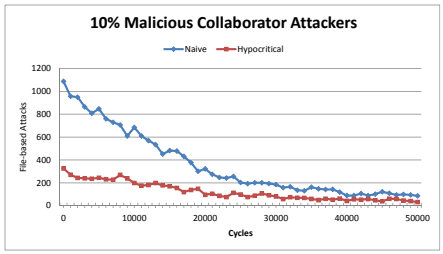

Fig. 3. File-based attacks over time in a network consisting of 10% collaborators

Recommendation-based attacks carried out by collaborators are presented in Figure 4. High recommendations given by collaborators unfairly are also counted as recommendation-based attacks. Collaboration increases the number of misleading recommendations slightly. However, the trust model still mitigates the number of recommendation-based attacks. It also prevents misleading recommendations to increase over time.

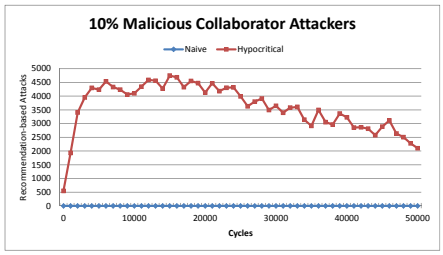

Fig. 4. Recommendation-based attacks over time in a network consisting of 10% collaborators

5 Conclusion

This paper proposes a trust model evolved by using genetic programming. Trust values of peers are calculated by a formula generated by this model. Malicious and benign peers are distinguished from each other based on these trust values. The experimental results show that the model could distinguish different types of attacks from benign behavior of good peers successfully. Naive and hypocritical attacker models are studied with individual and collaborative behaviors. The

model is trained against these types of attacks and evaluated on various network setups containing different ratio of malicious peers. Naive attackers are identified easily in both individual and collaborator scenarios. Hypocritical attackers are more difficult to deal with and more successful when they collaborate. The evolved trust model has decreased the number of file-based attacks in all scenarios with promising success ratios. Recommendation-based attacks are mitigated but not decreased as much as file-based attacks due to the difficulty of recognizing misleading recommendations. The evolved model showed that genetic programming could be employed to build a trust model in peer-to-peer networks.

References

1. K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proc. 10th International Conference on Information and Knowledge Management (2001 ACM CIKM)*, 2001.
2. F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "Choosing reputable servents in a p2p network," in *Proc. of the 11h Int. World Wide Web Conf.*, May 7–11 2002.
3. S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proc. of the 12th international conference on World Wide Web*, ser. WWW '03. ACM, 2003, pp. 640–651.
4. Clip2, "The gnutella protocol specification v0.4 (document revision 1.2)," http://www.clip2.com/GnutellaProtocol04.pdf, 2001,.
5. A. A. Selcuk, E. Uzun, and M. R. Pariente, "A reputation-based trust management system for p2p networks," in *Proc. of the 2004 IEEE International Symposium on Cluster Computing and the Grid*, ser. CCGRID '04. IEEE Computer Society, 2004, pp. 251–258.
6. R. Zhou, K. Hwang, and M. Cai, "Gossiptrust for fast reputation aggregation in peer-to-peer networks," *IEEE Trans. on Knowl. and Data Eng.*, vol. 20, no. 9, pp. 1282–1295, 2008.
7. L. Xiong and L. Liu, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Trans. on Knowl. and Data Eng.*, vol. 16, no. 7, pp. 843–857, Jul. 2004.
8. L. Xiao, Y. Liu, and L. M. Ni, "Improving unstructured peer-to-peer systems by adaptive connection establishment," *IEEE Transactions on Computers*, vol. 54, no. 9, pp. 1091–1103, 2005.
9. N. Stakhanova, S. Ferrero, J. S. Wong, and Y. Cai, "A reputation-based trust management in peer-to-peer network systems." in *ISCA PDCS*. ISCA, 2004, pp. 510–515.
10. E. Brunskill, "Building peer-to-peer systems with chord, a distributed lookup service," in *Proc. of the Eighth Workshop on Hot Topics in Operating Systems*, ser. HOTOS '01. IEEE Computer Society, 2001, pp. 81–.
11. E. Damiani, D. C. D. Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *In Proc. of the 9th ACM Conference on Computer and Communications Security*. ACM Press, 2002, pp. 207–216.
12. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati, "Managing and sharing servents' reputations in p2p systems," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 15, no. 4, July/August 2003.

13. W. Conner, A. Iyengar, T. A. Mikalsen, I. Rouvellou, and K. Nahrstedt, "A trust management framework for service-oriented environments." in *WWW*. ACM, 2009, pp. 891–900.
14. R. V. V. S. V. Prasad, V. Srinivas, V. V. Kumari, and K. V. S. V. N. Raju, "An effective calculation of reputation in p2p networks." *JNW*, vol. 4, no. 5, pp. 332–342, 2009.
15. W. Song, V. V. Phoha, and X. Xu, "An adaptive recommendation trust model in multiagent system." in *IAT*. IEEE Computer Society, 2004, pp. 462–465.
16. R. Beverly and M. Afergan, "Machine learning for efficient neighbor selection in unstructured p2p networks," in *Proc. of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques*, ser. SYSML'07. USENIX Association, 2007, pp. 1:1–1:6.
17. X. Liu, G. Tredan, and A. Datta, "A generic trust framework for large-scale open systems using machine learning," *CoRR*, vol. abs/1103.0086, 2011.
18. M. Crosbie and G. Stafford, "Applying genetic programming to intrusion detection," in *Proc. of AAAI Symposium on Genetic Programming*. Cambridge, MA, 1995, pp. 1–8.
19. A. Abraham and C. Grosan, "Evolving intrusion detection systems," in *Genetic Systems Programming: Theory and Experiences*. Springer, 2006, vol. 13, pp. 57–79.
20. D. Wilson and D. Kaur, "Knowledge extraction from kdd'99 intrusion data using grammatical evolution," *WSEAS Transactions on Information Science and Applications*, vol. 4, pp. 237–244, February 2007.
21. S. Sen and J. A. Clark, "A grammatical evolution approach to intrusion detection on mobile ad hoc networks," in *Proc. of the Second ACM Conference on Wireless Network Security*. ACM, 2009, pp. 95–102.
22. S. Sen and J. Clark, "Evolutionary computation techniques for intrusion detection in mobile ad hoc networks," *Computer Networks*, vol. 55, no. 15, pp. 3441–3457, oct 2011.
23. H. Chen, J. A. Clark, J. E. Tapiador, S. A. Shaikh, H. Chivers, and P. Nobles, "A multi-objective optimisation approach to ids sensor placement," in *CISIS*, 2009, pp. 101–108.
24. C. Selvaraj and S. Anand, "Peer profile based trust model for p2p systems using genetic algorithm." *Peer-to-Peer Networking and Applications*, vol. 5, no. 1, pp. 92–103, 2012.
25. J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.
26. W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin, *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.
27. M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, 1999.
28. A. B. Can and B. Bhargava, "Sort: A self-organizing trust model for peer-to-peer systems," *IEEE Trans. Dependable Sec. Comput.*, vol. 10, no. 1, pp. 14–27, 2013.
29. N. L. Cramer, "A representation for the adaptive generation of simple sequential programs." in *ICGA*, J. J. Grefenstette, Ed. Lawrence Erlbaum Associates, 1985, pp. 183–187.
30. "Ecj 21: A java-based evolutionary computation and genetic programming research system," http://www.cs.umd.edu/projects/plus/ec/ecj/, 2013.
31. C. Dellarocas, "Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior," in *Proc. of the 2nd ACM conference on Electronic commerce*, ser. EC '00. ACM, 2000, pp. 150–157.