Towards Trustworthy Multi-Cloud Services Communities: A Trust-based Hedonic Coalitional Game

Omar Abdel Wahab, Jamal Bentahar, Hadi Otrok, and Azzam Mourad

Abstract—The prominence of cloud computing led to unprecedented proliferation in the number of Web services deployed in cloud data centers. In parallel, service communities have gained recently increasing interest due to their ability to facilitate discovery, composition, and resource scaling in large-scale services' markets. The problem is that traditional community formation models may work well when all services reside in a single cloud but cannot support a multi-cloud environment. Particularly, these models overlook having malicious services that misbehave to illegally maximize their benefits and that arises from grouping together services owned by different providers. Besides, they rely on a centralized architecture whereby a central entity regulates the community formation; which contradicts with the distributed nature of cloud-based services. In this paper, we propose a three-fold solution that includes: trust establishment framework that is resilient to collusion attacks that occur to mislead trust results; bootstrapping mechanism that capitalizes on the endorsement concept in online social networks to assign initial trust values; and trust-based hedonic coalitional game that enables services to distributively form trustworthy multi-cloud communities. Experiments conducted on a real-life dataset demonstrate that our model minimizes the number of malicious services compared to three state-of-the-art cloud federations and service communities models.

Index Terms—Cloud Computing; trust; malicious service; game theory; services community; cloud federation; social network; bootstrapping.

1 Introduction

The advent of cloud computing has led to a rapid increase in the number of Web services that are deployed in cloud data centers. Practically, cloud providers have tendency to publish their computing services as Web services to keep up with the industry standards such as WSDL, SOAP, UDDI, and RESTful Web services [1]. Although this increase has the advantage of expanding the user's choice set, it entails several challenges related to the discovery, composition, and QoS management issues. In fact, discovering the relevant services among hundreds or even thousands of services offering the same functionality becomes a challenging task for cloud users. Similarly, selecting the appropriate services to participate in composition sequences in the cloud is becoming of combinatorial complexity. Moreover, the highly competitive market imposed by the large number of services that offer various functional/non-functional properties induces a continuous dilemma for service providers between delivering high-quality services and affording their associated operational costs. On the other hand, the demands for cloud-based services is expected to increase in such a way that makes providers' available resources insufficient to deal with [2]. This raises the need for providers to reshape their business strategies in order to upgrade their resource scaling capabilities. Communities of Web services [3] provide an effective platform for addressing the aforementioned challenges. The idea is to group services sharing same domain of interest or functionality into a set of homogenous clusters. This has the advantages of (1) facilitating the discovery of services by enhancing their visibility towards users; (2) reducing the complexity of building composition sequences in the cloud by localizing the selection process; and (3) improving the QoS management by overcoming the problem of limited resources; particularly at the Infrastructure-as-a-Service (IaaS) layer [4].

- *O. Abdel Wahab and J. Bentahar are with the Concordia Institute for Information Systems Engineering, Concordia University, Montréal, Canada.*
- *H. Otrok is with the Department of ECE, Khalifa University of Science, Technology and Research, Abu Dhabi, UAE.*
- *A. Mourad is with the Department of Mathematics and Computer Science, Lebanese American University, Beirut, Lebanon.*

1.1 Problem Definition

Several services' community formation models have been proposed in the literature [3], [5], [6], [7], [8]. Although these models may work well when services reside in the same data center, they are unable to support multi-cloud communities formation. Nonetheless, cooperation between services from different clouds is often needed. In fact, besides the aforementioned benefits of the traditional services' communities, multi-cloud services' communities provide additional benefits for both providers and customers. Practically, such an architecture increases the flexibility of providers in managing clients' requests and meeting the Service-Level Agreement (SLA) requirements by allowing them to move workloads from one provider's site to another, when needed, in a seamless manner. It allows as well to break down the customer's data at the bit level in order to enable its parallel processing by service instances sharing the same application logic but residing at different clouds. Besides optimizing latency and throughput via enabling the assignment of the requests to the most appropriate Virtual Machines (VMs) at each provider's site, protecting the privacy and security of the data is an additional motivation for customers to favor such a model, where each provider can be aware of only a small part of the data [9]. As a tangible example, allowing users to toggle between Bing or Google Maps in a visualization Web site would, besides improving the service's availability, enhance the user's experience by allowing him to select the service that he feels more familiar with [1].

The main limitation that makes the existing community formation models inappropriate for a multi-cloud environment is mainly their reliance on a centralized architecture whereby a central third-party called *master* [6] is responsible for managing the formation, membership, and security issues in the community. However, the distributed nature of cloud-based services plays against the existence of such a central entity. Practically, the fact that Web services are deployed in data centers that are located in disparate parts of the World means that these services are managed by different parties and are owned by different providers. Therefore, finding a common third party that is trusted by all providers and that can manage all these services is quite

1. http://www.jonasson.org/maps

challenging. Back to the previous example of grouping Google and Bing in one community, finding a common master for that community is difficult since Bing is owned by Microsoft and Google Maps is owned by Google that are from the geographical point of view separate and from the economic point of view competitors; which makes the decision about the master problematic.

Federated cloud [10] offers a practical platform for joining together services from different clouds for better resource scaling based on providers' agreement. Despite their effectiveness for resource scaling, cloud federations are unable to handle the composition and discovery issues due to the fact that they operate at the provider's (not service) level and focus solely on the IaaS cloud services' layer. Moreover, both traditional community formation and cloud federation models rely on a honest adversary model wherein all services/providers are assumed to be trustworthy. However, in such a multilateral environment whereby multiple clouds and providers are involved, malicious services are likely to exist. Practically, these malicious services may misbehave by unilaterally deviating either from their agreements with other services upon community/federation formation or from the SLAs made with users, with the aim of saving resources and/or gaining advantage over other services. Such malicious services can be referred to as *passive malicious services*. Unlike the active malicious services that launch active attacks (e.g., Denial of Service) to harm other parties, passive malicious services misbehave to illegally maximize their own benefits.

1.2 Contributions

Trust is evolving as one of the hottest, yet challenging, topics in cloud computing [11]. The importance of trust arises mainly because of the unclear, inconsistent, and often uncommitted clauses of the official contracts such as SLAs [12]. A main challenge that encounters trust is how to ascertain the credibility of the trust relationships in the presence of collusion attacks in which attackers collude to give misleading judgments. The purpose of attackers in such a type of attacks is to either promote or demote some other services and deceive hence the trust result. Moreover, assigning initial trust values for the newly deployed services having no past interactions, known as trust bootstrapping, is a major challenge that encounters any trust establishment mechanism. Although many trust frameworks [12], [13], [14] can be found in the literature in the domains of cloud computing and Service-Oriented Computing (SOC), most of these approaches focus solely on one particular issue (particularly the trust aggregation) and ignore some other important issues that encounter any trust framework. In this work, we propose a comprehensive trust framework called DEBT (Discovery, Establishment, and Bootstrapping Trust) that provides a stepwise guide for services to build trust relationships starting from discovering services and collecting feedback, down to bootstrapping new services and aggregating feedback in a collusion-resistant manner. Each step in the proposed trust framework is designed to work both effectively and efficiently, while bridging the gap of the state-of-the-art trust problems. DEBT is designed in a fully distributed manner that eliminates the need for any central entity to regulate the different steps of the trust framework. Specifically, we model trust as a private relationship between each pair of services rather than a public measure. For example, if service S_1 trusts another service S_2 with a belief degree of b, it is not necessary that another service S_3 would trust S_2 with the same belief b.

Based on DEBT, we design a trust-based hedonic coalitional game with nontransferable utility that aims to find the optimal coalition partition that minimizes the number of malicious members. Hedonic games are a type of coalitional games in which players have preferences over the coalitions that they may be member of and the utility of any player in a certain coalition depends *solely* on the *identity* of the members of that coalition regardless of how other services are structured. Although hedonic games have been already used in similar settings [2], [15], our work is the first that employs the hedonic coalitional game in conjunction with the notion of trust for security purposes in the domain of SOC. Moreover, our work is the first in this domain that considers a fully and effectively hedonic setting in the formulation of the utility function. Specifically, the existing approaches satisfy only some of the requirements of hedonic games, namely the use of preference relations for building coalitions and the fact that the utility of any player in a given coalition depends only on the members of that coalition. However, the utility functions used in all these approaches are formulated in such a way to be transferable among players, which contradicts with the definition of hedonic games that are subsets of non-transferable utility games [16]. Moreover, the essence of hedonic games, by which the term *hedonic* is inspired, relies on the idea that players seek to enjoy each other's partnership, apart from numeric considerations. Therefore, we believe that trust is the best candidate to be used for formulating the non-transferable utility function without contradicting with the spirit of hedonic games. Summarizing, the main contributions of this paper are:

- Proposing a polynomial-time services discovery algorithm that enables services to inquire about each other's behavior based on the concept of *tagging* in online social networks.
- Proposing a trust aggregation technique that uses the Dempster-Shafer [17] theory of evidence and advances a credibility update mechanism to ensure obtaining trust results that are resilient to the collusion attacks even when attackers are the majority.
- Proposing a trust bootstrapping mechanism that combines the concept of *endorsement* in online social networks with the decision tree classification technique to assign initial trust values for the newly deployed services.
- Modeling the multi-cloud community formation problem as a trust-based hedonic coalition formation game and proposing a relevant algorithm that converges to a final Nash-stable and individually stable coalition partition of services.

The performance of the proposed game is analyzed both theoretically and experimentally using a real-life cloud services dataset obtained from the CloudHarmony service[2]. Simulation results show that the proposed game is able to produce trustworthy communities that minimize the number of malicious members, while improving their overall performance in terms of availability, throughput, and response time compared to three state-of-the-art cloud federation and services community formation models.

2 Related Work

In this section, we provide a literature review on the concepts of *Web Services Communities* and *Cloud Federations*.

2.1 Communities of Web Services

The concept of community has been extensively investigated in the context of Web services. In [18], the authors envisaged communities as service containers that share functionally-similar services. The aim is to facilitate the composition processes when the number of services is large. Maamar et al. [3] proposed a hierarchical architecture for such communities that has become later a common trend for this concept. In this architecture, a central entity called *master* is responsible for administrating the operations of the *slave* services. In [6], Medjahed et al. considered communities as ontological entities providing generic functions for a set of functionally-similar services. These functions may be customized later based on the underlying purposes. All of the aforementioned approaches focus on the composition process optimization and ignore the incentives that would encourage services

2. http://cloudharmony.com/

to form communities. In [7], Liu et al. investigated a coalitional game for the community formation while accounting for the incentives of the services in that process. The worth of the community is evaluated in terms of price and cost that depend heavily on the availability provided by the services. In a close work, Khosrowshahi-Asl et al. [8] proposed a coalitional game to boost the cooperation among services within communities. The key idea is to guarantee the stability of the communities in the sense that membership requests to the community are accepted only if they do not get the utility of the existing members decreased. In [5], the authors modeled the community formation as a Stackelberg game that differentiates between services on the basis of their advertised parameter such as reputation, market share, and capacity of handling requests.

Overall, the problem of the existing community formation models is their ineffectiveness in a multi-cloud community formation environment. On the one hand, they rely on a centralized architecture in which a central entity coordinates the operations of the community, which contradicts with the distributed nature of cloud-based services. On the other hand, they overlook the malicious services in the formation process whose presence is likely in the multi-cloud environment.

2.2 Cloud Federations

In [10], Rochwerger et al. paved the way for the notion of cloud federations by introducing the concept of RESERVOIR whose main goal is to explore the technologies needed to handle the scalability problem faced by a single provider model. They discussed the notion of cloud federations wherein providers characterized by large capacity of computing infrastructure may lease some of their resources to other providers who lack temporarily for such resources. Goiri et al. [19] addressed the problem of cloud federations from the perspective of increasing providers' profits. They proposed several equations to help providers decide when to outsource resources to other providers, when to insource free resources to other providers, and when to shutdown unused nodes to save power. In [20], Van den Bossche et al. formulated a Linear Programming model to assist providers with deciding which workloads to outsource and to which providers in such a way to maximize the utilization of internal data center, minimize the cost of running outsourced tasks, and keep up high QoS constraints. Recently, game theory has been widely used to address the problem of forming cloud federations. In [21], Niyato et al. proposed a coalitional game among cloud providers. First, a stochastic linear programming game model that takes into account the random internal demand of cloud providers is formulated to analyze the resource and revenue sharing for a certain group of providers. Thereafter, a coalitional game that enables cloud providers to form cooperative groups for resource and revenue sharing is presented. The objective is to exploit the under-utilized resources when the internal demand in cloud data centers is less than the capacity of providers. In [2], Mashayekhy et al. investigated a hedonic coalitional game that focuses on the cooperation among IaaS services to improve resources scaling capabilities. The resources considered are provisioned as VM instances of different types (e.g., small, large, etc.). The objective is to form the federations that yield the highest profit in terms of cost and price of the underlying VMs.

Overall, cloud federations focus exclusively on improving the resource scaling capabilities among IaaS providers. On the other hand, communities is a more general architecture that supports, in addition to resource scaling, discovery, marketing and composition facilitation. Thus, cloud federations may be considered as a subset of services communities. Besides, the existing cloud federations formation models overlook the problem of having malicious services/providers that constitute a serious challenge for the success of such an idea. To the best of our knowledge, our work is the first that accounts for the problem of malicious services in both services' communities and cloud federations formation scenarios.

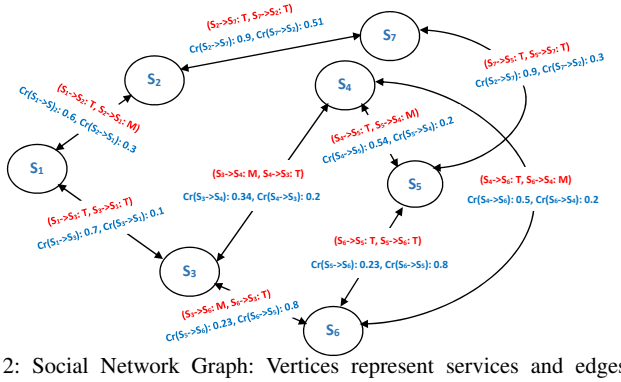


Fig. 2: Social Network Graph: Vertices represent services and edges represent the interactions among services.

3 System Model and Assumptions

In this section, we present and discuss the system and attack models.

3.1 System Model

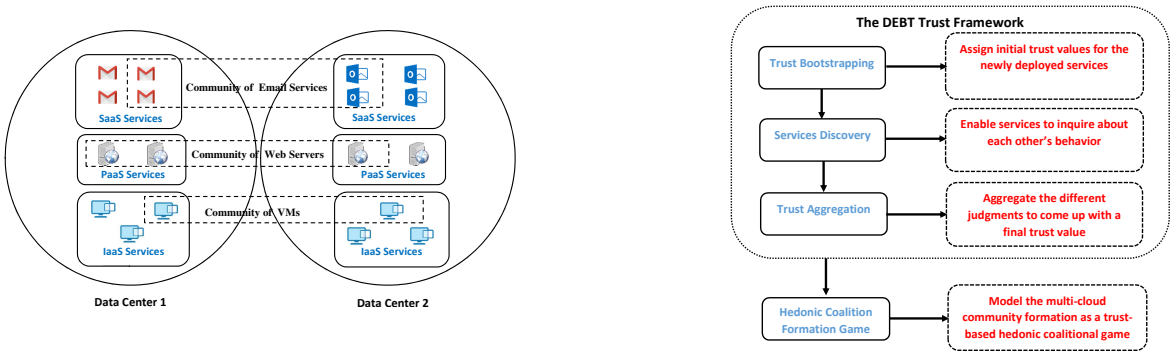
Let $S = \{S_1, ..., S_n\}$ be a finite set of services, where each service $S_i \in S$ is characterized by a set of available resources $R = \{R_1, ..., R_n\}$ (e.g., amount of memory, number of cores, etc.). Let $G = (S, E, J)$ be a directed social network graph representing the relationships between these services, where each edge $(S_i, S_j) \in E$ indicates an interaction between service S_i and service S_j. Thus, if $(S_i, S_j) \notin E$ then services S_i and S_j had no interaction in common. Each edge (S_i, S_j) is associated with a judgement $J(S_i, S_j) \neq J(S_j, S_i) \in \{T, M\}$ that denotes each service's judgment on any other service based on their previous interactions. For example, the judgment pair $(S_1 \rightarrow S_2 : T, S_2 \rightarrow S_1 : M)$ between services S_1 and S_2 in Fig. 2 indicates that S_1 rates S_2 as trustworthy, whereas S_2 rates S_1 as malicious (i.e., $J(S_1, S_2) = T$ and $J(S_2, S_1) = M$). To decide about this judgement, services use the following well-known satisfaction metric [22]:

$$J(S_i, S_j) = \frac{Sat(S_i, S_j)}{Tot(S_i, S_j)}, \tag{1}$$

where $Sat(S_i, S_j)$ denotes the number of interactions between S_i and S_j that S_i considers are satisfactory and $Tot(S_i, S_j)$ denotes the total number of interactions between S_i and S_j.

Based on Eq. (1), if $J(S_i, S_j) > \beta$ then S_i rates S_j as trustworthy, where β is a threshold that is set by each service depending on the type of the other service(s) being evaluated. For example, the interactions with an online payment service would be weighted higher than those of a weather forecasting service. Otherwise, S_j would be rated as malicious. As mentioned earlier, the objective is to form trusted multi-cloud services communities (Fig. 1a) between services geographically distributed across multiple cloud data centers using a distributed trust model. The trust towards a certain service S_j is built by collecting judgments on this service from its direct neighbors $N(S_j)$ in G (i.e., the services that had interacted with S_j). To this end, each service S_i holds a fixed number of inquiries it is allowed to make and denoted by $Inq(S_i)$. Initially, all services have an equal amount of this metric, which is updated later during the trust establishment process (See Section 4). Since services may be either *truthful* or *collusive* in judging the other services, each pair of services $(S_i, S_j) \in S$ has a belief in credibility $(Cr(S_i \rightarrow S_j) = n, Cr(S_j \rightarrow S_i) = m)$ that represents each service's accuracy level in judging the other services, where n and m are two decimal numbers. Based on the collected judgments, each service S_i builds a belief in trustworthiness denoted by $belief_{S_i}^{S_j}(T)$ and a belief in maliciousness denoted by $belief_{S_i}^{S_j}(M)$ for any other service S_j it is interested in forming community with. It's worth noting that such a mechanism does not entail any privacy breach as only the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2549019, IEEE Transactions on Services Computing



(a) Multi-Cloud Community Architecture (b) Methodology

Fig. 1: Model architecture and methodology

value of $J(S_i,S_j)$ is shared between services without revealing any sensitive information such as the volume or type of interactions.

The community formation is modeled as a hedonic coalition formation game, where each coalition $C \subseteq S$ represents a given community[3]. Let Π denote the set that partitions services S into non-empty coalitions and that is referred to as a *coalition structure* [16].

Definition 1 (Coalition Structure). A *coalition structure* or *partition* is a set of coalitions $\Pi = \{C_1,...,C_S\}$ that splits the set of services S into disjoint coalitions such that $\forall l \neq l', C_l \cap C'_l = \emptyset$ and $\bigcup_{k=1}^{l} C_k = S$. The coalition to which service S_i belongs is denoted by $C_l^{S_i}$.

Let $U_{S_i}(C_k)$ denote the utility of service S_i in a certain coalition $C_k \in \Pi$. $U_{S_i}(C_k)$ is obtained by summing up S_i's beliefs in trustworthiness in C_k's members. Thus, $U_{S_i}(C_k)$ is computed as follows:

$$U_{S_i}(C_k) = \sum_{S_j \in C_k} belief_{S_i}^{S_j}(T) \quad (2)$$

The methodology followed in the rest of the paper is depicted in Fig. 1b.

3.2 Attack Models and Assumptions

Attacks may occur in our model either (1) during trust establishment and/or (2) during and after communities formation. During trust establishment, collusion attacks may take place between services to mislead the results. During and after communities formation, passive malicious services may misbehave to save their resources and gain illegal advantage over the other services. At this stage, we consider only passive attacks in the sense that the malicious services considered during and after communities formation are assumed to misbehave in order to increase their own benefits without having a direct intention to harm the other services and/or communities. Therefore, active attacks such as Sinkhole and Denial of Service (DoS) are beyond the scope of this work. In particular, we consider the following types of attacks:

- **Collusion Attacks.** Such attacks occur when several malignant services collaborate together to give misleading judgments either to increase the trust score of some services (i.e., a promoting attack) or to decrease the trust score of some other services (i.e., a slandering attack). Note that this type of attacks cannot occur in a non-collusive way in the sense that a single malicious service cannot submit multiple misleading judgements to conduct promoting attack and/or slandering attacks as judgments are given upon request in our trust framework.
- **Passive Attacks.** Such attacks occur when passive malicious services cheat about their available resources and/or QoS potential during communities' formation in order to increase their chances of being grouped into powerful communities. After communities are formed, these malicious services would renege on their agreements with both services and clients by benefiting from the other services' resources (e.g., physical computing infrastructure) and refraining from sharing their own resources to dedicate them for their own workload.

3. In the rest of the paper, the terms *coalition* and *community* are used interchangeably.

4 The DEBT Trust Framework

In this section, we present the details of our proposed trust framework.

4.1 Service Discovery

Algorithm 1: Services Discovery Algorithm

```
1:  Input: Source node src
2:  Input: Destination node d
3:  Output: Set of direct neighbors of d, N(d).
4:  procedure SERVICEDISCOVERY
5:      s = src
6:      Create an empty tag instance t = ⟨⟩
7:      for each not explored node y ∈ N(s) do
8:          Mark y as explored
9:          if s ∉ t
10:             Append s to t
11:         end if
12:         Append y to t
13:         if (y,d) ∈ E then
14:             send t to src
15:             empty the tag instance
16:         else
17:             s = y
18:         end if
19:         if all y ∈ N(S) are explored then
20:             s = src
21:         end if
22:     end for
23:     Append the last element of the tag instance to N(d)
24:     return N(d)
25: end procedure
```

In order to establish trust between services in the social network (Fig. 2), judgments should be collected first. Therefore, we propose a discovery algorithm that allows services to inquire about each other from their direct neighbors (i.e., the services that had dealt with). The proposed algorithm capitalizes on the concept of *tagging* in online social networks (e.g., facebook, LinkedIn) to explore the direct

neighbors of a certain service. The basic idea is to keep tagging or nominating intermediate services until identifying all the reachable direct neighbors of the intended service. Let us consider the case of service s that asks service x about its judgment on service d. If x has a direct interaction (i.e., edge) with d, it reports its judgment directly to s. Otherwise, it tags its neighbors that had such an interaction (if any) or those that may introduce s to some other services that had such an interaction with d. The details of the algorithm are explained in Algorithm 1. The inputs of the algorithm are a source node *src* (the service that inquires about another service) and a destination node d (the service about which judgements are being collected) in a social network graph (lines 1-2). The output of the algorithm is the set of all reachable direct neighbors of d (line 3). The algorithm creates an empty tag instance (line 6) and loops over the direct neighbors of the source node one by one and marks them as explored (lines $7-8$). This has the advantage of avoiding the revisit of any already visited node. Each explored neighbor gets appended to the tag instance (lines 9-12). If the neighbor has a direct edge with the destination service d, then the tag instance is directly returned to the source node and the tag instance is emptied to start a new tagging process (lines 13-15). Otherwise, the algorithm loops recursively over the "neighbors of the neighbors" and adds them to the tag instance in case they have a direct edge with d. This is done by recursively assigning each neighbor the role of the source node (line 17). This process stops at the level of a node that reports either a judgement to the source node or reports neither a judgement nor a tag instance. Thus, each tag instance can be considered as a sequence of services $\langle S_i, S_{i+1}, ..., S_n \rangle$ starting from a direct neighbor S_i of a certain requestor service S_0 and leading to one or more neighbor(s) (i.e., S_{i+1},..., S_n) of a destination service S_{n+1}.

The motivations for the services to be socially active are three-fold. First, this improves their ability to select the suitable partners to collaborate with based on their previous experience; particularly in case of compositions. In fact, the overall quality of a composite service is influenced by the quality offered by each single service in that composition. Therefore, each service is interested in selecting the appropriate partners in such a way that allows it maintaining a good record among other services offering similar functionalities. Second, by maintaining networks of contacts, the service may learn about the non-functional properties of its peers to adjust its performance accordingly in such a way that increases its competitiveness in the market. Third, participating in the tagging process increases the number of inquiries that each service can make from other services, which increases hence the chances of the service in participating in further communities. Thus, if the service is repeatedly not available for tagging, then it will end up being unable to participate in further communities (See Section 4.2).

As a practical example of why keeping track of neighbors is useful for both services and users, suppose that Bob wants to get the travel and accommodation options from Paris to New York, translate the information from English to French, and send it by SMS to his friend Alain. To this end, Bob uses a composition of FlightBooking service $F1$, HotelReservation service $H1$, Translation service $T1$, and Messaging service $M1$. Suppose also that Bob wants to get the travel and accommodation options from Montreal to New York and send this information by email to another friend Alex. To do so, Bob uses a composition of FlightBooking service $F1$, HotelReservation service $H1$, and Email service $E1$. Here, Bob is using the combination of the FlightBooking service $F1$ and HotelReservation service $H1$ for the second time without having an idea about how this combination has performed in the first transaction. However, if $F1$ has kept track of its previous interaction with $H1$, then it could have advised Bob not use $H1$ for the second time because of a previous bad performance and moreover it could have tagged another hotel reservation service $H2$ to be used instead. Since end-users deal with composite services as a monolithic entity, Bob would refrain from using $F1$ in the future although the performance trouble came from $H1$. Thus, keeping track of the interactions with the other services could have saved $F1$ from being replaced by another service and could have made Bob enjoy high-quality transactions.

As for the complexity of Algorithm 1, the algorithm is a variation of the Breadth-First Search (BFS) strategy [23] in graph theory. Therefore, the computational complexity of Algorithm 1 is $O(|S| + |E|)$ since the worst case would be when all the vertices and edges have to be visited exactly once, where $|S|$ is the number of services in the social network graph and $|E|$ denotes the number of interactions between them. The storage overhead of the algorithm is $O(|S|)$ since the worst case occurs when all the services would need to be held in the tag instance.

4.2 Trust Establishment

Having discussed the discovery algorithm in the previous section, the next step is to establish the trust relationships between services. As mentioned earlier, trust is constructed by collecting judgments about services based on their previous interactions. This trend for establishing trust is very common in the field of trust and reputation and is referred to as recommendation-based or feedback-based trust establishment mechanism [24]. The power of this mechanism stems from its ability to produce meaningful judgements by considering the opinions of multiple parties. Nonetheless, several challenges encounter such a mechanism in the real-world scenarios [24]. Practically, services may be tempted to engage in some collusion scenarios and provide dishonest judgments, which leads to misleading trust results. Moreover, these services usually tend to refrain from revealing their opinions lack of incentives for doing so, which leads to meaningless or biased computations of the aggregate trust value. To tackle these problems, we propose (1) an aggregation model for the collected judgments that is able to overcome the collusion attacks even when attackers are the majority, and (2) an incentive model for the services to motivate them to participate in the trust establishment process.

That is, the aggregation technique should take into account the existence of colluding services. Therefore, simplistic combination techniques such as averaging and majority voting are unsuitable for the considered problem. To address this challenge, we propose an aggregation technique based on the Dempster-Shafer theory of evidence. Dempster-Shafer [25] is a mathematical theory that combines evidences from independent sources to come up with a degree of belief regarding a certain hypothesis. Dempster-Shafer is well-suited for the considered problem for two main reasons [17]: (1) unlike the Bayesian approach that demands complete knowledge of both prior and conditional probabilities, Dempster-Shafer can represent uncertainty or lack of complete knowledge, and (2) it provides a powerful rule for combining observations from multiple (possibly unreliable) parties. The first property is important to guarantee the fairness in the trust aggregation process as some services may misbehave due to some out of control circumstances (e.g., problems in the physical infrastructure they are hosted on) and not as a result of some malicious behavior. For example, in Dempster-Shafer, if a service A confirms that service B is trustworthy with probability p, it does not follow that B is malicious with probability $1-p$ as is the case in the Bayesian inference. Hence, A would have p degree of belief in the trustworthiness of B and 0 degree of belief in B's maliciousness. The second property is important to prevent colluding services from misleading the final aggregate trust value. It is worth noting that Dempster-Shafer has been already used for trust establishment in multi-agent systems [26], [27]. The difference between these approaches and our approach is that the latter requires no threshold for deciding whether to trust another agent or not. Besides, we propose in our model a credibility update function and link the credibility scores of the services with the number of inquiries that they are allowed to make with the aim of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2549019, IEEE Transactions on Services Computing

encouraging services to participate in the trust establishment process and provide truthful judgments. Moreover, in the referred approaches if no information about the newcomer agents may be obtained then these agents are deemed to have no reputation at all, which may end up overlooking these agents in future community formation processes in the presence of other well-reputable agents. To handle this issue, we propose in the next section a bootstrapping mechanism to assign initial trust values for such services.

The proposed aggregation technique works as follows. Let $\Omega = \{T, M, U\}$ denote a set consisting of three hypotheses. T denotes that a certain service X is trustworthy; M denotes that X is malicious; and U denotes that X is either trustworthy or untrustworthy to express the uncertainty or partial knowledge in the decisions. The basic probability assignment (bpa) of a service S in judging another service S', denoted by $m_S^{S'}$, defines a mapping function of the set Ω to a real-valued interval bounded by 0 and 1, i.e., $m_S^{S'} : \Omega \longmapsto [0,1]$. In our framework, the bpa for a certain hypothesis is equal to the credibility score believed on the service giving the judgement. In other words, suppose that service S is asked by service q to judge another service S', where q has a belief in S's credibility equal to α. Assume that S claims that S' is trustworthy, then the bpa's of S would be: $m_S^{S'}(T) = \alpha$, $m_S^{S'}(M) = 0$, and $m_S^{S'}(U) = 1 - \alpha$. On the other hand, if S reports that S' is malicious, then the bpa's of S would be: $m_S^{S'}(T) = 0$, $m_S^{S'}(M) = \alpha$, and $m_S^{S'}(U) = 1 - \alpha$. It is worth noting that, throughout the paper, when S' is unique or understood from the context, we simplify by writing m_S and when both S and S' are understood from the context, we simply write m. To aggregate the different evidences (i.e., bpa's), a belief function is used. The belief function represents the total bpa's supporting a given hypothesis H and maps H to a real-valued number between 0 and 1. The belief function of service S in service S' regarding a certain hypothesis H (where $H = T, M$, and U respectively) after inquiring two other services 1 and 2 is given as follows:

$$bel_S^{S'}(T) = m_1(T) \oplus m_2(T) = \frac{1}{K}[m_1(T)m_2(T) + m_1(T)m_2(U) + m_1(U)m_2(T)] \tag{3}$$

$$bel_S^{S'}(M) = m_1(M) \oplus m_2(M) = \frac{1}{K}[m_1(M)m_2(M) + m_1(M)m_2(U) + m_1(U)m_2(M)] \tag{4}$$

$$bel_S^{S'}(U) = m_1(U) \oplus m_2(U) = \frac{1}{K}[m_1(U)m_2(U)] \tag{5}$$

where:

$$K = \sum_{h \cap h' = \emptyset} m_1(h)m_2(h') \tag{6}$$

Thus, the problem is turned into computing the beliefs in trustworthiness $bel_S^{S'}(T)$ and maliciousness $bel_S^{S'}(M)$ of service S in service S'.

Theorem 1. *The proposed aggregation technique overcomes the collusion attacks even when attackers are the majority, if the credibility scores of the truthful raters are higher than those of colluding raters.*

Proof. For simplicity and without loss of generality, suppose that three services A, B, and C are asked by service S to judge another service D. Assume that D is trustworthy and that services A and C collude together to demote D by claiming the contrary, whereas B is truthful and reports that D is trustworthy. Assume as well that the credibility scores of A, B, and C believed by S are $Cr(S \to A) = \frac{\alpha}{2}$, $Cr(S \to B) = \alpha$, and $Cr(S \to C) = \frac{\alpha}{3}$ respectively such that $Cr(S \to B) > Cr(S \to A) > Cr(S \to C)$. As mentioned earlier, A and C claim unjustly that D is malicious whereas A reports that D is trustworthy. Thus, the bpa's of the three services are given as follows:

- **Service** A: $m_A(T) = 0$, $m_A(M) = \frac{\alpha}{2}$, and $m_A(U) = 1 - \frac{\alpha}{2}$.
- **Service** B: $m_B(T) = \alpha$, $m_B(M) = 0$, and $m_B(U) = 1 - \alpha$.
- **Service** C: $m_C(T) = 0$, $m_C(M) = \frac{\alpha}{3}$, and $m_C(U) = 1 - \frac{\alpha}{3}$.

The theorem may be proved by contradiction. Thus, assuming that the theorem does not hold, we should get that S's belief in D's maliciousness is higher than its belief in D's trustworthiness based on the judgments of the colluding services A and C and that are the majority, i.e.,

$$bel_S^D(M) > bel_S^D(T) \tag{*}$$

Let's start by combining the bpa's of A and B as per Table 1.

Table 1: Combination of the bpa's of services A and B

A \ B	$m_B(T) = \alpha$	$m_B(M) = 0$	$m_B(U) = 1 - \alpha$
$m_A(T) = 0$	0	0	0
$m_A(M) = \frac{\alpha}{2}$	$\frac{\alpha^2}{2}$	0	$\frac{\alpha - \alpha^2}{2}$
$m_A(U) = 1 - \frac{\alpha}{2}$	$\frac{2\alpha - \alpha^2}{2}$	0	$\frac{\alpha^2 - 3\alpha + 2}{2}$

Note that the cell values in Table 1 are obtained by multiplying the corresponding rows and columns. Now, let's compute the combined beliefs of services A and B.

- $K = m_A(T)m_B(T) + m_A(T)m_B(U) + m_A(U)m_B(T) + m_A(M)m_B(M) + m_A(M)m_B(U) + m_A(U)m_B(M) + m_A(U)m_B(U) = \frac{-\alpha^2+2}{2}$.

  We abuse the notation of the function m and define $m_{AB}(T)$, $m_{AB}(M)$, and $m_{AB}(U)$ as follows:

- $m_{AB}(T) = m_A(T) \oplus m_B(T) = 1/K[m_A(T)m_B(T) + m_A(T)m_B(U) + m_A(U)m_B(T)] = \frac{-2\alpha^2+4\alpha}{-2\alpha^2+4}$.
- $m_{AB}(M) = m_A(M) \oplus m_B(M) = 1/K[m_A(M)m_B(M) + m_A(M)m_B(U) + m_A(U)m_B(M)] = \frac{-2\alpha^2+2\alpha}{-2\alpha^2+4}$.
- $m_{AB}(U) = m_A(U) \oplus m_B(U) = 1/K[m_A(U)m_B(U)] = \frac{2\alpha^2-6\alpha+4}{-2\alpha^2+4}$.

Then, we combine in Table 2 the combined belief of services A and B with the bpa of service C. By computing the combined beliefs

Table 2: Combination of services A and B's belief with the bpa of C

AB \ C	$m_C(T) = 0$	$m_C(M) = \frac{\alpha}{3}$	$m_C(U) = 1 - \frac{\alpha}{3}$
$m_{AB}(T) = \frac{-2\alpha^2+4\alpha}{-2\alpha^2+4}$	0	$\frac{-2\alpha^3+4\alpha^2}{-6\alpha^2+12}$	$\frac{2\alpha^3-10\alpha^2+12\alpha}{-6\alpha^2+12}$
$m_{AB}(M) = \frac{-2\alpha^2+2\alpha}{-2\alpha^2+4}$	0	$\frac{-2\alpha^3+2\alpha^2}{-6\alpha^2+12}$	$\frac{2\alpha^3-8\alpha^2+6\alpha}{-6\alpha^2+12}$
$m_{AB}(U) = \frac{2\alpha^2-6\alpha+4}{-2\alpha^2+4}$	0	$\frac{2\alpha^3-6\alpha^2+4\alpha}{-6\alpha^2+12}$	$\frac{-2\alpha^3+12\alpha^2-22\alpha+12}{-6\alpha^2+12}$

of services A, B, and C, we get:

- $K = \frac{2\alpha^3-10\alpha^2+12}{-6\alpha^2+12}$.
- $m_{AB}(T) \oplus m_C(T) = \frac{-12\alpha^5+60\alpha^4-48\alpha^3-120\alpha^2+144\alpha}{-12\alpha^5+60\alpha^4+24\alpha^3-192\alpha^2+144}$.
- $m_{AB}(M) \oplus m_C(M) = \frac{-12\alpha^5+72\alpha^4-36\alpha^3-144\alpha^2+120\alpha}{-12\alpha^5+60\alpha^4+24\alpha^3-192\alpha^2+144}$.
- $m_{AB}(U) \oplus m_C(U) = \frac{12\alpha^5-72\alpha^4+108\alpha^3+72\alpha^2-264\alpha+144}{-12\alpha^5+60\alpha^4+24\alpha^3-192\alpha^2+144}$.

We have that $\sum_{x \in \{T,M,U\}} m_{AB}(x) \oplus m_C(x) = 1$ and $m_{AB}(T) \oplus m_C(T) > m_{AB}(M) \oplus m_C(M)$ for every $0 < \alpha \leq 1$; meaning that $bel_S^D(T) > bel_S^D(M)$ for every $0 < \alpha \leq 1$, which contradicts with (*). Thus, S's belief in D's trustworthiness exceeds its belief in D's maliciousness although the majority of raters (i.e., A and C) colluded to state the contrary. Generally speaking, the proposed aggregation technique overcomes the collusion attacks even when attackers are the majority if and only if (1) the credibility values are between 0 and 1, and (2) the credibility scores of the trustworthy raters are higher than those of colluding ones. Formally, let:

- V_T: denote the set of truthful services participating in the judgement process.
- V_C: denote the set of colluding services participating in the judgement process.
- V: denote the set of services participating in the judgement process, i.e., $V = V_T \cup V_C$.

The proposed aggregation technique overcomes the collusion attacks even when attackers are the majority, i.e., $|V_C| \geq |V_T|$ if and only if:

$$\forall v \in V,\ 0 < Cr(v) \leq 1 \tag{7}$$

$$\forall\ t\ \in\ V_T\ and\ c\ \in\ V_C,\ Cr(t)\ >\ Cr(c) \tag{8}$$

□

Obviously, the performance of the aggregation technique depends heavily on the credibility scores assigned to the services. Thus, maintaining healthy values of this metric is a building block for achieving truthful decisions. Therefore, the credibility metric should be updated continuously to ensure that truthful services always hold higher credibility scores than those of colluding ones. That is, the credibility scores of the services that report truthful judgments should be promoted and those of the colluding services should be demoted by the service requesting the judgement after each aggregation process. Eq. (9) depicts the credibility update function for each service x after having participating in judging service y in favor of service s.

$$Cr(s \rightarrow x) = \begin{cases} \min(1, Cr(s \rightarrow x) + |Z - Cr(s \rightarrow x)|), & \text{if C 1} \\ \left|Cr(s \rightarrow x) - \min(belief_s^y(T), belief_s^y(M))\right|, & \text{if C 2} \end{cases} \tag{9}$$

where $Z = \max(belief_s^y(T), belief_s^y(M))$ and $C1$ and $C2$ are two conditions such that:

C 1. $J(x,y) \in \{T\}$ & $belief_s^y(T) > belief_s^y(M)$ *or* $J(x,y) \in \{M\}$ & $belief_s^y(T) < belief_s^y(M)$

C 2. $J(x,y) \in \{T\}$ & $belief_s^y(T) < belief_s^y(M)$ *or* $J(x,y) \in \{M\}$ & $belief_s^y(T) > belief_s^y(M)$

The intuition behind Eq. (9) is that truthful services whose judgments agree with the winner belief receive a reward that is equal to the difference between their current credibility scores and the value of that belief. For the untruthful services whose judgments disagree with the winner belief, they undergo a decrease in their credibility scores that is equal to the value of the loser belief. The idea is to avoid harsh punishments in one round of aggregation. In this way, only the services that repeatedly report misleading judgments will get their credibility scores drastically decreased.

Finally, services should receive rewards for their participation in both the services discovery and trust aggregation processes. This reward is important to motivate further participation from these services. To this end, we link the number of inquiries that a service is allowed to make about other services with its credibility score and the number of tags it has made and consequently its level of participation in the trust framework. The reward function is given by Eq. (10), where x is a given service being rewarded by service s for which it has provided judgement, $Inq(x \rightarrow s)$ denotes the total number of inquiry requests that x is allowed to make from s, $|Tags(x \rightarrow s)|$ denotes the number of neighbors tagged by x in favor of S, and 1 represents a fixed reward for x for providing its own judgment.

$$Inq(x \rightarrow s) = Inq(x \rightarrow s) + (|Tags(x \rightarrow s)| + \lceil |Tags(x \rightarrow s)| \times Cr(s \rightarrow x) \rceil + 1) \tag{10}$$

In this way, services would tend to contribute in the trust framework in order to increase their total number of inquiry requests that they can make and be able hence to participate in the coalition formation process. Moreover, by linking the number of possible inquiries with the credibility scores of the services, we are encouraging services to provide truthful judgment in order to increase their credibility scores and increase hence their share of inquiry requests. Over the time, colluding services will get their number of possible inquiry requests drained, which makes them unable to participate in further coalition formation processes.

4.3 Trust Bootstrapping

Trust bootstrapping, i.e., assessing trust for newly deployed Web services, is a major issue that encounters our trust framework as no historical information about newcomers may be available. For example, when a service is initially registered in a cloud datacenter, no service has interacted with it and hence there is no record of its former behavior. As a result, its initial trust cannot be evaluated, which may lead to overlook this service in the future coalition formation processes. Therefore, a mechanism to allocate initial trust values for newly deployed services in the absence of historical information about their past behavior is needed.

The existing bootstrapping mechanisms in the SOC domain can be classified into three main categories [28]: (1) Default-value-based, (2) punishment-based, and (3) adaptive mechanisms. The first approach assigns a default trust value for all new services. A self-evident drawback that encounters this approach is that it can arbitrarily favor either the already existing services or the newly deployed ones. Specifically, if the assigned default trust value is low, newly deployed services will be overlooked in the future coalition formation processes. On the other hand, if the assigned default trust value is high, newcomer services are favored over existing services that may have interacted and strived to achieve their trust values. This motivates malicious providers to continuously publish new identities to clear the past bad trust history of their services and gain high trust scores (a.k.a white-washing). As a remedy to white-washing, the punishment strategy proposes to assign low initial trust values for the newcomer services. In this way, however, the new services are disadvantaged since they will have no chance to make interactions and gain trust. In the adaptive strategy, the newcomer service is bootstrapped based on the rate of maliciousness in the community in which it registers. More specifically, this strategy assumes a community-based architecture in which a community groups a number of services sharing the same functionality to simplify the bootstrapping process. When a new service is registered in a certain community, it gets a trust value based on the rate of maliciousness in that community. However, the problem with this strategy is that it still allows malicious services to benefit from a low rate of maliciousness by leaving the network and rejoining again to obtain higher trust scores.

In this paper, we propose a bootstrapping mechanism, based on the concept of endorsement in online social networks, that is resilient to white-washing. As mentioned earlier, each service maintains a dataset that records its previous interactions with several services having different functional and non-functional specifications. Whenever a request from service i to bootstrap a new service j is received, the services that are interested in the request train a decision tree classifier on their datasets to predict an initial trust value for j. A decision tree is a classification technique that recursively and repeatedly partitions the training set into subsets based on an attribute value test until all the samples at a given node belong to the same class or until splitting adds no more value to the classifications. Test attributes are selected based on some heuristic or statistical measure (e.g., information gain) that determines their relative importance in discriminating between classes being learned. Unlike some other classification techniques such as support vector machines and neural networks that entail high time and space complexity, the main advantages that make decision tree suitable for our bootstrapping problem are mainly its intuitive simplicity and computational time and space efficiency [29], which is important for resource-constrained nodes such as Web services.

The decision tree classifier analyzes services' training dataset that contains properties and specifications for some existing services (e.g., provider's name, deployment country, etc.) and learns the patterns of the data by pairing each set of inputs with the expected output (e.g., the judgment on the services). To create the training and test sets, bootstrappers use the k-fold cross-validation with $k = \gamma$, where γ

represents the number of folds. In this method, the dataset is split into k subsets, each of which is used each time as test set and the other $k-1$ subsets are merged together to compose the training set. The accuracy is then computed by averaging the error across all the k trials. This method has the advantage of reducing the bias of the classification results on the way based on which data is being divided due to the fact that each data point will be part of the test set exactly once and part of the training set $k-1$ times. Obviously, the computational complexity of this method grows as the size of k increases. Thus, the choice of γ would vary from one service to another depending on the available resources that each bootstrapper decides to dedicate to the bootstrapping process. Bootstrappers use the learned classifier to predict an initial judgment for the services being bootstrapped. Based on the classification results, each service may endorse the underlying services either positively or negatively. Nonetheless, this does not constitutes the final judgement. In fact, judgements from all bootstrappers are aggregated again by the bootstrapping requestor using Dempster-Shafer as described in the previous section to come up with a combined belief that is resilient to unreliable endorsements from colluding bootstrappers. Note that the results of the aggregation are not influenced by the number of bootstrappers even when this number is minimal since Dempster-Shafer is independent from the number of incoming observations. The extremely worst cases in this regard would be when only one service participates in the bootstrapping process or even no service is able to participate. In the former case, the requestor may rely on the opinion of the bootstrapper without having to use Dempster-Shafer for aggregation. In the latter case, the requestor may bow to reality and use random guessing or default-value-based techniques. It's worth signaling that the bootstrapping process is voluntary for bootstrappers in the sense that each service has the right to decide whether to train its classifier or not after each inquiry request received based on its available resources and whether to endorse services or not based on the underlying accuracy. This aspect is important to guarantee the fairness for both bootstrapper and bootstrapped services. In fact, after training the classifier and computing classifications' accuracy, some services may notice that they have no sufficient accuracy to make judgments lack of similarity between the properties of the services that they had dealt with and the services being bootstrapped. Therefore, these services are better off refraining from submitting inaccurate endorsements.

Finally, the credibility scores of the bootstrappers are updated by the bootstrapping requestor according to Eq. (9). The credibility update has two main advantages. On the one hand, it motivates the services having high levels of classification accuracy to participate in the bootstrapping mechanism to get their credibility scores increased. On the other hand, it demotivates the malicious services from submitting false endorsements to illegally promote some services or demote some other services and exclude them from future competitions.

4.4 Illustrative Example

In this section, we discuss an illustrative example to show how our proposed trust framework works practically. Consider the social network graph in Fig. 2 and assume that service S_1 wants to establish a trust relationship toward service S_5. To do so, it has first to discover all of S_5's reachable direct neighbors using Algorithm 1. Thus, it creates an empty tag instance t (line 6), contacts its one-edge away neighbors S_2 and S_3 one by one, and marks them as explored for the current tag instance (line 8). Starting with S_2, this latter gets appended directly to the tag instance (i.e., $t = \langle S_2 \rangle$) (line 12) and checks whether it does have a direct edge with S_5 (line 13). Since this is not the case, the algorithm recursively assigns the role of the source node to S_2 in lieu of S_1 (line 17) to start inquiring its direct neighbor S_7 about S_5. S_7 gets marked as explored, gets appended to the tag instance (i.e., $t = \langle S_2, S_7 \rangle$), and checks whether it does have a direct edge with S_5. As this is the case, then the tag instance $t = \langle S_2, S_7 \rangle$ is returned to S_1 through the path $S_7 \rightarrow S_2 \rightarrow S_1$ (line 14) and set to empty to restart the tagging process (line 15). Since all of S_2's direct neighbors have been explored (lines 19-21), the algorithm moves to S_1's second neighbor S_3. S_3 gets appended to the empty tag instance (i.e., $t = \langle S_3 \rangle$) and starts inquiring its direct neighbors S_4 and S_6 one by one. Starting with S_4, the node gets appended to t (i.e., $t = \langle S_3, S_4 \rangle$) and gets marked as explored. It check then whether it does have a direct edge with S_5. Since such an edge exists, the tag instance $t = \langle S_3, S_4 \rangle$ is returned to S_1 through the path $S_4 \rightarrow S_3 \rightarrow S_1$ and the tag instance is emptied again. Since not all of S_3's direct neighbors have been explored yet, the algorithm moves to S_3's second direct neighbor S_6 and repeats the same process that took place with S_4, where the tag instance $t = \langle S_3, S_6 \rangle$ gets returned to S_1 through the path $S_6 \rightarrow S_3 \rightarrow S_1$. Since all of S_3's direct neighbors have been explored now, the algorithm moves to checks for any additional neighbor of S_1 and stops after noticing that all the neighbors of S_1 have been explored. In this example, only 5 services (i.e., S_2, S_3, S_4, S_6, S_7) out of 7 and 8 edges (i.e., $S_1 \leftrightarrow S_2$, $S_1 \leftrightarrow S_3$, $S_2 \leftrightarrow S_7$, $S_7 \leftrightarrow S_5$, $S_3 \leftrightarrow S_4$, $S_3 \leftrightarrow S_6$, $S_4 \leftrightarrow S_5$, $S_6 \leftrightarrow S_5$) out of 9 have been explored. Therefore, the time complexity of the tagging process is linear in the number of services and edges in the social network graph thanks to the fact that each service and edge will be visited once in the worst case. In the example as well, 3 paths having each a size of 2 are returned in tagging instances; so in total $2 \times 3 = 6$ services are stored. This confirms that the space complexity, in its turn, is linear in the number of services.

As a second step, S_1 has to aggregate the judgments of S_5's direct neighbors, namely S_4, S_6, and S_7. Ostensibly, all of these three services would report that S_5 is trustworthy as depicted in Fig. 2. However, assume that S_4 and S_6 decide to collude and perform a slandering attack against S_5 by claiming both that this latter is malicious. As explained before, the judgement of each service is weighted based on its credibility score. Thus, the beliefs of the three services would be:

- **Service** S_4: $m_{S_4}(T) = 0$, $m_{S_4}(M) = 0.34$, and $m_{S_4}(U) = 0.66$.
- **Service** S_6: $m_{S_6}(T) = 0$, $m_{S_6}(M) = 0.23$, and $m_{S_6}(U) = 0.77$.
- **Service** S_7: $m_{S_7}(T) = 0.9$, $m_{S_7}(M) = 0$, and $m_{S_7}(U) = 0.1$.

First, let's combine the beliefs of the services S_4 and S_7. The details are explained in Table 3.

Table 3: Combination of S_4 and S_7's beliefs

S_4 \ S_7	$m_{S_7}(T) = 0.9$	$m_{S_7}(M) = 0$	$m_{S_7}(U) = 0.1$
$m_{S_4}(T) = 0$	0	0	0
$m_{S_4}(M) = 0.34$	0.306	0	0.034
$m_{S_4}(U) = 0.66$	0.594	0	0.066

- $K = m_{S_4}(T)m_{S_7}(T) + m_{S_4}(T)m_{S_7}(U) + m_{S_4}(U)m_{S_7}(T) + m_{S_4}(M)m_{S_7}(M) + m_{S_4}(M)m_{S_7}(U) + m_{S_4}(U)m_{S_7}(M) + m_{S_4}(U)m_{S_7}(U) = 0.694$.
- $m_{S_4}(T) \oplus m_{S_7}(T) = 1/K[m_{S_4}(T)m_{S_7}(T) + m_{S_4}(T)m_{S_7}(U) + m_{S_4}(U)m_{S_7}(T)] = \frac{0.594}{0.694} = 0.856$.
- $m_{S_4}(M) \oplus m_{S_7}(M) = 1/K[m_{S_4}(M)m_{S_7}(M) + m_{S_4}(M)m_{S_7}(U) + m_{S_4}(U)m_{S_7}(M)] = \frac{0.034}{0.694} = 0.049$.
- $m_{S_4}(U) \oplus m_{S_4}(U) = 1/K[m_{S_4}(U)m_{S_7}(U)] = \frac{0.066}{0.694} = 0.095$.

Then, we combine in Table 4 the combined beliefs of S_4 and S_7 with the beliefs of S_6.

Table 4: Combining S_4 and S_7's combined beliefs with the beliefs of S_6

S_6 \ S_4S_7	$m_{S_4S_7}(T) = 0.856$	$m_{S_4S_7}(M) = 0.049$	$m_{S_4S_7}(T) = 0.856 = 0.095$
$m_{S_6}(T) = 0$	0	0	0
$m_{S_6}(M) = 0.23$	0.19688	0.01127	0.02185
$m_{S_6}(T) = 0.77$	0.65912	0.03773	0.07315

- $K = 0.8031277$.
- $bel_{S_1}^{S_5}(T) = m_{S_4S_7}(T) \oplus m_{S_6}(T) = 0.821$.
- $bel_{S_1}^{S_5}(M) = m_{S_4S_7}(M) \oplus m_{S_6}(M) = 0.088$.

- $bel_{S_1}^{S_5}(U) = m_{S_4 S_7}(U) \oplus m_{S_6}(U) = 0.091$.

Thus, S_1's belief in S_5's trustworthiness is still high (i.e., 0.821) although the majority of services colluded to claim the contrary.

Having computed the beliefs in S_5, S_1 should now update its credibility beliefs towards the services that have participated in the trust establishment process. Based on Eq. (9), S_1's credibility belief towards S_4 would decrease to become: $Cr(S_1 \rightarrow S_4) = |0.34 - 0.088| = 0.252$ as its judgment does not agree with the computed belief. Similarly, S_1's credibility belief towards S_6 would decrease down to: $Cr(S_1 \rightarrow S_6) = |0.23 - 0.088| = 0.142$. On the other hand, the credibility towards the truthtelling service S_7 would increase up to: $Cr(S_1 \rightarrow S_7) = 0.9 + |0.821 - 0.9| = 0.979$. As a reward for tagging neighbors, S_2 (assuming that it was able to make one inquiry from S_1 before the tagging), that has tagged one of its neighbors to S_1, gets the number of inquiries it is able to make from S_1 increased up to $Inq(S_2 \rightarrow S_1) = 1 + (1 + \lceil 1 * 0.6 \rceil) + 1 = 4$ as per Eq. (10). S_3 (assuming that it was able to make one inquiry from S_1 before the tagging), that has tagged two of its neighbors to S_1, gets the number of inquiries it is able to make from S_1 increased up to $Inq(S_3 \rightarrow S_1) = 1 + (2 + \lceil 2 * 0.7 \rceil) + 1 = 6$.

5 Trust-based Hedonic Coalitional Game

In this section, we model the problem of forming trusted multi-cloud communities as a hedonic coalitional game with non-transferable utility, propose the appropriate preference function, and analyse the properties of the game.

5.1 Game Formulation

A coalitional game is a game-theoretical model that analyzes the interactions among players when they gather into groups. The output of the coalitional game is a partition of the players' set into coalitions. For the proposed game, the players are the services that seek to form multi-cloud communities. The objective is to form trusted communities wherein the number of malicious members is minimal. Coalitional games may be either cohesive or non-cohesive games. In cohesive games, the motivation for coalescing is extreme [30]. That is, the formation of the single coalition that includes all the players referred to as *grand coalition* is the best for all the players. In such situations, the *grand coalition* can yield outcomes that are at least as attractive for every player as those realizable by any partition of the players into sub-coalitions. In contrary, non-cohesive games are interested in studying situations wherein forming the grand coalition is costly for the players. Thus, the objective of non-cohesive games is to generate a set of disjoint coalitions. This type of games is often referred to as *coalition formation game*.

Property 1. *The proposed game is a coalition formation game.*

As mentioned earlier, the objective is to form trusted multi-cloud coalitions in which the number of malicious members is minimal. Obviously, the probability of encountering malicious services increases as the size of the coalitions increases. In other words, the grand coalition entails grouping all the trustworthy and malicious services together into a single coalition. Therefore, the objective of this paper is to produce a set of disjoint coalitions instead of forming the grand coalition. Thus, the proposed game is a *coalition formation game*.

Coalitional games may be differentiated as well based on the utility that they assign to each coalition. Specifically, coalitional games may be either of Transferable Utility (TU) or Non-Transferable Utility (NTU). In TU games, the utility associated with each coalition of players worth the same for all the players who, as a result, can distribute and transfer this utility (e.g., money). In contrary, NTU games assume that the utility of the coalitions is non-distributable nor transferrable (e.g., happiness).

Property 2. *The proposed coalitional game is an NTU game.*

The utility of each service in a certain coalition is obtained by summing up the service's beliefs in trustworthiness in each of the coalition's members (Eq. (2)). Apparently, the belief in trustworthiness is a social relationship in which an agent assigns a probability about another agent's future behavior [24]. That is, trust cannot be neither distributed nor transferred among services. Therefore, the proposed game is an NTU game.

A hedonic game is a special case of NTU games in which players have preferences over the coalitions that they may join and the utility of any player in a certain coalition depends *exclusively* on the *identity* of the members in that coalition regardless of how other services are structured. In simple words, the term *hedonic* comes from the idea that players seek to enjoy each other's partnership, apart from numeric considerations. Therefore, we believe that hedonic games are the best type of coalitional games that can model the trust relationships among services. More specifically, a hedonic game is a subclass of coalitional games that satisfies the two following requirements [16]:

Condition 1. *A coalition formation game is hedonic if:*

1) *The utility of any player in a given coalition depends only on the members of that coalition.*
2) *The players have preferences over the set of possible coalitions and coalitions form based on these preference relationships.*

Property 3. *The proposed coalitional game is hedonic.*

In our game, the utility of the services in a certain coalition is obtained by summing up the service's beliefs in trustworthiness in each of the coalition's members (Eq. (2)). Thus, the utility of services in a given coalition is solely dependent on the members of that coalition, which satisfies the first condition. For the second condition, we will formally define in the rest of this section the preference function that enables services to build preference relations between coalitions and compare them during the coalition formation process. Before discussing the preference function, let's define first the concept of *preference relation* [16].

Definition 2 (**Preference Relation).** For every service $S_i \in \mathbb{N}$, a preference relation $(\geq_{S_i})_{S_i \in S \subseteq \mathbb{N}}$ is a complete, reflexive, and transitive binary relation over the set of all possible coalitions that S_i may join. $C_l \geq_{S_i} C_l'$ means that service S_i prefers to be a member of coalition C_l over being a member of C_l' or at least S_i prefers to be a member of both coalitions equally. $C_l >_{S_i} C_l'$ denotes that S_i strictly prefers to be part of C_l over being part of C_l'.

Based on the definition of preference relation, the preference function of the services can be defined as follows:

$$C_l \geq_{s_i} C_l' \Leftrightarrow P_{S_i}(C_l) \geq P_{S_i}(C_l'), \tag{11}$$

where $C_l \subseteq \mathbb{N}$ and $C_l' \subseteq \mathbb{N}$ are any two coalitions that service S_i is member of and $P_{S_i} : 2^{\mathbb{N}} \longmapsto \mathbb{R}$ is a preference function for any service S_i such that:

$$P_{S_i}(C) = \begin{cases} -\infty, & \text{if } a \in C \ \& \ belief_{S_i}^{a}(T) < belief_{S_i}^{a}(M) \\ 0, & \text{if } C \in h_{S_i}(t) \\ U_{S_i}(C), & \text{otherwise,} \end{cases} \tag{12}$$

where $h_{S_i}(t)$ represents the history set of service S_i at time t. The history set $h_{S_i}(t)$ contains the coalitions that service S_i has already joined and left at any time $t' < t$ before the formation of the current coalition structure $\Pi(t)$. The main intuition behind the preference function P_{S_i} defined in Eq. (12) is to allow each service S_i to choose the coalition that maximizes its belief in trustworthiness, while avoiding the coalitions that S_i believes contain malicious members. Particularly, the service S_i assigns a minimum preference (i.e., $-\infty$) to any coalition

Algorithm 2: Hedonic Coalition Formation Algorithm

1: **Input**: Initial partition of services $\Pi(t)$ at time t
2: **Output**: Final coalition structure $\Pi^*(t_f)$ at time t_f
3: **procedure** COALITIONFORMATION
4: Initialize $t=0$, $\Pi(t)=\{C_1(t),...,C_S(t)\}$, $h_{S_i}(t)=C_k^{S_i}(t)$
5: **repeat**
6: **repeat**
7: **for each** service $S_i \in \Pi(t)$ **do**
8: Select a coalition $C_l \in \Pi(t) \backslash C_k^{S_i}(t) \cup \{\emptyset\}$
9: **for each** service $S_j \in C_l$ **do**
10: **if** S_i has no previous interactions **then**
11: Request bootstrapping for S_j
12: **else**
13: Run Algorithm 1 to get $N(S_j)$
14: Compute $belief_{S_i}^{S_j}(T)$
15: **end if**
16: **end for**
17: Compute $belief_{S_i}^{C_l}(T)$ and $belief_{S_i}^{C_l}(M)$
18: Compare $P_{S_i}(C_l(t) \cup \{S_i\})$ and $P_{S_i}(C_k^{S_i}(t))$
19: **if** $C_l(t) \cup \{S_i\} >_{S_i} C_k^{S_i}(t)$
20: Leave C_k, i.e, $C_k(t) = C_k(t) \backslash \{S_i\}$
21: Join C_l, i.e., $C_l(t) = C_l(t) \cup \{S_i\}$
22: Update history, i.e., $h_{S_i}(t) = h_{S_i}(t) \cup \{C_l\}$
23: **else**
24: $\Pi(t+1) = \Pi(t)$
25: **end if**
26: **end for**
27: $t = t+1$
28: **until** no change in the partition happens.
29: **until** ε elapses
30: $\Pi^*(t_f)^* = \Pi(t)$
31: **return** $\Pi^*(t_f)$
32: **end procedure**

that contains a member that S_i believes is malicious (i.e., $a \in C$ & $belief_{S_i}^a(T) < belief_{S_i}^a(U)$). This condition is important to avoid being grouped with any malicious service in the same coalition. Moreover, the service avoids rejoining any previously visited coalition as long as its members do not change. This may be considered as a basic learning process and is important to reduce the complexity of the coalition formation process since the already visited coalitions are excluded from the choice set of the services [31]. Otherwise, the service prefers the coalition that maximizes its utility and that represents its belief in trustworthiness in the coalition's members (Eq. (2)). In this situation, the preference relation is determined by comparing the service's beliefs in trustworthiness for each pair of coalitions. In summary, services prefer the coalitions that contain no malicious members (based on their beliefs), those that have not been visited and left yet, and those that maximize the belief in trustworthiness.

5.2 Hedonic Coalition Formation Algorithm

To achieve the solution of the game, we propose in this section a distributed hedonic coalition formation algorithm that enables services to make decisions about which coalitions to join in such a way to minimize the number of malicious services in the final coalition structure. The algorithm is depicted in Algorithm 2.

Algorithm 2 works as follows. The algorithm takes as input an initial partition of services at a certain time t (line 1) and outputs the final coalition structure obtained after applying the trust-based hedonic coalition formation algorithm (line 2). First, the time t is initialized along with the initial partition of services at that time $\Pi(t)$ and the history set of each service S_i belonging to that partition (line 4). The algorithm repeats the following steps. Each service S_i in the initial partition selects a given coalition C_l (line 8). For each member of C_l, if the member is newly deployed and having no past interactions, S_i makes a request to bootstrap it (line 11); otherwise it runs the discovery algorithm described in Algorithm 1 to discover the member's direct neighbors (line 13). Thereafter, it computes the beliefs in trustworthiness and maliciousness for that member using Eq. (3) and Eq. (4) respectively (line 17). S_i uses then the preference function defined in Eq. (12) to determine the preference order between its current coalition and the selected coalition (line 18). If the utility of S_i in the new coalition $C_l(t) \cup \{S_i\}$ exceeds its utility in the current coalition $C_k^{S_i}(t)$ (line 19), then it leaves the current coalition (line 20), joins the new coalition (line 21), and updates its history set by adding the newly joined coalition to it (line 22). Otherwise, the partition of services remains unchanged (line 24). This process continues until converging to a Nash-stable coalition structure, i.e., the case where no service prefers to leave its current coalition and join another one (line 28). Note that the whole process is repeated periodically after a certain fixed period of time ε (line 29) to capture the changes that may occur in the partition; especially the dynamism in the services' trust values, arrival of new services, and leave of existing services.

For the computational complexity of Algorithm 2, the main complexity lies in the switch operations, i.e., the process of finding the next coalition to join (lines 7-18). The computational complexity of performing a switch operation is $O(\Pi)$, where Π is the coalition partition consisting of the disjoint coalitions of services. The worst case would be when each service acts alone in a singleton coalition as it implies that the number of coalitions in the coalition structure is exactly the number of services, i.e., $|\Pi| = |S|$.

5.3 Analysis of the Trust-based Hedonic Game

In this section, we analyze the properties of the proposed hedonic game. In particular, we analyze the convergence of the proposed coalition formation algorithm to a final solution and some stability concepts of the generated coalitions. Before starting the analysis, let's highlight some useful definitions and properties [16].

Definition 3 (**Nash Stability).** A partition Π is Nash-stable if no player in Π has incentive to leave its current coalition and move to any other coalition (possibly empty) in such a way that makes the coalition structure to change, assuming that the other coalitions remain unchanged, i.e., $\forall i \in N, C_k^{S_i} \geq_i C_l \cup \{i\}$ for all $C_l \in \Pi$.

In other words, a coalition structure Π is Nash-stable if (1) there exists no service S_i that prefers to leave its current coalition $C_k^{S_i} \in \Pi$ and act alone by forming its singleton coalition $\{S_i\}$, and (2) there exists no service S_i that has incentive leave its current coalition $C_k^{S_i} \in \Pi$ and join any other coalition $C_l \in \Pi$ in such a way that makes the coalition structure to change.

Definition 4 (**Individual Stability).** A partition Π is individually stable if no player in Π can benefit by moving from its current coalition to another coalition without making the members of the latter coalition worse off, i.e., $\nexists i \in N$ and $C_l \in \Pi \cup \{\emptyset\}$ such that $C_l \cup \{i\} >_i C_k^{S_i}$ and $C_l \cup \{i\} \geq_j C_l, \forall j \in C_l$.

In simple words, a coalitional structure Π is individually stable if there exists no coalition $C_l \in \Pi$ that a service S_i prefers over its current coalition $C_k^{S_i} \in \Pi$ without making its members worse off.

Property 4. *The number of coalition structures for N services is finite and given by D_N , where D_N represents the N^{th} Bell number and is computed as follows:*

$$D_N = \sum_{i=0}^{N-1} \binom{N-1}{i} \cdot D_i \quad for \quad N \geq 1 \quad and \quad D_0 = 1 \qquad (13)$$

Now, let's move to the analysis of the proposed coalition formation algorithm's properties; particularly the three common properties of hedonic games: convergence to a final coalition structure, Nash-stability, and individual stability. It's worth noting that the methodology followed in the analysis is inspired by that presented in [31].

Theorem 2. *Algorithm 2 converges to a final coalition structure* $\Pi^*(t_f)$ *consisting of a number of disjoint coalitions.*

Proof. The proof involves showing that the algorithm leads to distinct coalitions from time t to time $t+1$ and that the number of coalition structures is finite. Given any initial partition $\Pi(t)$ of services at time t, Algorithm 2 switches the partition at hand $\Pi(t)$ into another partition $\Pi(t+1)$ at time $t+1>t$ and so on until reaching the final partition $\Pi^*(t_f)$. Moreover, the preference function defined in Eq. (12) states that services will not revisit any coalition that has been already visited and left. Thus, any switch operation done in Algorithm 2 leads to a new partition that has not been visited yet. Given this property and the fact that the number of coalition structures is finite as per Property 4, we can conclude that the number of switch operations is finite and that the switch operation always leads to a final partition $\Pi^*(t_f)$. Hence, Algorithm 2 always converges to a final coalition structure comprising a number of disjoint services coalitions. □

Theorem 3. *Algorithm 2 converges to a Nash-stable coalition structure* $\Pi^*(t_f)$.

Proof. The theorem may be proved by contradiction. Assume that the final coalition structure $\Pi^*(t_f)$ is not Nash-stable. Then, there exists a service S_i that prefers to leave its current coalition $C_k^{S_i}(t_f)$ and join another coalition $C_l(t_f)$ at time t_f (i.e., $C_l(t_f)\cup\{S_i\} >_{S_i} C_k^{S_i}(t_f)$). Consequently, the coalition structure $\Pi^*(t_f)$ changes to a new coalition structure $\Pi^{**}(t_f)$ such that $\Pi^{**}(t_f)\neq\Pi^*(t_f)$, which contradicts with Theorem 2. Hence, we can conclude that Algorithm 2 always converges to a Nash-stable coalition structure $\Pi^*(t_f)$. □

Theorem 4. *Algorithm 2 converges to an individually stable coalition structure* $\Pi^*(t_f)$.

Proof. It has been proven that every Nash-stable coalition structure is also individually stable [16]. Since Algorithm 2 converges to a Nash-stable coalition structure as per Theorem 3, Algorithm 2 converges also to an individually stable coalition structure. □

6 Experimental Results and Analysis

In this section, we first explain the experimental setup used to perform our simulations and then study the performance of the trust-based hedonic coalition formation game by means of simulation experiments.

6.1 Experimental Setup

We implement our framework in a 64-bit Windows 7 environment on a machine equipped with an Intel Core i7-4790 CPU 3.60 GHz Processor and 16 GB RAM. Throughout simulations, we vary the percentage of malicious services from 0% to 50% and compare our model with three benchmarks: (1) Availability-based Coalition Formation [7], (2) QoS-based Coalition Formation [8], and (3) Hedonic Cloud Federations [2]. The Availability-based Coalition Formation considers the availability of the services as a building block in the community formation process. The QoS-based Coalition Formation considers, in addition to availability, several QoS metrics such as throughput and response time in the community formation process. The Hedonic Cloud Federations considers the prices and costs of the services (i.e., VMs) to formulate the utility function. The comparison is possible since all these approaches are based on a coalition formation algorithm. MATLAB has been used as a simulation tool to implement the different algorithms, where service instances have been modeled as objects; each of which having a set of QoS parameters. The QoS values such as promised and monitored availability are obtained from the CloudHarmony dataset [4], which contains information about services owned by well-known providers such as Amazon Web Services and Agile Cloud. The dataset comprises 53 different services operating in different parts of the world and 187 different activities for these services. The availability of the services has been studied during a period of a whole month and the average availability is recorded. During coalitions formation, the malicious services are considered as those that deviate from the SLA clauses. In particular, the initial decision on whether a certain service is trustworthy or not (i.e., before applying the proposed aggregation technique) is obtained by comparing the promised availability with the monitored availability. After coalitions are formed, the malicious services are considered those that refuse to share their needed resources with the coalition colleagues. To make our experiments fair with the Hedonic Cloud Federations model [2], which uses different parameters from ours (i.e., price and cost of VMs) and a small number of providers (i.e., eight providers), we have selected a subset of eight AmazonEC2 services (the same type of services used in [2]) from the used dataset, assigned them the same prices and costs used in [2] and that are publicly available[5], and run independent simulations from the ones used to compare with the other two models.

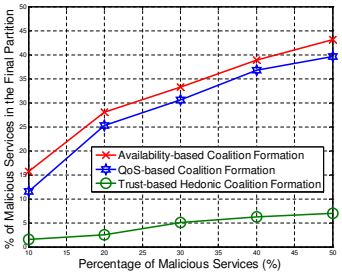
6.2 Experimental Results

First, we study in Fig. 3 the percentage of malicious services that exist in the final coalition structure w.r.t the percentage of malicious services that existed in the initial partition of services. In other words, the aim is to study how effective is each of the compared models in avoiding the malicious services during communities' formation. Fig. 3a shows that the percentage of malicious services in the final partition keeps increasing in the availability-based, QoS-based, and our trust-based hedonic coalition formation models with the increase in their percentage in the initial partition. However, the trust-based coalition formation model is more resilient to that increase and is able to reduce the percentage of malicious services up to 30% compared to the other models. The reason is that our model takes into account the trust relationships among services in the preference function (Eq. (11)) of the hedonic game used during the coalition formation process and is able as well to overcome the collusion attacks that may affect the trust establishment results as per Theorem 1. On the other hand, the percentage of malicious services in the other two models turns out to be high. The reason is that although these models take into account some QoS metrics in the community formation, the declared metrics may not be consistent with the actual metrics in case of passive malicious misbehavior. Moreover, Fig. 3b reveals that our trust-based model outperforms the hedonic cloud federations model in terms of minimizing the percentage of malicious members. It is worth noticing that the hedonic cloud federations model entails higher percentage of malicious members than both the availability-based and QoS-based models. The reason is that the former, contrary to the other two models, focuses solely on the prices and costs of services and disregards totally both the performance and security perspectives. Overall, we can conclude that if we allow services to rationally select their coalitions without considering their trust relationships, these services may have incentives to structure themselves into coalitions consisting of a large number of malicious services.

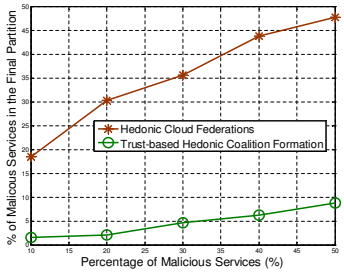
Next, we test the performance of the generated communities for a period spanning over more than 3 days (i.e., 260,000 iterations) and compute the average availability, response time, and throughput; where each single iteration represents a second. At each iteration, we assign 1000 requests for every community; meaning that each

4. http://cloudharmony.com/
5. http://aws.amazon.com/ec2/pricing/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2549019, IEEE Transactions on Services Computing

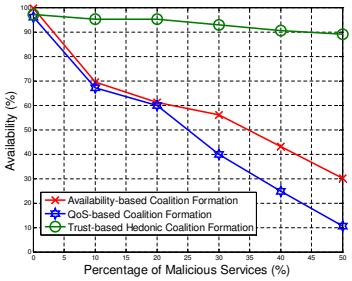


(a) Trust-based vs. Availability-based and QoS-based models

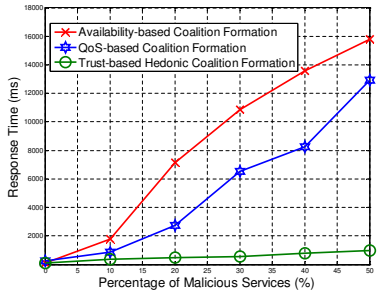


(b) Trust-based vs. Hedonic cloud federations models

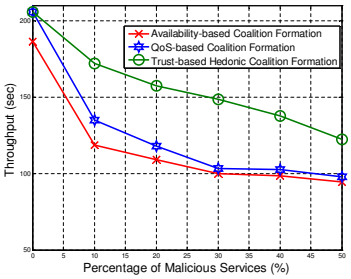
Fig. 3: Percentage of malicious services: Our trust-based model minimizes the number of malicious services



(a) Communities' Availability

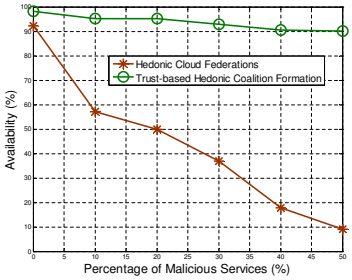


(b) Communities' response time

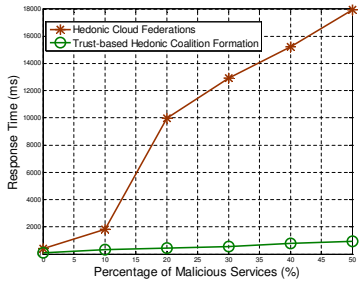


(c) Communities' throughput

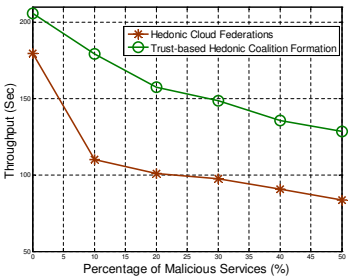
Fig. 4: Our model improves the availability, response time, and throughput compared to the Availability-based and QoS-based models



(a) Communities' availability



(b) Communities' response time



(c) Communities' throughput

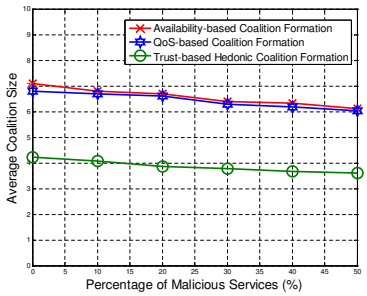
Fig. 5: Our model improves the availability, response time, and throughput compared to the Hedonic Cloud Federations model

community receives 1000 requests per second, which is realistic to a large degree. The malicious services at this stage are those that benefit from the resources of the other community colleagues but refuse to share their needed resources with them. Figs. 4 and 5 studies how effective are the formed coalitions in terms of availability, response time, and throughput. Availability depicts the time period in which a community of services is ready for use and is obtained by dividing the number of performed requests by the total number of received requests. Fig. 4a shows that in the absence of malicious services in the initial partition, the availability-based coalition formation model outperforms the other two model by achieving an availability percentage of $\approx 100\%$. This result is expected since this model takes the availability as a sole factor for forming communities. However, starting from 10% of malicious services, our model outperforms both the availability-based and QoS-based models whose performance begins to decrease drastically. This is due to the fact that our trust-based model minimizes the percentage of malicious services in the final partition as per Fig. 3a. Practically, the increase in the number of malicious services that refrain from sharing their resources when these resources are needed leads to an increase in the number of unfulfilled requests. Similarly, Fig. 5a reveals that our model outperforms the hedonic cloud federations model in terms of availability for the same above-discussed arguments.

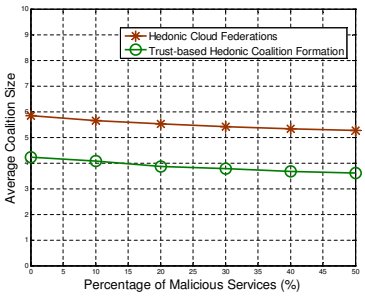
Figs. 4b and 5b studies how effective are the formed coalitions in terms of response time. Response time represents the time between the submission of the request and the receipt of the response, which incudes both service time and wait time. Fig. 4b reveals that our trust-based model yields much less response time compared to the availability-based and QoS-based models in the presence of malicious services. Similarly, Fig. 5b shows that our model outperforms the hedonic cloud federations model in terms of response time. This is also due to the fact that our trust-based model minimizes the percentage of malicious services in the final partition as per Fig. 3. Practically, the increase in the number of malicious services that refuse to share their needed resources entails additional wait time to find alternative non-malicious services and offering the same type of resources, which augments consequently the whole response time.

Figs. 4c and 5c study the performance of the produced coalitions in terms of throughput. Throughput describes the number of requests that coalitions can handle in a given time. For our simulations, we measure the throughput per second. Fig. 4c reveals that our trust-based model yields much higher throughput compared to the other two models in the presence of malicious services. Similarly, Fig. 5c shows that our model outperforms the hedonic cloud federations model in terms of throughput. Obviously, the improvement in terms of throughput is a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2549019, IEEE Transactions on Services Computing



(a) Trust-based vs. Availability-based and QoS-based models



(b) Trust-based vs. Hedonic cloud federations models

Fig. 6: Average Coalition Size: Our trust-based model achieves coalitions of less size

natural result of the improvement in terms of response time.

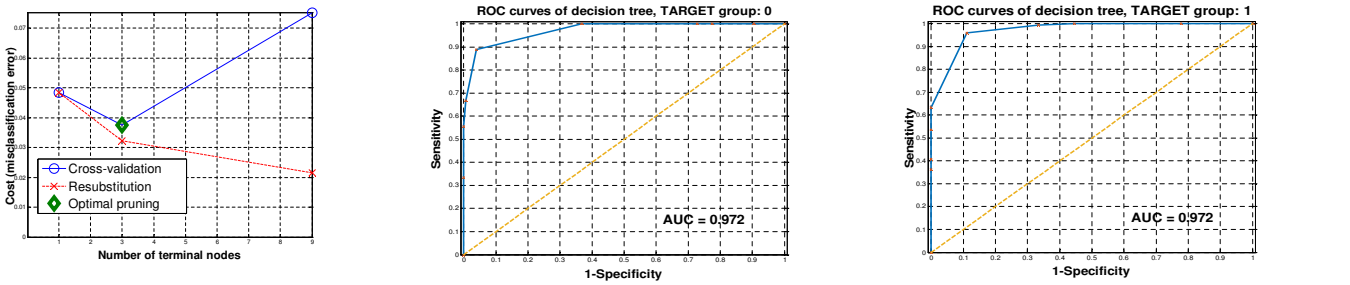
Fig. 6 measures the average coalitions size w.r.t. the increase in the percentage of malicious services in the initial partition. Fig. 6a reveals that our trust-based model generates coalitions of smaller size than those generated by the availability-based and QoS-based models. Moreover, Fig. 6b shows that the size of the coalitions produced by our trust-based model is smaller than that of the coalitions produced by the hedonic cloud federations model. The intuition behind this result is that coalitions of smaller sizes are able to reduce the number of malicious services. In other words, the size difference between our model and the other models may be thought of as the malicious services that our model excludes from the coalition structure. Thus, we can conclude that our model produces a network of large number of small disjoint coalitions.

Finally, we study the effectiveness of the proposed trust bootstrapping mechanism in providing accurate initial trust values. To this end, we train a decision tree classifier on our dataset using the 10-fold cross-validation model. The dataset consists of four attributes: *Service_Provider*, *Operation_Country*, *Promised_Availability*, and *Status*. The first attribute denotes the service provider's name, *Operation_Country* denotes the country in which the service was used, *Promised_Availability* denotes the availability promised for this service in the SLA contract, and *Status* denotes the status of the service either trusted or not. The *relative importance* of the promised availability attribute is reported to be 0% since usually all providers tend to promise optimal availability (i.e., 100%), compared to 100% for the service provider name's attribute and 64.1% for the deployment country attribute. The classifications resulted initially in a tree consisting of 9 leaves. In order to guarantee the accuracy of the classifier and minimize the overfitting, we measure, plot, and compare in Fig. 7a the cross-validation and resubstitution errors on several subsets of the original tree in order to find the optimal pruning level that best classifies the new data (other than the training set). For a decision tree consisting of 9 leaves (i.e., the initially obtained tree), Fig. 7a shows that the resubstitution error is significantly less than the cross-validation error, which indicates that the tree in its current form overfits the training dataset (i.e., although the tree classifies the training dataset well, its structure is sensitive to this specific training set in such a way that its performance would decrease when used on new data). The Fig. reveals as well that the resubstitution error is excessively optimistic in the sense that this rate is continuously decreasing as the size of the tree increases. On the other hand, the cross-validation error rate decreases initially with the increase in the tree size but begins to increase beyond a certain point. Therefore, the optimal tree size that best performs on the new data is that with with the smallest cross-validation error, which is 3 in our case (Fig. 7a).

Fig. 7b and Fig. 7c represent the Receiver Operating Characteristic (ROC) curves [32] generated by the classifier, where sensitivity measures the proportion of positives that are correctly classified as such (a.k.a true positive rate) and specificity measures the proportion of negatives that are correctly classified as such (a.k.a true negative rate). Thus, 1− specificity means the proportion of positives that are misclassified as negatives (i.e., false positive rate). Fig. 7b measures the accuracy of the bootstrapping mechanism in classifying the malicious services as such. Thus, sensitivity means in this figure the percentage of malicious services that are correctly classified as malicious and 1−specificity means the percentage of malicious services that are correctly misclassified as trustworthy. On the contrary, Fig. 7c measures the accuracy of the bootstrapping mechanism in classifying the trustworthy services as such. Thus, sensitivity means in this figure the percentage of trustworthy services correctly classified as trustworthy, whereas 1−specificity means the percentage of trustworthy services that are misclassified as malicious. The best possible classification model would yield 100% sensitivity (no false negatives) and 100% specificity (no false positives); thus a point whose coordinates are (0, 1). The dashed diagonal line represents a fully random guess (Fig. 7). By carefully inspecting both Fig. 7b and Fig. 7c, we can notice that the sensitivity and specificity measures are nearly optimal in our bootstrapping mechanism. The overall classifier's accuracy is quantified in terms of Area Under the Curve (AUC) [32], where a value of 1 represents a perfect test and a value of 0.5 represents a worthless test. Figs. 7b and 7c reveal that our bootstrapping mechanism yields high AUC values up to 0.972 in both Figs.

7 Conclusion

Services' communities provide an effective solution to the discovery, composition, and resources scaling problems. The existing community formation models can support only a single cloud model but are ineffective when services are distributed across several data centers since they rely on centralized architecture and overlook the problem of encountering malicious services. In this paper, we proposed a comprehensive trust framework that allows services to establish credible trust relationships in the presence of collusion attacks in which attackers collude to mislead the trust results. We discussed as well a trust bootstrapping mechanism that capitalizes on the concept of endorsement in online social networks to assign initial trust values for the newly deployed services. Thereafter, we designed a trust-based hedonic coalitional game that is able to form trusted multi-cloud services' communities and proposed a relevant algorithm that converges to a stable coalition structure. Simulation experiments conducted on a real cloud dataset revealed that our proposed game minimizes the number of malicious services in the final coalition structure up to 30% compared to three state-of-the-art cloud federations and service communities formation models. Moreover, our game improves the performance of the formed communities in terms of availability, response time, and throughput. Besides, the proposed bootstrapping mechanism yields high accuracy levels up to 97.2%.



(a) Optimal pruning level

(b) Accuracy in classifying malicious services

(c) Accuracy in classifying trustworthy services

Fig. 7: Bootstrapping accuracy: Our bootstrapping mechanism achieves high accuracy rate

8 Acknowledgments

This work was supported by the Fonds de Recherche du Québec - Nature et Technologie (FRQNT), Natural Sciences and Engineering Research Council of Canada (NSERC), Khalifa University of Science, Technology & Research (KUSTAR), Associated Research Unit of the National Council for Scientific Research (CNRS-Lebanon), and Lebanese American University (LAU).

References

[1] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud Computing: A Perspective Study," *New Generation Computing*, vol. 28, no. 2, pp. 137–146, 2010.

[2] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Cloud federations in the sky: Formation game and mechanism," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 14–27, 2015.

[3] Z. Maamar, S. Subramanian, J. Bentahar, P. Thiran, and D. Bensilamane, "An approach to engineer communities of web services: Concepts, architecture, operation, and deployment," *International Journal of E-Business Research (IJEBR)*, vol. 5, no. 4, pp. 1–21, 2009.

[4] O. Abdel Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Misbehavior Detection Framework for Community-based Cloud Computing," in *Proceedings of the 3rd International Conference on Future Internet of Things and Cloud*. IEEE, 2015.

[5] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "A stackelberg game for distributed formation of business-driven services communities," *Expert Systems with Applications*, vol. 45, pp. 359–372, 2016.

[6] B. Medjahed and A. Bouguettaya, "A dynamic foundational architecture for semantic web services," *Distributed and Parallel Databases*, vol. 17, pp. 179–206, 2005.

[7] A. Liu, Q. Li, L. Huang, S. Ying, and M. Xiao, "Coalitional game for community-based autonomous web services cooperation," *IEEE Transactions on Services Computing*, vol. 99, no. 3, pp. 387–399, 2012.

[8] E. Khosrowshahi-Asl, J. Bentahar, H. Otrok, and R. Mizouni, "Efficient Community Formation for Web Services," *IEEE Transactions on Services Computing*, vol. in press, 2014.

[9] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud federation," in *Proceedings of the 2nd International Conference on Cloud Computing, GRIDs, and Virtualization*, 2011.

[10] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4–1, 2009.

[11] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "A View of Cloud Computing," *Communications of the ACM*, vol. 53.

[12] T. Noor, Q. Sheng, L. Yao, S. Dustdar, and A. Ngu, "CloudArmor: Supporting Reputation-based Trust Management for Cloud Services," *IEEE Transactions on Parallel and Distributed Systems*, 2015.

[13] A. F. Barsoum and A. Hasan, "Enabling dynamic data and indirect mutual trust for cloud computing storage systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 12, pp. 2375–2385, 2013.

[14] R. Shaikh and M. Sasikumar, "Trust model for measuring security strength of cloud computing service," *Procedia Computer Science*, vol. 45, pp. 380–389, 2015.

[15] M. Guazzone, C. Anglano, and M. Sereno, "A game-theoretic approach to coalition formation in green cloud federations," in *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, 2014, pp. 618–625.

[16] A. Bogomolnaia and M. O. Jackson, "The stability of hedonic coalition structures," *Games and Economic Behavior*, vol. 38.

[17] T. M. Chen and V. Venkataramanan, "Dempster-shafer theory for intrusion detection in ad hoc networks," *Internet Computing, IEEE*, vol. 9, no. 6, pp. 35–41, 2005.

[18] B. Benatallah, Q. Sheng, and M. Dumas, "The self-serv environment for web services composition," *IEEE Internet Computing*, vol. 7, no. 1, pp. 40–48, 2003.

[19] I. Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 123–130.

[20] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 228–235.

[21] D. Niyato, A. V. Vasilakos, and Z. Kun, "Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach," in *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2011, pp. 215–224.

[22] Z. Malik and A. Bouguettaya, "Reputation bootstrapping for trust establishment among web services," *IEEE Internet Computing*, vol. 13, no. 1, pp. 40–47, 2009.

[23] R. Zhou and E. A. Hansen, "Breadth-first heuristic search," *Artificial Intelligence*, vol. 170, no. 4, pp. 385–408, 2006.

[24] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "A survey on trust and reputation models for Web services: Single, composite, and communities," *Decision Support Systems*, vol. 74, pp. 121–134, 2015.

[25] O. A. Wahab, H. Otrok, and A. Mourad, "A cooperative watchdog model based on Dempster-Shafer for detecting misbehaving vehicles," *Computer Communications*, vol. 41, pp. 43–54, 2014.

[26] B. Yu and M. P. Singh, "An evidential model of distributed reputation management," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. ACM, 2002, pp. 294–301.

[27] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[28] H. Yahyaoui, "A trust-based game theoretical model for web services collaboration," *Knowledge-Based Systems*, vol. 27, pp. 162–169, 2012.

[29] M. A. Friedl and C. E. Brodley, "Decision tree classification of land cover from remotely sensed data," *Remote Sensing of Environment*, vol. 61, no. 3, pp. 399–409, 1997.

[30] M. Osborne, *An Introduction to Game Theory*, 1st ed. Oxford University Press, 2003.

[31] W. Saad, Z. Han, T. Başar, M. Debbah, and A. Hjorungnes, "Hedonic coalition formation for distributed task allocation among wireless agents," *IEEE Transactions on Mobile Computing*, vol. 10, no. 9, pp. 1327–1344, 2011.

[32] T. Fawcett, "An introduction to ROC analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.