Expert Systems With Applications 45 (2016) 359–372

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A Stackelberg game for distributed formation of business-driven services communities

Omar Abdel Wahab [a], Jamal Bentahar [a,*], Hadi Otrok [a,b], Azzam Mourad [c]

[a] Concordia Institute for Information Systems Engineering, Concordia University, Montreal Quebec, H3G 2W1,Canada
[b] Department of Electrical & Computer Engineering, Khalifa University of Science, Technology & Research, Abu Dhabi, UAE
[c] Department of Computer science and Mathematics, Lebanese American University, Beirut, Lebanon

ARTICLE INFO

Keywords:
Service federation
Stackelberg game theory
Distributed community formation
Cloud computing
Business ecosystem

ABSTRACT

With the advent of cloud computing and the rapid increase in the number of deployed services, the competition between functionally-similar Web services is increasingly governing the markets of services. For example, Amazon and Google are in an intense competition to dominate the market of cloud-based Web services. Such a highly competitive environment motivates and sometimes obliges services to abandon their pure competitive strategies and adopt a cooperative behavior in order to increase their business opportunities and survive in the market. Several approaches have been advanced in the literature to model the cooperation among Web services in a community-based environment. However, the existing approaches suffer from two main drawbacks that limit their effectiveness in the real-world services market. First, they rely on a centralized architecture wherein a *master* entity is responsible for regulating the community formation process, which creates a single point of failure. Second, they ignore the business potential of the services and treat all of those services in the same way, which demotivates the participation of the well-positioned ones in such communities. To tackle these problems, we distinguish in this paper between two types of services: *leaders* and *followers*. Leaders are those services that enjoy high reputation, market share, and capacity of handling requests; whereas followers are those services that cannot compete against the leaders. Thereafter, we model the community formation problem as a virtual trading market between these two types of services and propose a distributed Stackelberg game for this purpose. Promisingly, the proposed model gives guidance to a cooperative model that can be applied in the real markets of Web services in order to achieve higher performance, efficient services compositions, and better resources utilization. The performance of the model is analyzed using a real-life flight booking dataset that includes 2507 services operating on the Web. Simulation results show that the proposed model is able to increase the satisfaction of Web services in terms of gained payoff and reputation and the satisfaction of users in terms of quality of service provided to their requests compared to the existing models, namely the one-stage game theoretical model and a heuristic model.

1. Introduction

The industry push is increasingly shifting toward the execution of distributed business processes that spread over multiple applications. This requires assuring high-levels of interoperability and wider flexibility in managing business processes (Agostino, Michele, & Paola, 2007). The trend is to adopt the service-oriented architecture (SOA) as a strategic paradigm for achieving and automating the business processes, thanks to the wide set of benefits it provides such as: loosely coupling, reusability, location transparency, parallel development, and better scalability (Eric & Greg, 2004; Li, Zhang, & O'Brien, 2011). SOA is constructed by means of a collection of services that communicate with each other. This demands software components to be able to understand the objectives of the process, deduct the steps needed to attain these objectives, and select the adequate services to communicate with Barros et al. (2011). These requirements emphasize the need for a certain degree of autonomy enabling Web services to adapt to the circumstances that they may face during the run-time such as the need to cooperate with other services or the ability to manage a composition request. In this scope, autonomy has been widely investigated as a building block property allowing Web services to cope with the surprising challenges that may arise due to the dynamic, heterogeneous, and complex environment within which they operate (Alferez & Pelechano, 2011; Hamadi & Benatallah, 2003; Paolucci & Sycara, 2003). Agent-based Web services

* Corresponding author. Tel.: +1 514 848 2424; fax: +1 514 848 3171.
E-mail addresses: o_abul@encs.concordia.ca (O.A. Wahab), bentahar@ciise.concordia.ca (J. Bentahar), Hadi.Otrok@kustar.ac.ae (H. Otrok), azzam.mourad@lau.edu.lb (A. Mourad).

http://dx.doi.org/10.1016/j.eswa.2015.09.047
0957-4174/

(García-Sánchez, Valencia-García, Martínez-Béjar, & Fernández-Breis, 2009; Gibbins, Harris, & Shadbolt, 2004; Maamar, Mostefaoui, & Yahyaoui, 2005; Maximilien & Singh, 2003) is a paradigm that implements this vision by allowing Web services to make appropriate decisions in order to adapt to the real-time challenges. Such an agent-based model has shown to be successful in a wide range of application domains including Grid and Cloud resource management (Kang & Sim, 2012), Intelligent Transportation Systems (Wahab, Otrok, & Mourad, 2013, 2014a, 2014b), and humans daily activities management (e.g., personalized home-care treatments) (Isern et al., 2011). In this context, two decision making levels have to be distinguished: (1) the strategic level; and (2) the operational level. The strategic level is about decisions related to the cooperation agreements between providers (companies). The operational level, which is the focus of the paper, deals with real-time decisions to participate in a particular cooperation request based on the current situation of the involved services. Thus, two Web services cannot decide to cooperate unless an agreement between their providers does a priori exist. On the other hand, the existence of such an agreement does not entail that the Web services will be always cooperating. This decision is mainly driven by the real-time parameters of the services.

Motivations. Communities of Web services have been proposed as virtual clusters grouping autonomous Web services offering similar functionalities to provide a joint environment of collaboration and interoperability among different applications (Khosravifar, Bentahar, Moazin, & Thiran, 2010; Maamar, Subramanian, Bentahar, Thiran, & Bensilamane, 2009; Yahyaoui, Maamar, Lim, & Thiran, 2013). The high-level objective of the community is optimizing the responsiveness, quality, and availability of the services as well as facilitating their discovery. The individual objective of each service in the community is to increase its share of assigned requests and augment the chances of participating in composition sequences.

From the business perspective, the concept of community is similar to that of *Business Ecosystem* (Gueguen, 2006; Muegge, 2013) wherein entities (e.g., companies, lobbies, associations, and so on) are motivated and sometimes obliged to form a joint business ecosystem to be able to survive and compete in the market. In this work, we envisage communities of Web services from the business perspective by considering each community as a business ecosystem gathering Web services offering a common functionality and seeking to improve their position in the market. Torrès-Blay (2009) defines a business ecosystem as "*a heterogeneous coalitions of organisations from different industries forming a strategic community of interests or values, structured as a network, around a leader that manages to impose or share its business vision or its technological standard*". In this paper, we adopt a similar definition for Web services community except that community members are not necessarily coming from "different industries", which reflects the fact that Web services within communities share similar functionalities. Based on Torrès-Blay's definition and as is the case in the real-world business markets, two types of Web services can be distinguished: *leaders* and *followers*. Leaders refer to as those services that are well-positioned in the market thanks to their strong resources and parameters.[1] As real-life examples, Google, Amazon, and eBay may represent such leaders. On the other hand, followers are those that have less resources and parameters compared to the leaders. In the rest of this paper, the terms "*leader (Web) services*" and "*follower (Web) services*" will be used to refer to these two kinds of Web services. Additionally, the terms "confederate" and "confederation", that are inspired by the real-world business context, will be used to describe the act when Web services join each other to form up a joint community.

Several approaches (Khosravifar, Alishahi, Bentahar, & Thiran, 2011; Khosravifar et al., 2013; Khosrowshahi-Asl, Bentahar, Otrok, & Mizouni, 2015; Lim, Thiran, Maamar, & Bentahar, 2012; Liu, Li, Huang, Ying, & Xiao, 2012; Maamar et al., 2009) have been proposed to tackle the problem of building Web services communities. The common trend in these approaches is the use of a Service Level Agreement (SLA; Bianco, Lewis, & Merson, 2008) contract to regulate the community formation process, where a certain Web service is statically designated as a master (or manager) for the community to control, among other issues, the membership of the services (i.e., accepting/refusing and inviting/firing Web services in the community). Different from these approaches, our proposal is a fully distributed formation model, where all Web services are autonomous in making their decisions and there is no need for a central static party to control the membership issue. Such a distributed approach complies better with the autonomous nature of the Web services. Moreover, the existing approaches do not consider the business perspective of the community formation process. Practically, in the real markets of services, some Web services may consider themselves strong enough to act alone and prefer thus not to give the other (less strong) Web services the advantage of being structured within the same community. Such services should receive some privileges in order to stimulate their contribution in the community formation process. To tackle this issue, we use in this paper a Stackelberg game model that gives the leader services, that are well-positioned in the market, the advantage of acting first by selecting the set of followers that they are willing to cooperate with and making the appropriate offer based on their own business needs and preferences.

Contributions. The main contribution of this paper is a distributed services community formation model that takes into account the business perspective of the services. To this end, we formulate the community formation problem as a virtual trading market that works as follows. First, a given leader pre-selects, at a time, a set of follower Web services to confederate with based on their advertised parameters (e.g., the top ten Web services in the domain of hotel reservation) and specifies a limited *quota* based on which the final selections will occur. The size of the quota is determined according to the need of the leader (i.e., based on its business requirements and objectives). The leader publishes its offer consisting of its own reputation, market share, and capacity to the pre-selected set. Under the pre-determined parameters, followers compete with each other in a non-cooperative game model to decide about the payment (community membership fee) to be made for the leader in response to his offer. This payment can be represented as credit, token, or other equivalents. The objective of the followers is to convince the leader to consider them when selecting its final quota, while minimizing the community membership fee. Based on the results of the followers' non-cooperative game, the leader adjusts its strategy by selecting the optimal quota of followers that maximizes its utility. In this scenario, all leader and follower Web services are rational; thus seeking foremost to maximize their own utilities. Therefore, the considered problem can be analyzed by means of game theory. Moreover, the model is characterized by a hierarchical structure, where the leader (first player) optimizes its strategy while being aware of the impacts of its decision on the behavior of the followers (second player). This naturally formulates a two-stage game, where leader services play first and follower services play second after observing leaders' strategies.

The main contributions of the paper are highlighted in the following:

1. Proposing a distributed formation model for services communities, where all services are totally autonomous in making their decisions. To the best of our knowledge, this is the first work that considers a fully distributed environment for Web services communities' formation. Existing approaches suppose the existence of the community and community manager (or

[1] Throughout the paper, the term parameters is used to represent the reputation, market share, and capacity of handling requests of the Web service.

master) statically as a result of an SLA and envisage the problem from the perspective of the Web services that are willing to join the preexisting communities. By proposing a fully distributed model, we are decentralizing the load placed on a single party (i.e., the master) and thus reducing the detrimental impacts caused by failures or malicious behaviors on a single point.
2. Formulating the problem as a two-stage sequential Stackelberg game model and deriving the equilibrium point analytically. By using a two-stage game, Web services are able to make more strategic and optimal decisions in the long-term as the game is played sequentially, which increases the learning space of the players.
3. Differentiating between Web services based on to their parameters. Such a taxonomy is important in the real-world markets since it links the advantages assigned to Web services to their actual position in the market; thus motivating these services to contribute in the community formation process.

The proposed model is validated through simulation experiments conducted on a flight booking dataset (Khosrowshahi-Asl et al., 2015) that includes 2507 real services operating on the Web and records the values of 9 QoS metrics populated from a real-life Web service dataset (Al-Masri & Mahmoud, 2009). Experimental results reveal that our model is able to increase the satisfaction of the Web services in terms of achieved payoff and reputation and the satisfaction of users in terms of QoS provided to their requests compared to a one-stage game theoretical model and a heuristic model used as a benchmark.

Organization. The rest of the paper is organized as follows. Section 2 discusses relevant related work and highlights the unique features of our model. In Section 3, the components of our model are introduced and explained. In Section 4, the game theoretical solution model is presented along with the associated mathematical derivations. In Section 5, we illustrate why and how can our model be useful in the real-world markets of Web services by explaining a complete scenario. Section 6 introduces our simulation settings and results to confirm the theoretical proofs. In Section 7, we highlight the threats of validity that encounter our model and explain how we react to mitigate them. Finally, Section 8 concludes the paper and identifies future work.

2. Related work

This section gives an overview on the approaches that tackled the concept of *communities* in the service-oriented paradigm and highlights the main application domains of the *Satckelberg* game theoretical model, since our work combines these two key concepts.

2.1. Communities of Web services

In Benatallah, Sheng, and Dumas (2003), the authors proposed the concept of Service Containers as a community gathering substitutable Web services that share a common functionality. These Containers are no more than Web services created, advertised, discovered, and invoked just as elementary and composite Web services are. The purpose is to facilitate the composition process when the set of Web services is large and dynamic. This is done by allowing the invoke of the service container operations instead of invoking elementary or composite service operations.

Maamar et al. (2009) defined a similar concept for the communities of Web services. They considered the community as a group of Web services sharing the same functionality but differing in their quality of service (QoS) parameters. This approach differentiates between the *master* Web service that is charged of managing the operations of the community and the *slave* Web services that are simply all the other community members. The responsibilities of the master include attracting new Web services, retaining well-performing services, firing bad-performing services, and selecting the services that will be part of the upcoming compositions.

Benslimane et al. (2007), the authors proposed a multi-layer approach for Web services composition made up of three constituents: *component*, *community*, and *composite*. The component layer comprises the Web services themselves, the composite layer illuminates the requirement of composing several Web services, while the community layer resides between the component and composite layers and has the role of organizing the Web services having common functionalities. The community is composed of abstract Web services describing the common functionality of the community and concrete Web services implementing this functionality. The composite layer is fed by the community layer with the needed components.

In Limam and Akaichi (2010), the authors defined the community of Web services as a centralized infrastructure accessed across distributed Web services and gathering services having the same functionality with the aim of improving the availability and enhancing the collaboration. This main focus of this approach is to target the issues related to community management, queries resolution among communities, and queries caching. To this end, the authors suggested the use of semantic cache whose basic idea is to store and reuse previously processed results in order to improve the performance while answering future queries.

In Medjahed and Bouguettaya (2005), the authors proposed a framework for ontological organization of Web services using communities. The community serves as a cluster grouping the Web services based on their domain of interest. The community is created by the service providers who identify the community of interest and register their services in it. Communities provide a set of generic functions that may be customized by the underlying services. In a close work (Zeng, Benatallah, Dumas, Kalagnanam, & Sheng, 2003), addressed the problem of enhancing the runtime of the process of selecting the Web services to participate in large compositions. As the number of Web services offering the same functionality tends to be very high, they proposed grouping these services into communities that provide descriptors to a certain functionality. Once the community receives a request, it selects one of its current members to fulfill that request. The selection is based on a set of criteria such as characteristics of the members, parameters of the request, status of ongoing executions, and history of past executions.

All of the aforementioned approaches focus either on user satisfaction and/or on performance optimization; thus ignoring Web services' satisfaction. Nonetheless, services are becoming more and more intelligent as a result of the technological advancements made in the domain of agent-based software engineering. Therefore, such services seek foremost to maximize their own utility, which discourages them from participating in the community formation process unless they receive some incentives.

Recently, several approaches have been proposed to study and analyze the objectives of the Web services as rational agents in the process of creating communities. In this context, Khosravifar et al. proposed initially in Khosravifar et al. (2011) and then comprehensively in Khosravifar et al. (2013) a decision making framework for both Web services and community master in order to help them adopt strategies resulting in higher utilities. The strategy set available for the Web services is: (1) to attempt to join the community, and (2) to accept the invitation of joining. The strategy set available for the community master is: (1) to accept the request of joining from Web services, (2) to refuse the request of joining, and (3) to send joining invitations. A one-stage non-cooperative game model is developed between these two players (master and Web services) and two thresholds are derived, one for Web services and the other for the master. Web services compare their expected performance (after joining) with the corresponding threshold and decide whether to join

or not, and whether to accept the joining invitation or not. Similarly, the master uses the derived threshold to decide whether to keep or fire the Web services from the community.

In Lim et al. (2012), the authors proposed a three-way satisfaction approach to help the master of the community select the Web services that will participate in fulfilling users' requests in an efficient manner. They considered the satisfaction of the three parties involved in this scenario; namely, Web services, users, and master. To this end, they formulated a satisfaction function for each party. For the user, the satisfaction is related mainly to the QoS provided to its requests; particularly, the availability, successability, and response time. For Web services, the satisfaction is expressed as the participation level of that Web service in the community's activity. The satisfaction of the master is expressed in terms of the revenue it earns. Then, a score function is formulated as a weighted sum of these satisfaction functions. The master uses this score function to quantitatively compare one selection against another during the selection process.

Liu et al. (2012) have introduced a coalitional game-theoretical model to create a cooperative scheme among autonomous Web services in a community-based context. Their approach allows Web services to reach individually stable coalition partition, where each Web service can maximize its utility through cooperation without reducing other Web services' utilities. The task-distribution process in the community is handled through a coordination chain, where services are invoked or replaced based on their availabilities. In a close work, a coalitional cooperative game model has been introduced in Khosrowshahi-Asl et al. (2015) with the aim of finding efficient ways of forming coalitions of Web service within communities. The key idea is to guarantee the fairness while distributing the gain among coalition members in order to maintain the stability of the coalitions. A coalition is deemed stable if no subset of its Web services can find significant gain by deviating from that coalition. Upon receiving new membership request, the community coordinator first verifies whether the new coalition taking into consideration that new member will remain stable. If so, the request is accepted; otherwise, the coordinator rejects the membership request.

Different from the above approaches, we propose a fully distributed community formation approach, where all the involved parties are autonomous in their behaviors. More specifically, the existing approaches assume the pre-existence of a community and community manager/master as a result of an SLA contract and investigate the problem in terms of services joining pre-defined communities. To the best of our knowledge, this is the first approach that considers a distributed community formation process "from scratch". Unlike the existing approaches that consider the community manager/master as a pre-defined, static, and abstract agent, the community leader in our approach is an active service that is dynamically designated based on its parameters. Moreover, the existing approaches ignore the business perspective of the services in the community formation process. In this work, we consider the business context by defining two types of Web services (i.e., leaders and followers) and employing a Stackelberg game theoretical model that gives the leader services, that are well-positioned in the market (i.e., having high reputation, market share, and capacity of handling requests), the advantage of acting first by selecting the set of followers that they are willing to cooperate with and making the appropriate offer based on their own business needs and preferences. Such a taxonomy is important in the real-world markets since it links the advantages assigned to Web services to their actual position in the market; thus motivating the participation of these services in the community formation process. In addition, our approach is the first that considers a two-stage sequential game theoretical model in the context of Web services communities formation. By using a two-stage model, we are enhancing the learning space of the services and enabling them to make strategic and long-term decisions. In fact, the players in our game decide about their current actions after considering the future consequences of these actions, and the effect of these actions on the behavior of the other players. This has the advantage of creating optimal, stable, and long-living communities.

2.2. Stackelberg Game theoretic approaches

Stackelberg game model has been widely used to model the situations that are characterized by a hierarchical structure. It was developed in 1934 by Heinrich von Stackelberg in his book "Market Structure and Equilibrium" (von Stackelberg, 1934)[2] to model the imperfect competition in the market study . It represented a turning point in the study of market structure; particularly the analysis of duopolies (i.e., the cases when two firms have full control over the market). The basic idea of Stackelberg is that one player (leader), thanks to its historical precedence, size, reputation, innovation, information, and so forth, has the right to make the first move. Then, the other player (denoted as follower), that is less strong, observes the leader's strategy and decides about its own accordingly. Stackelberg enjoys several characteristics that make it appealing to model the hierarchical situations such as: (1) it is a sequential game (non-simultaneous), where one player plays first and then plays the other one; (2) the leader knows ex ante that the follower observes his action; (3) the leader's action is irreversible; and (4) it is characterized by the *first mover's advantage* property, which states that the player who plays first yields a payoff that is higher than that of the second player. In the following, we highlight some application domains wherein Stackelberg game has shown to be successful.

In Zhang and Zhang (2009), the authors used a Stackelberg game to model the spectrum allocation problem in Cognitive Radio Networks. The idea is to allow primary (licensed) network users to use secondary (unlicensed) users as cooperative relays to improve the performance of primary transmission. This situation has been modeled as a two-stage Stackelberg game where primary users, denoted as leaders, decide about the slot of bandwidth to be used for direct transmission as well as the set of secondary users they are willing to cooperate with. The secondary users, acting as followers, decide about the payment to make under the pre-decided bandwidth slot with the target of maximizing the transmission rate without having to make large payments. There are some aspects of similarities between this approach and the approach considered in this paper in the sense that leaders decide about the set of followers that they are willing to cooperate with and followers have to make a payment in response. However, different context, model, and scenario are considered in our approach. In fact, the selection of followers in our approach occurs in two phases, where leaders pre-selects some followers for possible cooperation in the first phase and optimize by considering only a limited final quota of these pre-selected followers in the second phase. Moreover, we use different parameters and mathematical formulations and target a totally different scenario (i.e., Web services).

A Stackelberg game-based approach has been used also to model the problem of efficient bandwidth allocation in the cloud-based wireless networks (Nan et al., 2014), where desktop users watching the same live program may be willing to share their live-streaming with the nearby mobile users. This situation is modeled as a Stackelberg game that involves two-stages: (1) a non-cooperative game among desktop users to decide the bandwidth size to be shared with the mobile users along with the price of sharing, and (2) an evolutionary game among mobile users to decide the desktop users to connect with under the offered size and price.

[2] This book has been translated to English in von Stackelberg (2011) by Damien Bazin et al.

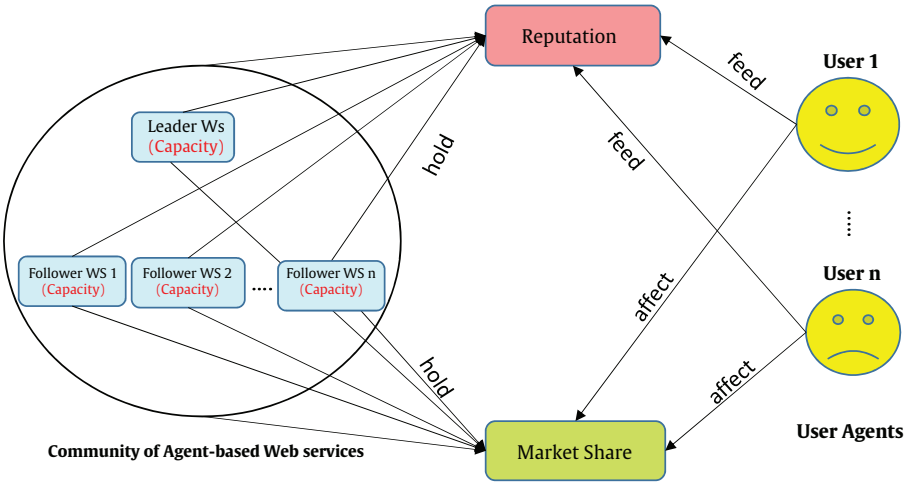

Fig. 1. Model architecture: Web service agents hold a fixed capacity value and variable reputation and market share values that are affected by the user agents.

Stackelberg game models have been widely used in the security domain to characterize the attacker–defender models (Clempner & Poznyak, 2015; Michael & Scheffer, 2011; Pita et al., 2009; Trejo, Clempner, & Poznyak, 2015). In Pita et al. (2009), the authors used a Bayesian Stackelberg to help Los Angeles International Airport (LAX) police officers protect the airport against adversaries. The challenges that led to the use of Bayesian Stackelberg game are related to (1) the impossibility to provide full security coverage at all times; (2) the ability of the adversaries to observe the security arrangements over time and adjust their malicious strategies accordingly, and (3) the uncertainty the police agents face over the type of adversaries (e.g., thieves, terrorists). In the proposed game, police officers play the role of the leader and adversaries are the followers. The goal is to find the optimal strategy the police should commit to, given that followers may be aware of the leader's strategy when choosing their own strategies and that the police is not able to learn the follower's type a priori.

In Michael and Scheffer (2011), a Stackelberg game has been used to model the adversial learning in which the adversary tries to manipulate the data miner's data to reduce the accuracy of the classifier. A Stackelberg game is employed, where the adversary is the leader, the data miner is the follower, and the objective is to help the data miner decide whether to retrain the classifier or maintain the Status Quo given the action of the adversary.

In gross, Stackelberg game theoretical models have been widely used to model different situations in numerous application domains. In particular, they have been extensively investigated in the security domain to characterize the attacker–defender model. In this work, we consider a Stackelberg game to model the Web services communities formation problem with the aim of creating optimal and stable communities in the long-term.

3. The model architecture

In this section, we describe the architecture of our proposed model and discuss each of its components in details. The aim is to clarify the different parameters and concepts that will be used in the rest of the paper. The model architecture is depicted in Fig. 1.

3.1. Agent-based Web service

Web services are software applications providing services to the end users via Web. The emergence of the agent-based software engineering (Jennings, 2000; Wooldridge, 1997) changed the notion of Web services from passive components to autonomous service agents. These agents are able to interact with one another in order to adapt to the real-time challenges (e.g., composition). This has the advantage of moving the micro-level management responsibilities from the user side to the service side. In this way, the user has only to specify the high-level goals (i.e., the "what"), leaving the details of executions (i.e., the "when" and "how") to the service agents (Maamar et al., 2005). Thus, Web services have to manifest a certain level of intelligence and rationality when performing the associated tasks (García-Sánchez et al., 2009; Gibbins et al., 2004; Maximilien & Singh, 2003). Consequently, as agents, Web services are utility-maximizers (Khosravifar et al., 2013; Russell & Peter, 1995). Therefore, these services are motivated to confederate with one another and form communities in order to increase their reputations, market shares, and capacity of handling more requests; which increases their opportunities of being assigned more requests. Each Web service is characterized by a set of internal and external parameters. Internal parameters are referred to as those that are fixed for each service. These parameters are the QoS provided by the service and its capacity to handle users' requests simultaneously. External parameters are those that are affected by the Web service's public actions such as reputation and market share. These parameters are affected in the long-term by the service's internal parameters that indicate how well this Web service performed while serving users' requests.

3.2. Community of agent-based Web services

A community of agent-based Web services is a virtual cluster grouping agent-based Web services sharing the same functionality but having different QoS properties. Communities allow Web services to interact and cooperate to provide higher-levels of quality, performance, and interoperability. For example, Web services within community may replace each other in case of execution problems. Such structure is beneficial for both Web services and users. On the one hand, Web services residing in a community will have more chances to receive bigger task pool from end users and to be exposed to a greater number of compositions. Moreover, by joining communities enjoying good reputation, market share, and capacity, Web services will increase their chances to receive more requests and thus increase their income (Elnaffar, Maamar, Yahyaoui, Bentahar, & Thiran, 2008). On the other hand, users will be more satisfied as their requests will be achieved with better quality, availability, responsiveness, and performance. At the macro-level viewpoint, a community can be seen as a service itself and thus, it holds all internal (capacity, QoS) and external (reputation and market share) characteristics of the single Web services. The community is managed and represented by the

Community Leader, whose responsibility is to monitor and regulate the *health* of the community. This leader faces the challenges of keeping up the performance of the community at an acceptable level, maintaining efficient and scalable task allocation mechanism, and maximizing user satisfaction. In particular, the leader uses a task allocation mechanism to select the service(s) that will be in charge of fulfilling the requests received from customers. The leader is responsible as well for monitoring the behavior of the community members and, if necessary, firing the poor-performing ones that harm the community's image toward users. The use of such a central entity is common in various domains of applications (Mohammed, Otrok, Wang, Debbabi, & Bhattacharya, 2011; Wahab, 2013).

3.3. Reputation mechanism

The concept of reputation has been proposed as a metric to measure the degree of trustworthiness assigned by users to a certain service based on its previous behavior and is used thus to predict the service's future behavior (Wahab, Bentahar, Otrok, & Mourad, 2015). Typically, the process of building the reputation of a certain service goes through the following steps (Liu & Munro, 2012). First, users submit ratings for the services that had interacted with based on their observations. Then, a mathematical model is used to aggregate the ratings given to a certain service and assess thus the reputation of that service. Finally, a dissemination mechanism is used to distribute the reputation information to the end users. Apparently, the most challenging phase in this process is finding the appropriate mathematical model for aggregating the reputation values. Numerous frameworks have been proposed to address this issue. These models can be categorized into simple average-based models, probabilistic and statistics-based models, flow-based models, and fuzzy-logic-based models. In order to avoid the problem of unfair ratings that encounters the average-based aggregation models, we adopt in this paper the model proposed in Whitby, Josang, and Indulska (2005) that accounts for the problem of unfair ratings and uses a Bayesian system to compute the aggregate reputation value. This model is based on the idea that whenever an agent changes its behavior, all honest raters who deal with this agent will change their ratings accordingly. Then, the model compares the overall reputation score of a certain agent with the probability distribution of the ratings assigned by each rater to that agent and defines an upper and lower thresholds according to which raters are considered unfair and thus excluded from the reputation aggregation mechanism. The model uses also a longevity factor to adjust the weights of the received ratings gradually according to their age.

3.4. Market share

The market share denotes the percentage of market, defined in terms of number of requests, accounted for by a specific Web service. In terms of value, the market share associated with a Web service agent is a value between 0 and 1. For example, if a service S_i holds a market share of 0.65, it is deemed to account for 65% of the market (i.e., requests) with respect to other services. As described in Eq. (1), each Web service S_i computes this metric by dividing the number of requests it receives by the total number of requests filed.

$$M_{Si} = \frac{R_{Si}}{\mathrm{TR}} \tag{1}$$

3.5. Capacity

As described in Eq. (2), the capacity assigned to a Web service S_i is an internal fixed parameter that indicates the maximum number of requests this service is able to handle simultaneously.

$$C_{Si} = \frac{\text{Maximum number of requests } S_i \text{ can handle}}{\text{Unit time}} \tag{2}$$

3.6. User agent

The role of user agents is to continuously search for Web service communities that match their requirements (e.g., availability, price, and so on) and generate service requests. Upon accomplishment of their requests, user agents express their satisfaction on the quality of interaction with the Web service/community in terms of feedbacks assigned to that service/community. These feedbacks are used to build the reputation of the Web service/community in the future. Moreover, increasing the number of user's requests will increase the market share of the Web service/community. Thus, Web services and communities are motivated to provide high quality in fulfilling users' request in order to obtain higher reputation scores and attract more requests.

4. Game theory analysis

In this section, we define the aggregation functions used by leaders and followers to compute the variations in their internal and external parameters, derive the utility functions for both leader and follower Web services, formulate the community formation problem as a Stackelberg game model, and compute the equilibrium of the game. It is worth mentioning that the methodology used to analyze the equilibrium is inspired by that used in Zhang and Zhang (2009); however, different parameters and formulas are used in this paper (see the discussions in Section 2.2). The solution can be summarized as follows. Thanks to its superiority in the market (in terms of parameters), the leader has the right to pre-select the set of followers that will be considered for possible confederation (e.g., top 10 Web services). It decides also about the quota of followers that will be considered for final selections (e.g., 5 services out of 10). These leaders publish their offers consisting of reputation, market share, and capacity to the set of pre-selected followers. For the followers, they compete with each other to decide about the payment to be given for the leader under the offered parameters with the purpose of convincing the leader to consider them when selecting the final quota without having to make too much payment. For the leader, the objective is to select the quota of followers that allows him to reach the optimal utility in terms of reputation, market share, capacity, and revenue. Therefore, it is a typical two-stage Stackelberg game model where leader Web service, the leader of the game, optimizes its strategy (i.e., quota) after learning the effects of its decision on the behavior of the followers who play their best response to the leader's offer. To solve the game, the *backward induction* is used to analyze the equilibrium. This is done by finding the optimal payment the followers tend to make as a best response to the leader's offer and substituting this information into the leader's utility function.

4.1. Aggregation functions

The decisions made by the Web services (both leaders and followers) are influenced by four main metrics: reputation, market share, capacity, and payment/revenue. As rational agents, Web services tend to increase their reputation scores to gain more reliability vis-à-vis users, which helps them receive more task pool. Thus, a key factor in forming communities is to ensure that the reputation of the group is at an acceptable level. Therefore, an aggregation function is needed for the Web services (both leaders and followers) to help them calculate their expected reputation score after confederation and compute the effects of this score on their utility functions. It is worth mentioning that we assume that the reputation score is bounded by 0 and 1. We propose a heuristic function each leader L and follower F uses to compute its expected aggregate reputation $E_R(L, F)$ after confederating with one another based on their current reputation scores R_L and R_F respectively. This function is given by Eq. (3):

$$E_R(L, F) = f(R_L, R_F) \tag{3}$$

The function f should satisfy the following properties:

Property 1. *R_L is always greater than R_F.*

This property indicates that the reputation of the leaders is always higher than that of followers.

Property 2. *f is monotonically increasing for the leader if the gap between R_L and R_F is small.*

This property states that the reputation of the leader will increase in a monotonic manner if it decides to confederate with followers having reputation scores that are relatively convergent to its reputation score.

Property 3. *f is strictly decreasing for the leader when the gap between R_L and R_F is big.*

This property says that if the leader selects followers having extremely low reputation scores, its reputation value will then drop dramatically.

Property 4. *f is strictly increasing for the followers if the gap between R_L and R_F* is small.

This property indicates that the reputation of the followers will increase in a considerable manner if they are to confederate with a leader whose reputation value is not that divergent from their reputation scores.

Property 5. *f is monotonically increasing for the followers if the gap between R_L and R_F is big.*

This property says that if the difference between the reputation values of leader and followers is big, then followers will get their reputation values increased but in a monotonic manner.

Property 1 is a common property of Stackelberg games and represents a natural consequence of being a leader. Properties 2 and 3 are important to motivate the leader to select followers having acceptable reputation scores and restrict hence the selection space of the leader to a certain number of well-reputable followers. Properties 4 and 5 will motivate the Web services having low reputation values to improve their performance and get their reputation scores increased in way that gives them the opportunity of being selected by the leader to be part of future communities.

A possible definition of f is given by Eq. (4).

$$E_R(L,F) = \min(R_L, R_F)^{|R_L - R_F|} \tag{4}$$

To illustrate how the aforementioned properties are important for maintaining healthy communities in terms of reputation, we give two examples that explain the aggregation process according to Eq. (4).

Example 1. Suppose a leader having a reputation score of 0.8 decides to confederate with a follower having a reputation value of 0.7, the aggregate reputation value according to Eq. (4) will be: $E_R = 0.7^{0.1} = 0.96$. In this example, the gap between the two reputation values is 0.1, which is relatively small. Thus, the reputation of the leader is increased in a monotonic manner (i.e., 16%), while the reputation of the follower is increased in a strict manner (i.e., 26%). Consequently, both the leader and followers are encouraged to confederate with each other in this example scenario.

Example 2. Suppose now a leader having a reputation score of 0.8 decides to confederate with a follower having a reputation value of 0.2, the aggregate reputation value according to Eq. (4) will be: $E_R = 0.2^{0.6} = 0.38$. In this example, the gap between the two reputation values is 0.6, which is obviously big. Thus, the reputation of the leader is decreased in a strict manner (from 0.8 to 0.38), while the reputation of the follower is increased in a monotonic manner (from 0.2 to 0.38). Consequently, the leader will not be motivated to confederate.

As for the market share, this metric constitutes an important factor for the Web services deciding whether to confederate or not. This metric is an indicator of the competitiveness of a certain community among the other communities. Thus, ensuring an acceptable level of this metric is a critical issue as increasing their competitiveness in the market is a main motive for the Web services to form communities. As is the case for reputation, an aggregation function is needed for the leader and followers to help them calculate their expected aggregate market share and compute the effects of this metric on their utilities. For this purpose, a heuristic function is proposed so that each leader L and follower F uses to compute its expected aggregate market share $E_{\mathrm{MS}}(L,F)$ after confederating with one another based on their current market shares MS_L and MS_F respectively. This function is given by Eq. (5):

$$E_{\mathrm{MS}}(L,F) = g(\mathrm{MS}_L, \mathrm{MS}_F) \tag{5}$$

As reputation and market share are similar in terms of value (i.e., both are in the interval [0, 1]), the aggregation function proposed for the market share should satisfy the properties proposed for the reputation aggregation function. However, the aggregation function for the market share should account for one additional constraint; namely the fact that the market share of a certain service is considered proportionally to the market shares of the other services. Therefore, the aggregation function of the market share should be divided by 2 (represents one leader and one follower) in order to reflect the fact that this aggregate market share will be partitioned between the leader and follower. Thus, a possible definition of g is given by (6):

$$E_{\mathrm{MS}}(L,F) = \frac{\min(\mathrm{MS}_L, \mathrm{MS}_F)^{|\mathrm{MS}_L - \mathrm{MS}_F|}}{2} \tag{6}$$

The capacity has an important effect on the confederation decision as it tells to what extent the possible community will be able to handle simultaneous users' requests. Apparently, getting the Web services together increases the group's ability to deal with more number of simultaneous requests. Thus, the expected aggregate capacity $E_C(L,F)$ for the leader L and follower F after confederating with one another is calculated according to Eq. (7) based on their current capacities C_L and C_F respectively.

$$E_C(L,F) = h(C_L, C_F) = C_L + C_F \tag{7}$$

4.2. Utility functions

The utility function of the follower Web services is defined to be the sum of the variations in the follower's reputation, market share, and capacity after confederating with the leader multiplied by its proportional payment with respect to the payments made by all other followers in the pre-selected set, minus its own payment given to the leader. The reasons behind considering the proportional payment are that (1) the payments made by the group of followers will affect the decision of the leader while making his final decision and will influence thus each follower's utility; (2) the payments made by the other followers are used by each follower to adjust its own payment. Thus, the utility function of each follower F is given by Eq. (8).

$$U_F = \frac{P_F}{\sum_{i \in S} P_i}[\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] - P_F, \tag{8}$$

where $\Delta(R_L)$ is the percentage of variation (e.g., +15%) in the follower's reputation after confederating with the leader L, $\Delta(M_L)$ is the percentage of variation in the follower's market share after confederating with the leader L, and $\Delta(C_L)$ is the percentage of variation in the follower's capacity after confederating with the leader L. P_F represents the payment given by the follower F to the leader, S is the set of followers pre-selected by the leader, and $\Sigma_{i \in S} P_i$ is the sum of payments given to the leader by all the pre-selected followers i in S. The variation percentages in the follower's reputation $\Delta(R_L)$, market

share $\Delta(M_L)$, and capacity $\Delta(C_L)$ are given by Eqs. (9), (10), and (11) respectively:

$$\Delta(R_L) = (E_R(L,F) - R_F) * 100 \tag{9}$$

$$\Delta(M_L) = (E_{MS}(L,F) - MS_F) * 100 \tag{10}$$

$$\Delta(C_L) = (E_C(L,F) - C_F) \tag{11}$$

The utility function of the leader is defined to be the sum of variations in the leader's reputation, market share, and capacity after confederating with the followers set, along with the total payments collected from these followers.

$$U_L = \sum_{F \in S} [\Delta(R'_F) + \Delta(M'_F) + \Delta(C'_F) + P_F], \tag{12}$$

where $\Delta(R'_F)$ denotes the percentage of variation in the leader's reputation (e.g., +5%) after confederating with follower F, $\Delta(M'_F)$ denotes the variation percentage in the leader's market share after confederating with follower F, $\Delta(C'_F)$ denotes the variation percentage in the leader's capacity after confederating with follower F, and P_F denotes the payment earned from each follower F. The variation percentages in the leader's reputation $\Delta(R_F)$, market share $\Delta(M_F)$, and capacity $\Delta(C_F)$ are given by Eqs. (13), (14), and (15) respectively:

$$\Delta(R_F) = (E_R(L,F) - R_L) * 100 \tag{13}$$

$$\Delta(M_F) = (E_{MS}(L,F) - MS_L) * 100 \tag{14}$$

$$\Delta(C_F) = (E_C(L,F) - C_L) \tag{15}$$

4.3. Followers payment selection game

Based on the utility functions illustrated in the previous section, the *backward induction* is used to analyze the equilibrium of the game. Given the reputation, market share, and capacity parameters decided by the leader along with the set S of pre-selected followers, each follower F in S decides an initial payment to make for the leader. This payment represents the variation percentage per unit in the follower's parameters. This value is computed according to Eq. (16):

$$\text{Initial Payment}(F) = [\Delta(R_L) + \Delta(M_L) + \Delta(C_L)]/3, \tag{16}$$

Then, the followers in the same set share their initial payments and compete with one another in a noncooperative game model G to select the optimal payment to give for the leader in such a way to maximize their own utilities. This forms a non-cooperative payment selection $G = \langle S, \{P_F\}, \{U_F(.)\}\rangle$.

Definition 4.1. A payment selection game is

$$G = \langle S, \{P_F\}, \{U_F(.)\}\rangle,$$

where:

- S denotes the set of followers pre-selected by the leader (i.e., the players of the game).
- P_F is the strategy set available for each follower F (i.e., the payment).
- $U_F(.)$ represents the utility function of the follower F.

Informally, each follower $F \in S$ selects its strategy within the strategy set P_F with the aim of maximizing its utility function $U_F(P_F, P_{-F})$, where P_{-F} represents the strategies selected by all other players in S except for F.

Definition 4.2. A payment vector $p = (p_1, \ldots, p_k)$ is a Nash equilibrium of $G = \langle S, \{P_F\}, \{U_F(.)\}\rangle$ if, for every $F \in S$, $U_F(p_F, p_{-F}) \geq U_F(p'_F, p_{-F})$ for all p'_F available for F, where:

- $p_{-F} = (p_1, \ldots, p_{F-1}, p_{F+1}, \ldots, p_n)$, i.e., the payment vector profile P without follower F's payment.
- $(p'_F, p_{-F}) = (p_1, \ldots, p_{F-1}, p'_F, p_{F+1}, \ldots, p_n)$.
- $U_F(p_F, p_{-F})$ is the resulting payment for the follower F given the other followers' payment selection result p_{-F}.

The strategy space is defined to be $P = [P_F]_{F \in S} : 0 \leq p_F \leq \bar{p}$, where $\bar{p}$ denotes the maximal value of payment that may be made. Obviously, the utility function of the followers defined in (8) is continuous in p_F. Thus, derivatives may be used to find the best response function. The Nash equilibrium is obtained by finding the best strategy (i.e., payment) each follower should play in response to the leader's offer and the other followers' strategies (i.e., best-response technique). The continuity of U_F in p_i allows us to use the derivative technique to find the best-response strategy. Thus, the problem can be turned into a problem of proving that the utility function given by (8) is concave down in P_F and then computing P_F when $\frac{\partial U_F}{\partial P_F} = 0$

Theorem 1. *$U_F = \frac{P_F}{\sum_{i \in S} P_i}[\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] - P_F$ is concave down in P_F*

Proof.

$$\frac{\partial U_F}{\partial P_F} = \frac{[\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] \sum_{i \in S, i \neq F} P_i}{(\sum_{i \in S} P_i)^2} - 1 \tag{17}$$

$$\frac{\partial^2 U_F}{\partial^2 P_F} = \frac{-2[\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] \sum_{i \in S, i \neq F} P_i}{(\sum_{i \in S} P_i)^3} < 0 \tag{18}$$

According to Eq. (18), the second order derivative of U_F with respect to P_F is always less than 0, which means that U_F is concave down in P_F. Hence, we get that P_F is optimal for U_F when $\frac{\partial U_F}{\partial P_F} = 0$. □

Theorem 2. *The equilibrium of the game G is given by*

$$P^*_F = \sqrt{[\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] \sum_{i \in S, i \neq F} P_i} - \sum_{i \in S, i \neq F} P_i \tag{19}$$

$$P^*_F = \begin{cases} 0, \\ \quad \text{if } \sqrt{[\Delta(R) + \Delta(M) + \Delta(C)] \sum_{i \in S, i \neq F} P_i} \leq \sum_{i \in S, i \neq F} P_i. \\ \sqrt{[\Delta(R) + \Delta(M) + \Delta(C)] \sum_{i \in S, i \neq F} P_i} - \sum_{i \in S, i \neq F} P_i, \\ \quad \text{if } \sqrt{[\Delta(R) + \Delta(M) + \Delta(C)] \sum_{i \in S, i \neq F} P_i} \geq \sum_{i \in S, i \neq F} P_i. \\ \bar{p}, \\ \quad \text{if } \sqrt{[\Delta(R) + \Delta(M) + \Delta(C)] \sum_{i \in S, i \neq F} P_i} - \sum_{i \in S, i \neq F} P_i > \bar{p} \end{cases} \tag{20}$$

Proof.

$$\begin{aligned} \frac{\partial U_F}{\partial P_F} = 0 &\Rightarrow \frac{[\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] \sum_{i \in S, i \neq F} P_i}{(\sum_{i \in S} P_i)^2} - 1 = 0 \\ &\Rightarrow \frac{[\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] \sum_{i \in S, i \neq F} P_i}{(\sum_{i \in S} P_i)^2} = 1 \\ &\Rightarrow (\sum_{i \in S} P_i)^2 = [\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] \sum_{i \in S, i \neq F} P_i \\ &\Rightarrow \sum_{i \in S} P_i = \sqrt{[\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] \sum_{i \in S, i \neq F} P_i} \\ &\Rightarrow P_F + \sum_{i \in S, i \neq F} P_i = \sqrt{[\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] \sum_{i \in S, i \neq F} P_i} \\ &\Rightarrow P^*_F = \sqrt{[\Delta(R_L) + \Delta(M_L) + \Delta(C_L)] \sum_{i \in S, i \neq F} P_i} - \sum_{i \in S, i \neq F} P_i \end{aligned}$$

□

Taking into account the boundary constraints of the payment value 0 and $\bar{p}$, the equilibrium of the game can be rewritten as in Eq. (20). The algorithm that describes the followers payment selection game is presented in Algorithm 1 . Clearly, the algorithm has a linear complexity in the number of preselected set of followers (i.e., $O(|S|)$).

Algorithm 1: Followers payment selection game.

Input: Pre-selected set of followers S
Input: Leader's reputation R_L
Input: Leader's market share M_L
Input: Leader's capacity C_L
Output: Optimal payment for followers P_F^*
procedure FollowersPhase
  for each follower $F \in S$ **do**
    Compute $\Delta(R_L)$ according to Eq. (9)
    Compute $\Delta(M_L)$ according to Eq. (10)
    Compute $\Delta(C_L)$ according to Eq. (11)
    Compute initial payment according to Eq. (16)
    Get information about the payment of all other followers in S
    Compute P_F^* according to Eq. (19)
  end for
end procedure

Algorithm 2: Leader's utility maximization game.

Input: Quota size $|Q|$
Input: Optimal payment for followers P_F^*
Output: Optimal quota of followers S^*
procedure LeaderPhase
  Pre-select a set of followers S
  Publish reputation, market share, and capacity to S
  Repeat
    Enumerate all possible combinations C in S so that $|C| = |Q|$
    Compute $\Delta(R_F)$ according to Eq. (13)
    Compute $\Delta(M_F)$ according to Eq. (14)
    Compute $\Delta(C_F)$ according to Eq. (15)
    Compute utility $U_L(C)$ based on P_F^* according to Eq. (12)
  Until $C = \arg\max_C U_L(C)$
  $S^* = C$
end procedure

4.4. Leader's utility maximization game

Based on the analytical results of the followers payment selection game, the leader optimizes its strategy by selecting the optimal quota S^* amongst S that maximizes its revenue according to (12). Substituting (19) into (12), the utility of the leader becomes:

$$U_L = \sum_{i \in S} (\Delta(R_i') + \Delta(M_i') + \Delta(C_i') + \left[\sqrt{(\Delta(R) + \Delta(M) + \Delta(C)) \sum_{i \in S, i \neq F} P_i} - \sum_{i \in S, i \neq F} P_i \right]), \quad (21)$$

It is worth mentioning that the size of the pre-selected set of followers tends to be limited to those Web services that are classified as first-class services (in terms of reputation, market share, and capacity). This is due to the aggregation functions proposed in Section 4.1 and that are designed in a way that demotivates leaders from selecting the followers having dramatically bad parameters (i.e., reputation, market share, and capacity). Moreover, the final quota of followers tends also to be small since the leader is aware that the gain and resources will be distributed among all the community members. Consequently, as the community size increases, the share of each member, including the leader, will be decreased. This may motivate some community members to leave the community if they consider that their shares are below expectations. This fact pushes leaders to consider the minimal quota of followers that maximizes their utilities in order to increase their own gains and maintain the stability of their communities. The algorithm of the leader's utility maximization game is given by Algorithm 2 .

As for the complexity of Algorithm 2, it is clear that steps (1) and (2) can be performed in polynomial time (i.e., $O(|S|)$). The main complexity lies in step (5), where the leader has to enumerate all the possible combinations of the preselected set of followers based on the quota size. Thus, step (5) is bounded by $O(|S|^{|Q|})$. Steps (6)–(9) can be executed in polynomial time and take at most $O(|Q|)$. Thus, the performance of the algorithm depends mainly on the size of the quota $|Q|$. As explained earlier, we argue that leaders tend to minimize the size of the preselected set of followers as a result of the design of the aggregation functions presented in Eqs. (3), (5), and (7) that restricts the choice of the leaders to those followers enjoying high reputation, market share, and capacity. Consequently, the quota size tends also to be small, i.e., $|Q| \leq |S|$. These claims are supported in Section 6 by means of simulation experiments.

5. Industrial impact: A complete scenario

In this section, we answer the "why" and "how" questions regarding our model by illustrating why it is useful and how it can be practically implemented in real applications of Web services. Consider a flight booking market consisting of five airline Web services S_1, S_2, S_3, S_4, S_5, and S_6. In this market, S_1 and S_6 are strong enough to take the lead, thanks to their high reputation, market share, and capacity. The fact that these Web services offer the same functionality (i.e., flight booking) and thus target the same customer community makes them in a continuous competition. However, the providers of these services have another *strategic* choice, which may be more beneficial for them rather than adopting pure competitive strategies. More specifically, the high reputation of the leaders S_1 and S_6 creates high demands for their services, especially during promotions and peak times, in such a way that may make their available resources (e.g., bandwidth, storage space) insufficient enough to cope with such demands. In this situation, these leaders have to choose between (1) dropping or delaying some of the incoming requests and (2) cooperating with other services to increase their power in responding to the requests. Obviously, the first choice is quite costly for such services as it may lead to harm their reputation and market share. For example, if S_1 chooses to delay some of its requests, it may end up losing a part of its customers and reputation for the benefit of S_6 and vice versa. On the other hand, follower services (S_2, S_3, S_4, and S_5) are likely to suffer, from time to time, from a lack in the incoming requests, which entails the problem of unused resources. In the same context, these latter services have poor chances to be selected to participate in composition sequences in the presence of stronger services (i.e., the leaders). Therefore, follower services have motivations to cooperate with the other stronger services in order to increase their market share and hence efficiently exploit their unused resources. Besides, structuring the large number of functionally-similar services into a set of communities helps reduce the complexity of building composition sequences since the responsibility of selecting the candidate(s) that will participate in such requests is localized within communities that serve as pockets of functionally-similar services. Thus, both leader and follower services are better off collaborating together in a community-based environment using our proposed model. Customers, in their turn, will benefit from the collaboration between leaders and followers according to our proposed model by enjoying high-quality and possibly cheaper services.

Suppose now that S_1 decides to confederate with some other less strong airline services (i.e., S_2, S_3, S_4, and S_5) to form a strong

Table 1
Airline Web services' parameters.

Service ID	S_1	S_2	S_3	S_4	S_5
Reputation	0.85	0.68	0.66	0.60	0.2
Market share	0.35	0.25	0.2	0.18	0.02
Capacity	20	16	14	12	5

community and gain advantage over the other market leader S_6. The parameters associated with S_1, S_2, S_3, S_4, and S_5 are given by Table 1.

As a first step, S_1 sets minimum requirements for the services it is willing to confederate with. Assume for example that these requirements are given respectively for the reputation, market share, and capacity as follows: $\min_{Rep} = 0.6$, $\min_{MS} = 0.15$, and $\min_{Cap} = 10$. According to Table 1, only the services S_2, S_3, and S_4 satisfy these requirements. Thus, the pre-selection space of S_1 is restricted to only these three services, i.e., $S = \{S_2, S_3, S_4\}$. S_1 sets also a quota of 2 for the services to be selected during the final phase. Now, S_1 publishes its offer consisting of its own reputation, market share, and capacity, O=(Rep=0.85, MS=0.35, Cap=20), to the services in S. Each Web service in S computes the variations in its parameters under the predecided offer and decides an initial payment. To do so, the aggregate reputation after confederation with the leader is calculated according to Eq. (4) as follows:

- $f(R_{S_1}, R_{S_2}) = 0.68^{0.17} = 0.94$
- $f(R_{S_1}, R_{S_3}) = 0.66^{0.19} = 0.92$
- $f(R_{S_1}, R_{S_4}) = 0.6^{0.25} = 0.882$

Similarly, the followers compute their new aggregate market shares according to Eq. (6):

- $g(\mathrm{MS}_{S_1}, \mathrm{MS}_{S_2}) = (0.25^{0.1})/2 = 0.43$
- $g(\mathrm{MS}_{S_1}, \mathrm{MS}_{S_3}) = (0.2^{0.15})/2 = 0.39$
- $g(\mathrm{MS}_{S_1}, \mathrm{MS}_{S_4}) = (0.18^{0.17})/2 = 0.37$

The aggregate capacity is calculated according to Eq. (7):

- $h(C_{S_1}, C_{S_2}) = 20 + 16 = 36$
- $h(C_{S_1}, C_{S_3}) = 20 + 14 = 34$
- $h(C_{S_1}, C_{S_4}) = 20 + 12 = 32$

Thereafter, followers compute the variations in their parameters based on the obtained aggregate functions. For the reputation (Eq. (9)):

- $\Delta(R_{S_2}) = 0.94 - 0.68 = 0.26 = 26\%$
- $\Delta(R_{S_3}) = 0.92 - 0.66 = 0.26 = 26\%$
- $\Delta(R_{S_4}) = 0.88 - 0.6 = 0.28 = 28\%$

For the market share (Eq. (10)):

- $\Delta(\mathrm{MS}_{S_2}) = 0.43 - 0.25 = 0.18 = 18\%$
- $\Delta(\mathrm{MS}_{S_3}) = 0.39 - 0.2 = 0.19 = 19\%$
- $\Delta(\mathrm{MS}_{S_4}) = 0.37 - 0.18 = 0.19 = 19\%$

For the capacity (Eq. (11)):

- $\Delta(C_{S_2}) = 36 - 16 = 20\%$
- $\Delta(C_{S_3}) = 34 - 14 = 18\%$
- $\Delta(C_{S_4}) = 32 - 12 = 20\%$

Based on the computed variations, each Web service decides about an initial payment for the leader consisting of the unitary variation percentage in its parameters according to Eq. (16). The calculations are described in the following:

- $\mathrm{InitialPayment}(S_2) = (26 + 18 + 20)/3 = 21.33$
- $\mathrm{InitialPayment}(S_3) = (26 + 19 + 18)/3 = 21$
- $\mathrm{InitialPayment}(S_4) = (28 + 19 + 20)/3 = 32.33$

Thereafter, the services in S exchange their initial payments and each follower computes its optimal payment. The optimal payment is calculated according to Eq. (20) as follows:

- $P^*_{S_2} = \sqrt{(26 + 18 + 20) \times 53.33} - 53.33 = 5.09$
- $P^*_{S_3} = \sqrt{(26 + 19 + 18) \times 53.66} - 53.66 = 4.48$
- $P^*_{S_4} = \sqrt{(28 + 19 + 20) \times 42.33} - 42.33 = 10.92$

The followers compute now their utilities according to Eq. (8) as follows:

- $U_{S_2} = 5.09/20.49 \times (26 + 18 + 20) - 5.09 = 10.81$
- $U_{S_3} = 4.48/16.01 \times (26 + 19 + 18) - 4.48 = 13.15$
- $U_{S_4} = 10.92/9.57 \times (28 + 19 + 20) - 10.92 = 65.53$

Now, the leader computes its new aggregate parameters after confederating with each of the services in S. For the reputation (Eq. (13)):

- After confederating with S_2, $\Delta(R_{S_1}) = 0.94 - 0.85 = 0.09 = 9\%$
- After confederating with S_3, $\Delta(R_{S_1}) = 0.92 - 0.85 = 0.07 = 7\%$
- After confederating with S_4, $\Delta(R_{S_1}) = 0.88 - 0.85 = 0.03 = 3\%$

For the market share (Eq. (14)):

- After confederating with S_2, $\Delta(\mathrm{MS}_{S_1}) = 0.43 - 0.35 = 0.08 = 8\%$
- After confederating with S_3, $\Delta(\mathrm{MS}_{S_1}) = 0.39 - 0.35 = 0.04 = 4\%$
- After confederating with S_4, $\Delta(\mathrm{MS}_{S_1}) = 0.37 - 0.35 = 0.02 = 2\%$

For the capacity (Eq. (15)):

- After confederating with S_2, $\Delta(C_{S_1}) = 36 - 20 = 16\%$
- After confederating with S_3, $\Delta(C_{S_1}) = 34 - 20 = 14\%$
- After confederating with S_4, $\Delta(C_{S_1}) = 32 - 20 = 12\%$

Next, the leader computes its utility for each possible combinations C of 2 (quota size) among these three services according to Eq. (12). The calculations go as follows:

- $C = \{S_2, S_3\}$, $U_L = (9 + 8 + 16 + 5.09 + 7 + 4 + 14 + 4.48) = 67.57$
- $C = \{S_2, S_4\}$, $U_L = (9 + 8 + 16 + 5.09 + 3 + 2 + 12 + 10.92) = 66.01$
- $C = \{S_3, S_4\}$, $U_L = (7 + 4 + 14 + 4.48 + 3 + 2 + 12 + 10.92) = 57.4$

Thus, the set $s^* = \{S_2, S_3\}$ is chosen by the leader S_1 since it gives the maximal utility value among the other combinations. To show that increasing the size of the community is not always the best choice for the leader, we calculate the utility of the leader S_1 if it decides to add S_5 to the selected set, knowing that S_5 has dramatically low parameters. Following the principles of calculations described above, the optimal payment for S_5 will be $P^*_5 = 0$ (according to the first constraint of Eq. (20)) and the utility of the leader after adding S_5 will decrease considerably from $U_L = 67.57$ to $U_L = 14.57$ (as its reputation, market share, and capacity are significantly decreasing without receiving any payment). It is worth mentioning as well that our approach satisfies the *first mover's advantage* of the Stackelberg game, which states that the utility of the leader tends to be greater than that of the followers as it has the advantage of making the first move. In our example, the utility of the leader $U_L = 67.57$ is greater than all followers' utility ($U_{S_2} = 10.81$, $U_{S_3} = 13.15$, and $U_{S_4} = 65.53$).

6. Simulations and empirical analysis

In this section, we provide empirical results to validate both the theoretical and numerical results obtained in the previous sections. The objective is to study the satisfaction of the intelligent service agents in terms of utility and reputation as well as the satisfaction of the users in terms of the QoS provided to their requests. The simulation application is written in C# using Visual Studio. Web services are modeled as classes; each of which is running as a thread.

The information related to Web services is populated from a real-life dataset that includes 2507 real services operating on the Web (Al-Masri & Mahmoud, 2009). The topic of flight booking has been used for the simulations. This dataset has been analyzed and used in Khosrowshahi-Asl et al. (2015). The dataset is based on a scenario where users send XML-based requests containing the flight dates, origin and destination, number of seats, and type of tickets (i.e., one-way or return) and receive an XML-based response consisting of different flights hosted by different companies along with the related information such as prices, timing, and so on. 200, 000 flights are collected and stored in the database that records the values of nine QoS metrics; namely, throughput, availability, reliability, response time, successability, compliance, latency, accessibility, and cost (Al-Masri & Mahmoud, 2009). The reputation is computed by aggregating the different QoS metrics using the concept of Web Service Relevancy Function (WsRF) presented in Al-Masri and Mahmoud (2007) to rank Web services based on their QoS metrics. The basic idea is to represent the QoS parameters for Web services as a matrix called WsRF matrix in which each row represents a single Web service and each column represents a single QoS parameter. Based on this matrix, QoS parameters are normalized by comparing each element in the WsRF matrix against the maximum QoS value in the corresponding column. Having normalized the QoS parameters, the WsRF value for each service can now be computed by summing up all the normalized QoS parameters for that service. The reputation score is then obtained by normalizing the WsRF value to a value between 0 and 1. Web services are divided into leaders and followers based on their reputation score, market share, and capacity of fulfilling requests.

To verify the effectiveness of our model, we compare it against two other models used as a benchmark, which we call "expected performance model" and "QoS-based model". In the QoS-based model, the leader accounts only for the QoS values of the followers while making his selections of the community members. For the expected performance model, the approach presented in Khosravifar et al. (2013) is used and updated slightly to fit our scenario. In fact, this approach uses a one-stage non-cooperative game-theoretical model to derive a threshold according to which the master (or leader) of the community decides whether to accept a Web service to be part of its community or not. The master compares the expected performance of the community after the joining of a certain Web service with the actual performance considering a risk factor and decides to accept that Web service if the expected performance exceeds the actual performance. The risk factor indicates how flexible the master agent is in loosing its performance. For example, if the risk factor associated with a certain master is 20%, then it would consider any situation in its strategy analysis where estimated performance is more than 80% of the community's current performance. Thus, the master m will accept the joining if $\overline{E_m} > (1 - S_m)E_m$, where $\overline{E_m}$ denotes the expected performance of m, E_m denotes its actual performance, and S_m is the associated risk factor. The performance of a certain Web service x is calculated based on its reputation R_x, market share M_x, capacity C_x, and obtained requests Rq_x according to Eq. (22).

$$E_x = \frac{R_x \times M_x}{|Rq_x - C_x| + 1} \tag{22}$$

This approach has been slightly modified to fit our community formation scenario since it originally tackles the problem of Web services joining preexisting communities (not a distributed formation model). In the modified scenario, the leader compares its expected performance after confederating with each Web service in the preselected set and considers that Web service in the final selection if the expected performance exceeds the actual performance. The risk factor considered in the simulations is 50% (Khosravifar et al., 2013).

We begin by measuring the satisfaction of the Web service agents. The simulations run for 50 iterations. At each iteration, the set of leaders and followers is changed and the average utility for both

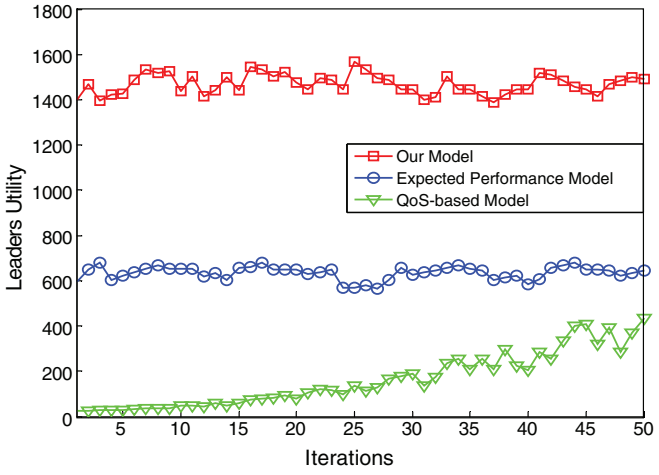

Fig. 2. Leader Web services utility.

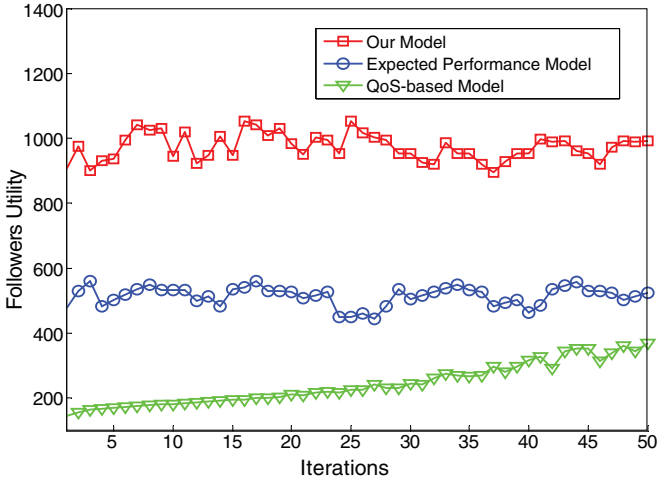

Fig. 3. Follower Web services utility.

leaders and followers is computed. The final selection of followers considered by the leaders represents 35% of the pre-selected followers set. Figs. 2 and 3 describe the average gain of leaders and followers respectively according to our model and the two aforementioned models. As shown in the figures, both leaders and followers can achieve higher utilities by using our model. This is due to the fact that the game in our model is played sequentially at two stages, which makes both leaders and followers aware of each other strategies and enables them hence to play their best responses that maximize their utility. In the expected performance model, the game is played one-shot so that players are playing simultaneously, which limits their learning space and introduces some uncertainty in their decisions. For the QoS-based model, the decision is done according to one particular metric, which makes the selection biased toward one exclusive parameter without considering other important metrics. For example, leaders may choose to confederate with followers enjoying high levels of QoS but having poor market shares and/or capacities, which negatively affects their utility.

We move now to measuring the satisfaction of users by studying the QoS and reputation metrics. After communities are formed, we generated 2500 random requests initiated by 1000 consumer agents to investigate how efficient are the formed communities in fulfilling users' requests. Each request is represented as a class that is characterized by the QoS value needed to perform this task and the QoS

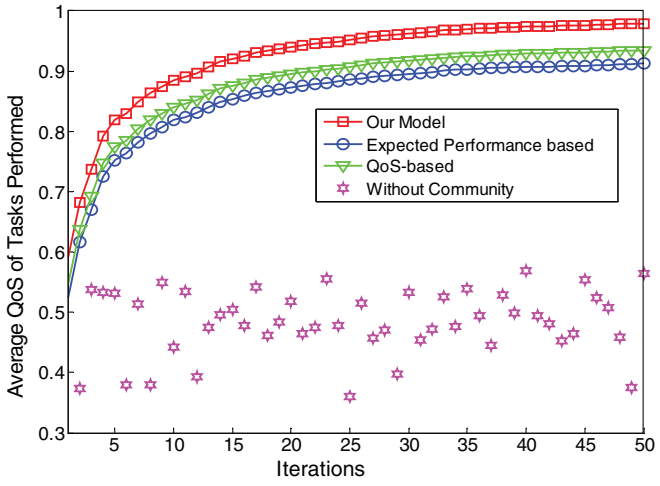

Fig. 4. QoS provided to users' tasks.

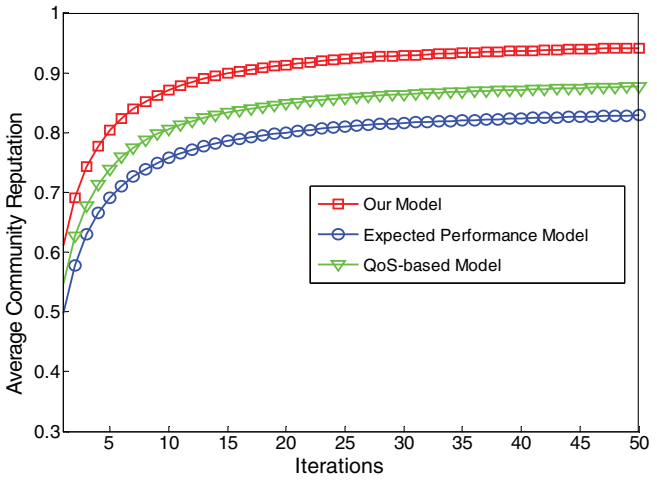

Fig. 5. Communities reputation.

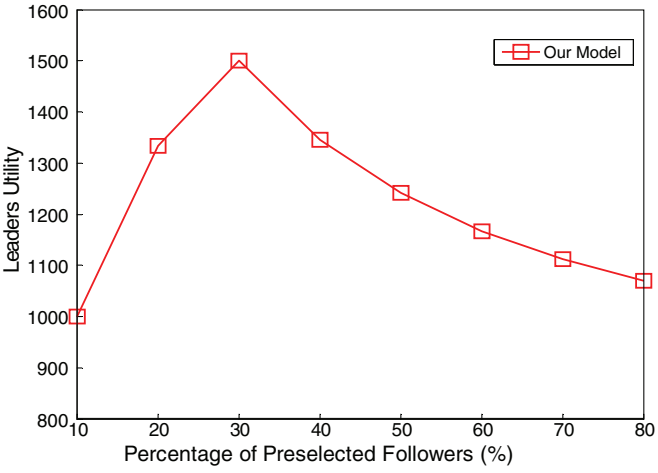

Fig. 6. Preselection size vs. leaders utility.

actually provided to the task. The QoS value is obtained as a result of the aggregation function used in Khosrowshahi-Asl et al. (2015). First, we evaluate the average QoS provided to the users' tasks according to the different aforementioned models while considering an additional scenario, called "Without Community", that represents the case where Web services respond to the requests individually without being organized into communities. Fig. 4 shows that the community-based models (our model, expected performance, and QoS-based models) are able to increase and stabilize the QoS provided to users' requests. This is justified by the fact that the community guarantees higher performance (i.e., availability, response time, and so on) as a result of the cooperation and interoperability that takes place among community members. For example, the community manages to replace the services that fail to respond to the user's request by other community members, which improve the availability of the services and decreases the response time. Moreover, Fig. 4 reveals that our model outperforms the expected performance and QoS-based models in terms of QoS. This is due to the fact that our model enables Web services to maximize their utilities upon forming communities, which makes the community stable and allows it hence to respond to the users' requests with better performance. In contrary, although the QoS-based model selects the Web services based on their QoS values, the services in this model may be encouraged to leave their communities and join other communities if they find better utility, which makes the community unstable and affects hence its performance in fulfilling requests. The same intuition applies for the expected performance model, where the services may be encouraged to leave their communities. As a result, Fig. 5 reveals that our model is able to increase the reputation of the formed communities. In fact, the reputation represents the degree of users' satisfaction concerning the QoS provided to their requests. Upon the accomplishment of a certain task, the community leader rewards/punishes its members based on their performance in fulfilling these tasks. If the QoS offered by the member meets the user's expectations, the member receives a positive reward; otherwise, it receives a punishment. In the former case, the member receives a small reward value assigned by the master in addition to a weighted reward that represents the satisfaction level (i.e., the difference between the QoS provided and the QoS needed). In the latter case, the member receives a default penalty value assigned by the master in addition to a weighted value that is also proportional to the satisfaction level. Each member's reputation is continuously updated by the amount of rewards/penalities it receives.

In order to study the impact of quota size and preselected followers set size on the leader's satisfaction, we vary these sizes and see how these variations influence the utility of the leaders. For example, 10% of preselected followers means that 10% of the follower Web services are preselected by the leader for possible confederation. Similarly, a quota percentage of 20% means that 20% of the preselected followers are considered for final confederation by the leader. As depicted in Fig. 6, the utility of the leaders increases initially with the increase in the size of preselected followers and begins to decrease at a certain point. This is due to the fact that as the size of preselected set continues to increase, the possibility of selecting Web services having drastically bad parameters increases also, which decreases the leader's utility as a result of the aggregation functions described in Section 4. Therefore, leaders tend to restrict this set as much as possible to the top Web services in terms of reputation, market share, and capacity. Thus, we can conclude that Algorithm 2 is computationally tractable. Moreover, Fig. 7 reveals that increasing the quota size results in a continuous increase in the utility of leaders since the leader is selecting from the preselected set, which is already filtered based on the Web services parameters. However, this does not mean that increasing the quota size is always the rational choice for leaders. In fact, the leader is aware that increasing the community size will have some negative side effects on the satisfaction of its community members as well as on its satisfaction since the gain and resources are distributed per member and thus increasing the community size would reduce the share of the community members, including the leader, and affect hence the stability of the group.

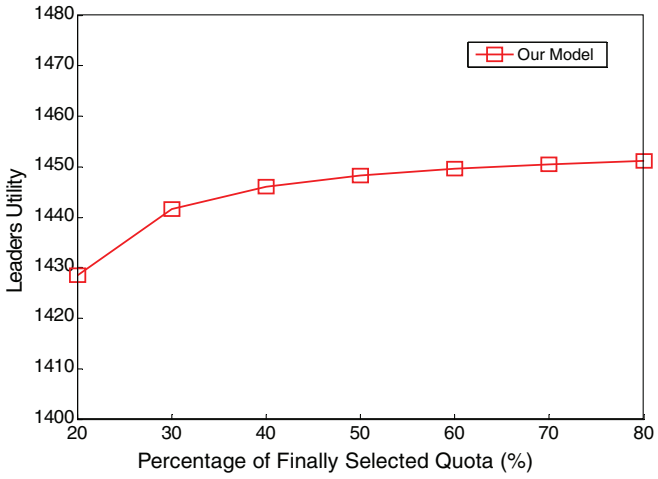

Fig. 7. Quota size vs. leaders utility.

7. Threats to validity

In this section, we shed light on the perceived threats to validity that encounter our work and explain how we address them.

7.1. Construct validity

Construct validity is interested in studying how well experimental results support theoretical foundations (Feldt & Magazinius, 2010). As shown in Figs. 2 and 3, simulation results support the theoretical proofs presented in Section 4 and confirm that the Web services are able to increase their utility by following the equilibrium points computed for leaders and followers in Eq. (20) and (21) respectively. Moreover, Fig. 6 confirms the arguments presented in Section 4.4 that argue that the size of preselected set of followers tends to be small as a result of the design of the aggregation functions derived in Eqs. (3) and (5). More specifically, Fig. 6 reveals that, at a certain point, increasing the size of the preselected set results in a decrease in the leader's utility. Consequently, we can conclude that Algorithm 2 is computationally tractable.

7.2. Internal validity

Internal validity is interested in studying the impact of the input values choices on the outcome of the experimental results (Feldt & Magazinius, 2010). The utility function relies on three main metrics: reputation, market share, and capacity of handling requests. Therefore, choosing wrong or exaggerated values of these parameters has a significant influence on the simulation experiments. In order to mitigate this threat, we linked the reputation of each Web service to the QoS provided by this service and that is gathered and aggregated based on several metrics populated from a real-life dataset (Al-Masri & Mahmoud, 2009). Moreover, we made the market share to be proportional with the reputation. This claim is realistic since the Web services enjoying better reputation scores have higher chances to be selected by users to fulfil their requests. As for the capacity, it is a fixed parameter for each Web service. Therefore, we employed a normal random distribution to estimate this parameter.

Another threat that may encounter our experiments and affect the results is the number of leaders and followers used during simulations. In order to alleviate this threat, we ran the simulations for 50 iterations and varied the set of leaders and followers from one iteration to another. This variation is function of the variation in the reputation, market share, and capacity parameters. At each iteration, we compute the average of the parameter in question (e.g., utility) for the current set of leaders and/or followers.

7.3. External validity

External validity is interested in measuring the extent to which the results are generalizable (Feldt & Magazinius, 2010). Our work uses a representative, well-established dataset that is widely used and publicly available in Al-Masri and Mahmoud (2009). This dataset consists of 2507 real Web service obtained from public sources including Universal Description, Discovery, and Integration (UDDI) registries, service portals, and Web search engines. For each Web service, nine well-representative QoS parameters are considered including response time, availability, throughput, latency, successability, reliability, and compliance. However, our results may not necessarily generalize to all community formation models since we restrict our analysis to three parameters we believe that they are representative and suitable for our model that is mainly business-oriented. Nonetheless, the model can be easily extended by simply appending the intended parameters to the utility function.

8. Conclusions

This paper investigates the problem of community-based cooperation among intelligent Web service agents by modeling the community formation problem as a Stackelberg game model. Our model enjoys three main advantages: (1) it considers a fully distributed environment, where all the services are completely autonomous in their decisions; (2) the community formation scenario is inspired by the real-world business context and the business-related objectives of the Web service agents forming communities are clearly defined; and (3) a two-stage sequential Stackelberg game model is used to ensure the formation of optimal and stable communities in the long-term and the equilibrium point of the game is derived analytically. As confirmed by the simulation results, the proposed model is able to increase the utility and hence the satisfaction of the Web service agents, which increases the stability of the formed communities. Moreover, the model allows communities to achieve high levels of QoS while performing users requests, which increases the user's satisfaction and improves the reputation of the communities.

Promisingly, the result gives guidance to a cooperative model that can be applied in the real markets of Web services for better performance and efficient resources utilization. For example, a leader Web service such as Google that may be overwhelmed by a large number of requests (e.g., during promotion periods) needs to cooperate with other follower Web services in order to respond to the requests in a timely fashion. In their turn, followers benefit from this opportunity in order to efficiently and beneficially exploit their unused resources. Furthermore, the model opens several research directions that may be considered by the research community such as the existence of (1) malicious leaders that publish exaggerated values of their parameters to the preselected sets of followers and/or manipulate their reputation, market share, and capacity values to receive more payments; and (2) malicious followers that proclaim bogus parameters and/or manipulate their reputation, market share, and capacity values to mislead leaders and push them to pre-select/select them for possible confederation.

Acknowledgements

This work was supported by the Fonds de recherche du Québec - Nature et technologies (FRQNT), Natural Sciences and Engineering Research Council of Canada (NSERC), Fonds de Recherche sur la Société et la Culture (FRQSC), 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

Agostino, P., Michele, T., & Paola, T. (2007). An agent-based service oriented architecture. In *Proceedings of the eighth workshop from objects to agents (WOA'07)* (pp. 157–165). Genova, Italy.

Alferez, G. H., & Pelechano, V. (2011). Context-aware autonomous web services in software product lines. In *2011 15th international software product line conference (SPLC)* (pp. 100–109). IEEE.

Al-Masri, E., & Mahmoud, Q. H. (2007). Qos-based discovery and ranking of web services. In *Proceedings of 16th international conference on computer communications and networks (ICCCN)* (pp. 529–534). IEEE.

Al-Masri, E., & Mahmoud, Q. H. (2009). Discovering the best web service: A neural network-based solution. In *Proceedings of the 2009 IEEE international conference on systems, man, and cybernetics* (pp. 4250–4255). San Antonio, TX, USA: IEEE.

Barros, H., Silva, A., Costa, E., Bittencourt, I. I., Holanda, O., & Sales, L. (2011). Steps, techniques, and technologies for the development of intelligent applications based on Semantic Web Services: A case study in e-learning systems. *Engineering Applications of Artificial Intelligence, 24*, 1355–1367.

Benatallah, B., Sheng, Q., & Dumas, M. (2003). The self-serv environment for web services composition. *IEEE Internet Computing, 7*(1), 40–48.

Benslimane, D., Maamar, Z., Taher, Y., Lahkim, M., Fauvet, M. C., & Mrissa, M. (2007). A multi-layer and multi-perspective approach to compose web services. In *AINA 07: Proceedings of the 21st international conference on advanced networking and applications* (pp. 31–37). IEEE Computer Society.

Bianco, P., Lewis, G., & Merson, P. (2008). Service level agreements in service-oriented architecture environments. *Technical Report*. Software Engineering Institute.

Clempner, J. B., & Poznyak, A. S. (2015). Stackelberg security games: Computing the shortest-path equilibrium. *Expert Systems with Applications, 42*(8), 3967–3979.

Elnaffar, S., Maamar, Z., Yahyaoui, H., Bentahar, J., & Thiran, P. (2008). Reputation of communities of web services—Preliminary investigation. In *22nd international conference on advanced information networking and applications, AINA, workshops proceedings, GinoWan, Okinawa, Japan, March 25–28* (pp. 1603–1608).

Eric, N., & Greg, L. (2004). *Understanding SOA with web services* (1st). Addison-Wesley Professional.

Feldt, R., & Magazinius, A. (2010). Validity threats in empirical software engineering research-an initial survey. In *Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering, SEKE 2010* (pp. 374–379). Knowledge Systems Institute Graduate School, Redwood City, San Francisco Bay, CA, USA.

García-Sánchez, F., Valencia-García, R., Martínez-Béjar, R., & Fernández-Breis, J. T. (2009). An ontology, intelligent agent-based framework for the provision of semantic web services. *Expert Systems with Applications, 36*(2), 3167–3187.

Gibbins, N., Harris, S., & Shadbolt, N. (2004). Agent-based semantic web services. *Web Semantics: Science, Services and Agents on the World Wide Web, 1*(2), 141–154.

Gueguen, G. (2006). Between cooperation and competition: The benefits of collective strategies within business ecosystems. The example of the software industry. In *EIASM 2nd workshop on competition strategy* (pp. 1–23). Milan, Italy.

Hamadi, R., & Benatallah, B. (2003). A petri net-based model for web service composition. In *Proceedings of the 14th Australasian database conference: 17* (pp. 191–200). Australian Computer Society, Inc.

Isern, D., Moreno, A., Sanchez, D., Hajnal, A., Pedone, G., & Varga, L. Z. (2011). Agent-based execution of personalised home care treatments. *Applied Intelligence, 34*(2), 155–180.

Jennings, N. R. (2000). On agent-based software engineering. *Artificial intelligence, 117*(2), 277–296.

Kang, J., & Sim, K. (2012). A multiagent brokering protocol for supporting Grid resource discovery. *Applied Intelligence, 37*, 527–542.

Khosravifar, B., Alishahi, M., Bentahar, J., & Thiran, P. (2011). A game theoretic approach for analyzing the efficiency of web services in collaborative networks. In *2011 ieee international conference on services computing* (pp. 168–175). IEEE.

Khosravifar, B., Bentahar, J., Mizouni, R., Otrok, H., Alishahi, M., & Thiran, P. (2013). Agent-based game-theoretic model for collaborative web services: Decision making analysis. *Expert Systems with Applications, 40*, 3207–3219.

Khosravifar, B., Bentahar, J., Moazin, A., & Thiran, P. (2010). Analyzing communities of web services using incentives. *International Journal of Web Services Research, 7*(3), 30–51.

Khosrowshahi-Asl, E., Bentahar, J., Otrok, H., & Mizouni, R. (2015). Efficient community formation for web services. *IEEE Transactions on Services Computing, 8*(4), 586–600.

Li, Z., Zhang, H., & O'Brien, L. (2011). Towards technology independent strategies for SOA implementations. In *6th international conference on evaluation of novel approaches to software engineering (ENASE'11)* (pp. 143–154). Australian National University.

Lim, E., Thiran, P., Maamar, Z., & Bentahar, J. (2012). On the analysis of satisfaction for web services selection. In *2012 IEEE ninth international conference on services computing* (pp. 122–129). IEEE.

Limam, H., & Akaichi, J. (2010). Managing web services communities: A cache for queries optimisation. *International Journal on Web Service Computing (IJWSC), 1*(1), 1–19.

Liu, A., Li, Q., Huang, L., Ying, S., & Xiao, M. (2012). Coalitional game for community-based autonomous web services cooperation. *IEEE Transactions on Services Computing, 99*(3), 387–399.

Liu, L., & Munro, M. (2012). Systematic analysis of centralized online reputation systems. *Decision Support Systems, 52*, 438–449.

Maamar, Z., Mostefaoui, S. K., & Yahyaoui, H. (2005). Toward an agent-based and context-oriented approach for web services composition. *IEEE Transactions on Knowledge and Data Engineering, 17*(5), 686–697.

Maamar, Z., Subramanian, S., Bentahar, J., Thiran, P., & Bensilamane, D. (2009). An approach to engineer communities of web services: Concepts, architecture, operation, and deployment. *International Journal of E-Business Research (IJEBR), 5*(4), 1–21.

Maximilien, E. M., & Singh, M. P. (2003). Agent-based architecture for autonomic web service selection. In *Workshop on web-services and agent-based engineering (WSABE)* (pp. 1–3). Citeseer.

Medjahed, B., & Bouguettaya, A. (2005). A dynamic foundational architecture for semantic web services. *Distributed and Parallel Databases, 17*, 179–206.

Michael, B., & Scheffer, T. (2011). Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 547–555). San Diego, California, USA: ACM.

Mohammed, N., Otrok, H., Wang, L., Debbabi, M., & Bhattacharya, P. (2011). Mechanism design-based secure leader election model for intrusion detection in manet. *IEEE Transactions on Dependable and Secure Computing, 8*(1), 89–103.

Muegge, S. (2013). Platforms, communities, and business ecosystems: Lessons learned about technology entrepreneurship in an interconnected world. *Technology Innovation Management Review, 3*(2), 5–15.

Nan, G., Mao, Z., Yu, M., Li, M., Wang, H., & Zhang, Y. (2014). Stackelberg Game for Bandwidth Allocation in Cloud-based Wireless Live-streaming Social Networks. *IEEE Systems Journal, 8*(1), 256–267.

Paolucci, M., & Sycara, K. (2003). Autonomous semantic web services. *IEEE Internet Computing, 7*(5), 34–41.

Pita, J., Jain, M., Ordóñez, F., Portway, C., Tambe, M., Western, C., et al. (2009). Using game theory for Los Angeles airport security. *AI MAGAZINE, 30*(1), 43–57.

Russell, S. J., & Peter, N. (1995). *Artificial intelligence: a modern approach*. Upper Saddle River, NJ: Prentice-Hall, Inc.

Torrès-Blay, O. (2009). *Économie d'entreprise: Organisation, stratégie et territoire* (3rd). Amazon.

Trejo, K. K., Clempner, J. B., & Poznyak, A. S. (2015). A Stackelberg security game with random strategies based on the extraproximal theoretic approach. *Engineering Applications of Artificial Intelligence, 37*, 145–153.

von Stackelberg, H. (1934). *Marktform und Gleichgewicht*. New York: Springer-Verlag Wien. http://books.google.ca/books?id=wihBAAAAIAAJ.

von Stackelberg, H. (2011). *Market structure and equilibrium* (Translation). Berlin Heidelberg: Springer.

Wahab, O. A. (2013). *Cooperative clustering models for vehicular ad hoc networks*. Lebanese American University Master's thesis.

Wahab, O. A., Bentahar, J., Otrok, H., & Mourad, A. (2015). A survey on trust and reputation models for Web services: Single, composite, and communities. *Decision Support Systems, 74*, 121–134.

Wahab, O. A., Otrok, H., & Mourad, A. (2013). VANET QoS-OLSR: QoS-based clustering protocol for vehicular ad hoc networks. *Computer Communications, 36*(13), 1422–1435.

Wahab, O. A., Otrok, H., & Mourad, A. (2014a). A cooperative watchdog model based on Dempster-Shafer for detecting misbehaving vehicles. *Computer Communications, 41*, 43–54.

Wahab, O. A., Otrok, H., & Mourad, A. (2014b). A Dempster-Shafer based tit-for-tat strategy to regulate the cooperation in VANET Using QoS-OLSR protocol. *Wireless Personal Communications, 75*(3), 1635–1667.

Whitby, A., Josang, A., & Indulska, J. (2005). Filtering out unfair ratings in Bayesian reputation systems. *The ICFAIAN Journal of Management Research, 4*(2), 48–64.

Wooldridge, M. (1997). Agent-based software engineering. *IEE Proceedings—Software, 144*(1), 26–37.

Yahyaoui, H., Maamar, Z., Lim, E., & Thiran, P. (2013). Towards a community-based, social network-driven framework for Web services management. *Future Generation Computer Systems, 29*(6), 1363–1377.

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., & Sheng, Q. Z. (2003). Quality driven web services composition. In *Proceedings of the 12th international conference on world wide web (WWW'03)* (pp. 411–421). ACM.

Zhang, J., & Zhang, Q. (2009). Stackelberg game for utility-based cooperative cognitive radio networks. In *Proceedings of the tenth ACM international symposium on mobile ad hoc networking and computing* (pp. 23–32). New Orleans, LA: ACM.