Using reputation and adaptive coalitions to support collaboration in competitive environments

Ana Peleteiro, Juan C. Burguillo

Department of Telematics Engineering. University of Vigo. Spain
{apeleteiro,J.C.Burguillo}@gti.uvigo.es

Michael Luck

Department of Informatics, King's College, London
michael.luck@kcl.ac.uk

Josep Ll. Arcos, Juan A. Rodríguez-Aguilar

IIIA-CSIC, Artificial Intelligence Research Institute, Barcelona, Spain
{arcos,jar}@iiia.csic.es

Abstract

Internet-based scenarios, like co-working, e-freelancing, or crowdsourcing, usually need supporting collaboration among several actors that compete to service tasks. Moreover, the distribution of service requests, i.e., the arrival rate, varies over time, as well as the service workload required by each customer. In these scenarios, coalitions can be used to help agents to manage tasks they cannot tackle individually. In this paper we present a model to build and adapt coalitions with the goal of improving the quality and the quantity of tasks completed. The key contribution is a decision making mechanism that uses reputation and adaptation to allow agents in a competitive environment to autonomously enact and sustain coalitions, not only its composition, but also its number, i.e., how many coalitions are necessary. We provide empirical evidence showing that when agents employ our mechanism it is possible for them to maintain high levels of customer satisfaction. First, we show that coalitions keep a high percentage of tasks serviced on time despite a high percentage of unreliable workers. Second, coalitions and agents demonstrate that they successfully adapt to a varying distribution of customers' incoming tasks. This occurs because our decision making mechanism facilitates coalitions to disband when they become non-competitive, and individual agents detect opportunities to start new coalitions in scenarios with high task demand.

Keywords: coalitions, collaboration, reputation, crowdsourcing, competitive environments

1. Introduction

In real world domains, individuals usually face the problem of solving tasks, composed of subtasks, that cannot be achieved by them individually; to address this, they

need to group together in order to be able to accomplish such tasks with guarantees. This may be the case when supporting collaboration in new Internet-based scenarios, like co-working [34], or crowdsourcing [30], which are becoming increasingly important. In these scenarios, customers submit tasks to be serviced, with several actors competing to service them. To make this more complex, however, the number and rate of service requests changes over time, as does the service workload (the number of subtasks per tasks) required by each customer.

Over the past decade, crowdsourcing has emerged as a cheap and efficient method of obtaining solutions to simple tasks that are difficult for computers to solve but possible for humans. Crowdsourcing markets bring together requesters, who have tasks they need to perform, and workers, who are willing to perform these tasks in a timely manner in exchange for payment [30]. There are several examples of crowdsourcing platforms, such as Amazon's Mechanical Turk [13] or oDesk [oDesk], and the popularity of crowdsourcing markets has led to empirical and theoretical research on the design of algorithms to optimize various aspects of these markets, such as the assignment of tasks [15, 14]. Thus crowdsourcing has appeared as a new application domain to model and analyze the problem of online decision making, as well as design algorithms to tackle it. Online decision algorithms have a rich literature in operations research, economics, and several areas of computer science including machine learning, theory of algorithms, artificial intelligence, and algorithmic mechanism design [30]. However, in the case of crowdsourcing, as tasks are usually not too complex, workers are typically recruited individually, without considering the possibility of recruiting groups of people to jointly perform more complex tasks. Such complex tasks include those demanded in several domains, such as in international commerce, bidding for government contracts or continuous auctions.

For example, producing a magazine, an academic paper, or a new movie may involve many individuals working in structured teams, each with different skills and roles, collaborating on a common goal. In this way, a fixed pool of workers may need to be allocated to perform tasks that arrive dynamically and that must be completed by some deadline. Crowdsourcing therefore poses unique challenges for both workers and requesters ranging from job satisfaction to direction-setting, coordination, and quality control [16]. However, currently crowdsourcing faces many challenges that must be addressed in order to achieve all of its potential. Kittur et al. [16] present a framework that envisions a future of crowd work enabling more complex, creative, and highly valued sustainable collaborations and co-working. In fact, they present several research challenges in crowdsourcing areas; in response, in this paper we explore a new crowdsourcing model based on their approach.

Most crowdsourcing platforms share the common feature of repeated interaction. In this respect, Afsarmanesh et al. [3] confirm that long-lived groups (those that last in duration beyond the servicing of a single job) are successfully used in real world scenarios, such as in manufacturing or ICT, among others. According to them [3], when groups are long-term creations, as in the case of our coalitions, successful repeated collaborations help these groups to enhance their service performance over time, since these repeated interactions improve agents collaboration. Therefore, to fully benefit from coalition-based collaborations, we must learn how to form coalitions as well as how to sustain them. However, sustaining a coalition poses two main challenges: (i)

how to cope with agents within the coalition that do not honor their commitments; and (ii) how to compete with other coalitions that offer the same services. To tackle these problems requires that a coalition, as a whole, continuously adapts to remain competitive, i.e., in order to have high probabilities of being assigned tasks. Indeed, in an open environment, several competing coalitions may be formed with the aim of performing the very same service. Thus, on the coalition side, this requires the capability of: (i) composing the most appropriate set of agents to perform a service; and (ii) deciding when to disband the coalition because it is no longer beneficial. Moreover, agents immersed in such competitive environments must also individually adapt by deciding: (i) whether to remain in a coalition or join another existing one; and (ii) whether to remain part of a coalition or to leave it in order to start up a new one. Therefore, both coalitions and agents require decision-making mechanisms that allow them to adapt and to remain competitive over time.

Slivkings et al. [30] propose specific directions to tackle the design of a crowdsourcing model: adaptive task assignment, dynamic procurement, repeated principal-agent problem, reputation systems, and the exploration-exploitation tradeoff. In this paper, we mainly focus on the first of these, also using reputation as a way to assess the risk of cooperating with others. However, while we propose to use coalitions of agents to perform complex tasks, most previous work on task allocation with coalitions does not consider how coalitions can be maintained over time in the face of a change once they are formed. For this reason, Klusch et al. [17] develop a dynamic coalition formation scheme (DCF-S) that helps agents react to changes in their set of goals and in the agent society. Similarly, Soh et al. [32] present a dynamic coalition formation mechanism in which learning mechanisms are used at several levels to improve the quality of the coalition formation process in a dynamic, noisy, and time-constrained domain. Nonetheless, such approaches suffer from several shortcomings. First, they mainly focus on supporting the formation of a single coalition for a single task. Thus, they do not consider the bigger picture (and more realistic situation), where there are several coalitions competing to provide the very same service. In fact, most previous work has commonly assumed that a coalition disbands when the current task is finished. Hence, a coalition disappears after the coalition fulfills its goal. Mérida-Campos et al. [22] explore this in the context of iterative games, where several coalitions compete to be assigned tasks in several rounds. They present a dynamic coalition formation mechanism where coalitions must adapt at each time step in order to be assigned more tasks. However, with their mechanism, agents use a pre-established strategy for joining or abandoning partners and, while there is adaptability regarding coalition composition, the adaptation of the the number of coalition is not specifically addressed , i.e., how many coalitions are necessary when they compete with each other to obtain tasks.

In response to these omissions, in this paper we present a model to build and adapt coalitions so that the can be assigned complex tasks with the goal of improving the quality and quantity of the tasks completed. Thus the key contribution in this paper is a decision mechanism that allows agents in a competitive environment to autonomously enact and sustain coalitions, not only in their composition, but also in the number of coalitions that are necessary depending on the incoming tasks. Two key components in such a mechanism are: the reputation, both of coalitions as a whole, and the reputation of individual agents; and the strength of collaboration *synergies* (resulting from

successful repeated collaborations) within coalitions. Reputation has been shown to be effective to assess the risk of cooperating with other individuals. The notion of synergy captures the insight that working together repeatedly improves cooperation among humans. In our model, when agents employ our decision mechanism, we show that it is possible for them to maintain high levels of customer satisfaction (in terms of percentage of services finished on time). Note that we focus on customer satisfaction because quality of service is a current and major problem in crowdsourcing solutions, since there are no guarantees that a service will be good enough. In more detail, we provide the following contributions.

- First, we provide a decision making mechanism for coalitions to help them continuously adapt to remain competitive, i.e., to have high probabilities of being assigned tasks. On the one hand, our mechanism allows a coalition to assemble the most reliable team of agents to service a certain task based on the reputation of agents. On the other hand, the mechanism also helps a coalition decide whether the coalition should be sustained or otherwise disbanded because it is no longer beneficial.

- Second, we provide a decision making mechanism that allows agents to remain competitive, i.e., to have high probabilities of being assigned subtasks. On the one hand, our mechanism allows an agent to decide whether to continue to remain as part of a coalition, or instead to join another coalition. Such a decision is based on: (i) the strength of the successful repeated collaborations of an agent within its coalition; and (ii) the overall reputation of the coalition. On the other hand, our mechanism allows an agent to decide when to start a new coalition.

- Finally, we provide an empirical analysis showing that the usage of our mechanisms by agents makes it possible to maintain high levels of customer satisfaction (percentage of tasks serviced on time). Here, we show that coalitions demonstrate high resilience: even when the percentage of reliable agents is low ($\sim$ 40%), the percentage of serviced tasks on time is beyond 80%. In addition, coalitions and agents demonstrate that they adapt to a varying distribution, i.e., the arrival rate, and characteristics, of customers' incoming tasks. Thus, we obtain $\sim$ 95% of tasks serviced on time despite significant variations in the incoming arrival rate and characteristics of tasks.

Altogether, we aim at providing an interesting and simple model for managing new emerging coalitions, composed by humans who work with new technologies. The subject of investigation in this paper can be somewhat complex depending on the conditions and the assumptions introduced in the model and implemented through the simulations. However, in order to ensure clarity, the assumptions underlying our agent-based model are deliberately simple, as our goal is to understand the fundamental processes. This is the approach followed throughout this work.

The paper is organized as follows. First, in Section 2 we present related work. Then, Section 3 presents a computational model for an environment in which multiple coalitions compete to service tasks. Next, Section 4 describes the decision making mechanisms employed by coalitions and agents during task allocation and execution,

and Section 5 introduces the adaptive mechanism employed by coalitions and agents. Section 6 details our empirical analysis. Finally, Section 7 draws conclusions and sets paths to future research.

2. Related work

In this section we present a review of related work in the literature. We begin with a brief review of coalition formation for task allocation, followed by related work on crowdsourcing scenarios.

2.1. Coalition formation for task allocation

In multi-agent systems, agents may face the problem of achieving tasks that are composed of subtasks that cannot be achieved by individual agents alone. Groups of agents are not only necessary when tasks cannot be performed by a single agent, but may also be beneficial when groups perform tasks more efficiently than individual agents [28]. Thus, given a set of agents and a set of tasks, the problem we are concerned with is deciding how to form coalitions to achieve tasks so as to maximize the total profit [20] (where profit can be understood in many interpretations, including as utility). Ideally, a coalition formation mechanism would allow agents not only to form coalitions for joint task execution, but also to achieve a coalition configuration which is optimal (in terms of utility maximization), stable, and fair [18]. However, the computational complexity required for such solutions is exponential [26]. Moreover, in most real-world scenarios we do not need coalitions to be optimal, but suboptimal, and formed in a dynamic manner.

As argued in [26, 28, 27, 17, 5], task allocation via coalition formation follows a three step process: (i) generating the coalition structures; (ii) selecting which structure will be adopted, and (iii) distributing gain between agents. At the same time, however, it is hard to specify one general framework for coalition formation. This is why most work tries to solve the coalition formation problem in a concrete environment that establishes certain constraints. Despite this, in [5], Amgoud provides a unified formal framework for constructing such coalitions structures. Her framework returns three semantics of coalition structures: the basic, which returns a unique coalition structure; and two different refinements of the basic — the stable and the preferred — each of which may return several coalition structures at a time. This framework is general enough to capture different proposals in the literature.

Task allocation coalition formation problems can be studied in cooperative or a non-cooperative environments. Coalition formation in cooperative environments has been investigated by Shehory et al. [28, 29], who assume that information about other agents can be known or communicated. Also in a cooperative environment, Lau et al. [20] propose a classification for the coalition formation problem, based on three driving factors: task demands, which are the quantities of service that are demanded; resource constraints, i.e., whether the resources are limited or unlimited; and an objective function, which is the profit obtained from achieving a task. In this work, Lau et al. also explore the runtime complexity and propose algorithms for each category.

Zheng et al. [36] present an approach where, unlike previous approaches, each agent can participate in coalitions for different tasks. They develop several efficient

and effective greedy hill-climbing strategies for determining both which agents belong to which coalitions (for their relevant task) and when these coalitiosn should start executing to achieve their tasks. In the context of non-cooperative environments, Aknine et al. [4] propose two methods for coalition formation, where agents cannot exchange their knowledge, in contrast to cooperative multi-agent systems. Abdallah et al. [1] propose to use an underlying organization to guide the coalition formation process, using Q-learning with neural nets to optimize decisions made locally by agents in the organizations. This underlying organization can be viewed as a search tree, which is modified, depending on the model environment and the agent population, to achieve the best performance.

However, the approaches above do not consider how coalitions can be maintained over time in the face of change once they are formed. In response, Klusch et al. [17] develop a dynamic coalition formation scheme (DCF-S) in an environment where agents have goals they cannot accomplish by themselves. Their dynamic mechanism helps agents react to changes in their set of goals and in the agent society. In this DCF-S scheme, leaders for each coalition (CLAs) are used to concurrently simulate, select, and negotiate coalitions that are able to accomplish at least one of their goals with an acceptable ratio between estimated risk of failure and individual profit. Soh et al. [32] present a scenario in which agents are not completely cooperative, but cautiously cooperative, i.e., they are not always willing to help, but only in the case that they obtain some benefit from doing so. Here, learning mechanisms are used at several levels to improve the quality of the coalition formation process in a dynamic, noisy, and time constrained domain. Moreover, the agent that initiates a coalition has the responsibility of overseeing and managing the formation process. Ye et al. [35] propose a dynamic coalition formation mechanism, endowed with self-organization, in a structured agent network. Based on self-organization principles, their mechanism enables agents to dynamically adjust their degrees of involvement in different coalitions and to join new coalitions at any time. The authors take the agents themselves to be the entities form the coalitions.

While this work on dynamic coalition formation has provided valuable contributions, it largely focuses on supporting the formation of a single coalition for each task. Thus, it generally does not consider the perhaps more appropriate scenario in which there are several coalitions competing to provide the same service. This kind of scenario can be found in such environments as international commerce, bidding for government contracts or continuous auctions, which have been explored by Mérida-Campos et al. [22], focussing on iterative games, where several coalitions compete to be assigned a task in several rounds. In this work it is addressed how to adapt the coalition composition, but it does not take into consideration the adaptation of the number of coalitions.

With the same idea but in a different context, Mérida-Campos et al. [23, 24] focus on the effects of heterogeneous tasks on the coalition adaptation process in a heterogeneous population of agents, namely: competitive, where agents try to be in a coalition with maximal competence; and conservative, where agents try to be in a coalition with optimal competence / size ratio in order to maximize their benefits. They investigate how agents in a heterogeneous population cluster together across multiple coalition formation episodes and varying tasks, and observe that the competitive strategy outper-

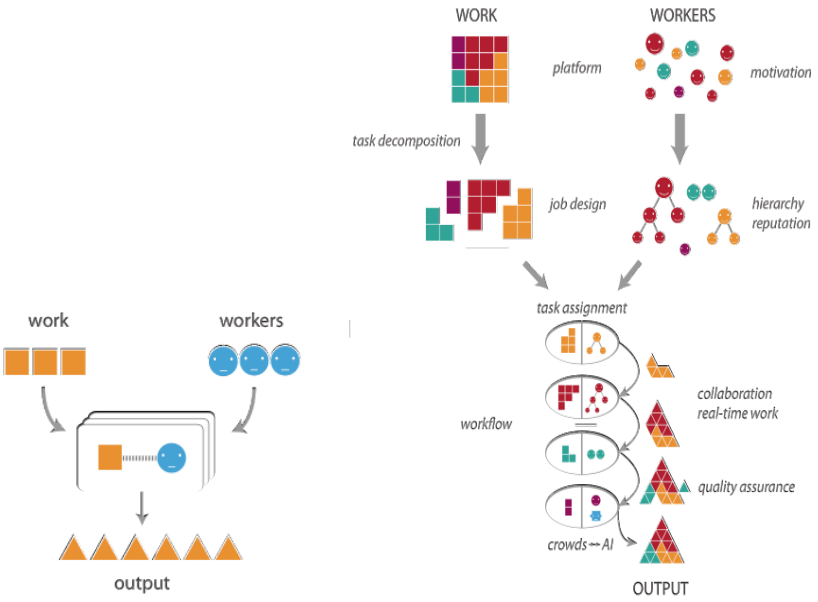
forms the conservative one.

Lappas et al. [19] introduce the problem of team formation in social networks, taking into account the compatibility of two agents when cooperating together. The focus of the work is to minimize the coordination cost, but ignoring the issue of balancing the workload. Anagnostopoulos et al. [7] study algorithms that allocate an incoming stream of tasks, so that no-one is overloaded or unfairly singled out. However, they ignore the coordination costs. To tackle this, Anagnostopoulos et al. [6] present online algorithms that assemble teams to deal with tasks, in a way that keeps coordination costs bounded and results in a fair allocation of the workload. In our work, we are not interested in work balancing, since, in this framework, we assume that the best available agents should perform the task, nor in the coordination costs, since we have mediators that are in charge of the communication and coordination of the agents.

2.2. Dynamic coalition formation (DCF) for crowdsourcing

Related to dynamic task allocation, over the past decade, crowdsourcing has emerged as a cheap and efficient method of obtaining solutions to simple tasks that are difficult for computers to solve, but possible for humans. In fact, crowdsourcing markets bring together requesters, who have tasks they need accomplished, and workers, who are willing to perform these tasks in a timely manner in exchange for payment. Thus crowdsourcing has appeared as a new application domain for online decision making algorithms, opening up a rich and exciting problem space in which the relevant problem formulations vary significantly along multiple modeling dimensions [30]. This popularity of crowdsourcing markets has led to both empirical and theoretical research on the design of algorithms to optimize various aspects of these markets, such as the assignment of tasks and pricing. In addition, several researchers have taken an interest in modeling and analyzing the problem of online decision making in crowdsourcing markets. However, current crowdsourcing work typically consists of a set of small, independent, and homogenous tasks. For example, Kittur et al. [16] criticize current crowdsourcing models (Figure 1a) and present the future of crowdsourcing (Figure 1b), that outlines a framework to enable complex and sustainable collaborations. In their view, complex tasks should be decomposed into smaller subtasks. These subtasks must be assigned to appropriate groups of workers that must be adequately selected (e.g., through reputation), and organized (e.g., through hierarchy). There may be a workflow in the workers' performance of tasks. Finally, quality assurance is needed to ensure each worker's output is of high quality and fits together. Similarly to their views, we also envision a future crowdsourcing where complex tasks composed of subtasks are dynamically created and where groups of agents are formed and adapted to the needs of the customers. This is why our mechanisms address several of the challenges identified in [16], such as: dynamicity, by presenting a mechanism that allows to achieve tasks that change dynamically over time; group formation, in order to collectively perform complex tasks; hierarchy, since we have different roles with different responsibilities; and reputation, since we consider not only reputation for individual workers, but also for the coalition.

There are several examples of crowdsourcing platforms, such as Amazon Mechanical Turk [13], that focus on small microtasks, e.g., filling out a survey, with small payments; or other platforms, such as oDesk [oDesk], that focus on larger jobs like



(a) Current crowdsourcing.

(b) Future crowdsourcing.

Figure 1: Comparison between current and future crowdsourcing approaches [16].

designing websites, for significantly larger payments. Most of these platforms share the common feature of repeated interaction. However, to the best of our knowledge, no crowdsourcing site offers the functionality of forming and sustaining groups for large tasks composed of subtasks. Thus, given a large task composed of multiple subtasks, all crowdsourcing sites enforce a contractor to separately contract each subtask, possibly from multiple workers that operate individually. Moreover, Amazon's Mechanical Turk has been overly criticized for its reputation system, since it needs a mechanism that provides more accurate information and takes more of the interaction between the two parties into account. [1] [2]

While there have also been several recent empirical or applied research projects aimed at developing online decision making algorithms that function effectively in practice on existing crowdsourcing platforms [8, 12], these are are generally useful only in their specific domain. From a more theoretical perspective, Slivkings et al. [30] present a detailed and up-to-date discussion of the modeling issues that inhibit theoretical research on repeated decision making in crowdsourcing. They point out that despite the vast scope of work in crowdsourcing, it brings several domain-specific challenges that require novel solutions. Moreover, they suggest that to address these challenges

[1] http://www.technologyreview.com/view/416966/how-mechanical-turk-is-broken/

[2] http://www.behind-the-enemy-lines.com/2010/10/plea-to-amazon-fix-mechanical-turk.html

in a principled way, one would like to formulate a unified collection of well-defined algorithmic questions with well-specified objectives, allowing researchers to propose novel solutions and techniques that can be easily compared, leading to a deeper understanding of the underlying issues. However, it appears very difficult to capture all of the pertinent aspects of crowdsourcing in a coherent model. As a result, many of the existing theoretical papers on crowdsourcing propose their own new models, making it difficult to compare techniques, and leading to uncertainty about which parameters or features matter most when designing new platforms or algorithms.

Given the above, Slivkings et al. [30] propose specific directions to tackle the design of a crowdsourcing model: adaptive task assignment, dynamic procurement, repeated principal-agent problem, reputation systems, and the exploration-exploitation tradeoff. In this paper, we are mainly focused on the first of these, but with the specific difference that instead of assigning subtask to individuals, we assign complex tasks to coalitions, with the goal of maximizing the quality and quantity of completed tasks subject to budget constraints. Generally, in the task assignment problem, strategic issues are ignored in order to gain analytical tractability. In fact, the model typically does not touch on the way in which prices are set, and does not include workers' strategic responses to these prices. In the most common variant of this problem, workers arrive online and the requester must assign a task (or sequence of tasks) to each new worker as she arrives. Karger et al. [14, 15] introduce one such model for classification tasks and proposed a non-adaptive assignment algorithm based on random graph generation along with a message-passing inference algorithm inspired by belief propagation for inferring the correct solution to each task. They prove that their technique is order-optimal in terms of budget when each worker finds all tasks equally difficult. In contrast, Ho et al. [10, 11] show that adaptive task assignment yields an improvement over non-adaptive assignment when the pool of available workers and set of tasks are diverse.

However, to the best of our knowledge, none of these previous approaches uses coalitions to model the problem of complex task assignment, but instead focus on individually assigning simple subtasks. In addition, they do not consider that different groups may apply for the very same task, thus competing with each other. These are the challenges that we address in the rest of this paper.

3. Computational model

The purpose of this section is to outline the computational model of a competitive environment in which agents are allowed to autonomously enact and sustain coalitions. With this aim, we consider that such an environment is instantiated as a scenario in which customers dynamically generate requests for their tasks to be serviced on time. By *dynamic* we mean that: (i) the customers' task arrival rate changes over time; and (ii) the workload required by each task may also vary. Within our environment, coalitions, which are groups of agents that join together, compete to service tasks. Once a task is assigned to a coalition, it may either complete the task on time or not. Over time, some coalitions may disappear (because they are no longer competitive), while others may be formed.

We begin with an example in order provide a better understanding of the model. Suppose a customer submits a new task, which consists of writing a book. This task

requires three workers: one that writes, one that corrects, and one that does the drawings. Therefore, the customer sends the task to be serviced by the system. However, the customer need not choose individual agents, since groups of agents are dynamically formed in order to service these tasks. Indeed, suppose that there is a leader agent, which acts as a mediator between the customer and the agents that will write the book. This agent can form a new group in order to compete with other mediators for the task. Therefore, in this scenario, several teams containing a writer, a corrector and a drawer may be created, and from those teams that apply to service the task, their prior performance can be used to choose one. Within this selected team, the worker agents can deliver their subtask on time or incur a delay, affecting the process of delivering the whole task on time. Therefore, the way in which individual workers perform their duties should affect the probability of being selected to become part of a coalition in the future. In addition, the way in which a team has performed will also influence the likelihood of the coalition being selected by a customer for a future task. Now, if an agent writer, that belongs to a team, has not been hired to work on a new task for a long time, then it might consider forming its own coalition, or changing to a new one. Similarly, if a mediator agent does not receive tasks to service due to the coalition it is leading not having a good reputation, then, it can also decide to become a worker (e.g., a writer) to find an alternative route to increasing its gains.

Formally, we represent each task request generated by some customer as a tuple (T_i, d_i), where T_i is the specification of a task to be serviced, and d_i is the deadline by which it must be completed. A task T_i is composed of a set of subtasks $\langle \tau_1^i, ..., \tau_{k_i}^i \rangle$, where each subtask requires some skill to be fulfilled, from a finite set of skills $S = \langle s_1, .., s_m \rangle$. In our environment, there is a set of agents $Ag = \{ag_1, ..., ag_n\}$ with different skills, and a coalition here is simply a group of agents, a subset of Ag, which gather together to perform some task. In this context, multiple coalitions compete to service task requests. Since agents may fail to fulfill their commitments, we say that a task is serviced on time when all subtasks are serviced on time, and serviced with delay when at least one subtask has not been serviced on time.

Here we assume that each coalition is led by a *mediator* agent. Such mediators have been used extensively in multi-agent systems because they play the important role of assisting in locating and connecting the providers of a service with its requester [9, 21, 33]. In our particular case, a mediator will be also responsible for the management of the composition of a coalition, a function that, according to [2, 3], is extremely important in supporting a coalition's activities. Thus, a mediator leading a coalition is responsible for searching for the agents to be part of a coalition, henceforth referred to as *worker* agents, assembling teams of workers to perform tasks, and evaluating the performance of workers. In general, a coalition (led by a single mediator) can service several tasks at the same time, depending on its mediator's *capacity*. We refer to each group of workers that perform a task within a coalition as a *team*. Thus, a coalition may contain several teams performing separate tasks at the same time.

Since the mission of a worker agent is to perform subtasks within a task, a worker must have the necessary skill to carry out a subtask. However, a worker may fail to complete a subtask on time, and a coalition may therefore fail to service a task on time because some of its workers may fail to complete their individual subtasks on time. Notice that we assume that an agent can take the role of either worker or mediator, but

never both at the same time. Moreover, for the sake of simplicity, a mediator can lead a single coalition and a worker can only belong to one coalition at the same time, since, in this framework, we assume that each coalition competes with others. Also, a worker cannot be part of more than one team at a time, since we consider that all its capability must be focused on a single task.

Figure 2 illustrates the components of our competitive environment. Customers submit tasks to be serviced, which come into the environment as a dynamic stream of tasks. These tasks are collected by a contractor, which is in charge of assigning tasks to coalitions using a contract net protocol (CNP) [31]. Note that in order to avoid complexity, and without loss of generality, we consider only one contractor in the system. The figure shows several coalitions $Coalition_1$, $Coalition_2$, and $Coalition_3$, as well as three independent agents (ag_1,ag_2, and ag_3) that do not belong to any coalition. Each coalition has a mediator agent (agents within hexagons in the figure). Upon task completion, the contractor rates the quality of the service provided by a coalition and, similarly, coalitions also rate their own workers. This rating information is maintained and aggregated by a reputation module.

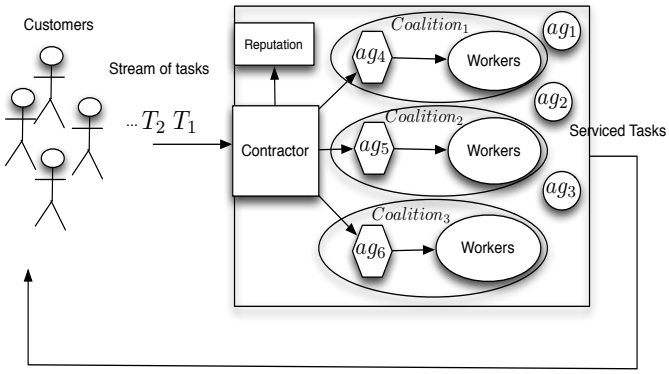


Figure 2: Competitive environment.

In Table 1 we detail several steps that describe how the cyclic process that is used in our competitive environment, addresses the servicing of incoming tasks. At the end of this cycle, a new incoming task takes the process to Step 1 again, i.e., it is a sequential process. Note that from here on, when we refer to steps, we mean the steps identified in Table 1.

To illustrate the coalition formation and adaptation processes, Figure 3 depicts an example showing a transition in the number and composition of coalitions within our competitive environment. The figure shows the coalitions at two different moments in time. Agents acting as mediators are within hexagons, while agents acting as workers are within circles. First, the top half of the figure shows the existing coalitions when a request to service task T_1 arrives. At that point there are two mediator agents (ag_1 and ag_2), each one leading a coalition composed of worker agents ($\{ag_3, ag_4, ag_5\}$ and $\{ag_6, ag_7, ag_8\}$ respectively). Of the agents within each coalition, some have been selected to be part of a team. There are also three independent agents that are not

1. Request for coalitions. For each incoming task, the contractor broadcasts a request, (T_i, d_i), to all coalitions.
2. Team formation. Once a coalition receives a task request, its mediator selects the best available team (a subset of agents in the coalition) to service the request. If the necessary agents are not available within the coalition, the mediator can contact free agents or agents within other coalitions. Then, a team is formed by selecting one agent per subtask.
3. Acknowledgement. If a coalition can put together a team of agents to service the task, it replies to the contractor that it can do so by the deadline d_i.
4. Task assignment. From all the positive replies received from coalitions, the contractor assigns the task to the coalition with the highest reputation (to avoid overfitting, with a certain small probability, p_{nov}, the task is randomly assigned to a coalition). Therefore, it follows that the greater the reputation of a coalition, the more competitive, and hence the greater the chance of it being awarded tasks to service. Note that to avoid the cold start problem, the reputation of coalitions at the beginning is the same for all the coalitions, so tasks are randomly assigned in this case.
5. Task execution. The coalition that is assigned a task starts the team that must service the task.
6. Task reward. Once a task is serviced, each worker in the servicing team obtains a reward for completing its subtask. The mediator also obtains a reward for servicing the task, which is higher than that of the workers. This is intended to compensate for the responsibility and the effort of a mediator in selecting, coordinating, and evaluating workers for the team. Moreover, the mediator takes a greater risk in accepting tasks initially.
7. Coalition and agent evaluation. Once a task is serviced: (i) the contractor evaluates the performance of the coalition in terms of the delay in servicing the task; and (ii) the coalition evaluates the performance of each member of the team that performed the task. Both evaluations are provided to the reputation module, where they are maintained and aggregated.
8. Coalition and agent adaptation. Since coalitions and agents must adapt to remain competitive, at this point a coalition (that has no pending tasks to service) may decide to disband, and a worker (without pending subtasks) may decide to form a new coalition.

Table 1: Steps that describe the cyclic process to serve incoming tasks.

part of any coalition (ag_9, ag_{10}, ag_{11}). Later on, after servicing several tasks, up to T_{10}, the bottom half of the figure shows the new distribution of coalitions. Now, several changes have occurred: agent ag_6 left $coalition_2$ to start and lead $coalition_3$ with agents $\{ag_9, ag_{10}, ag_{11}\}$; agent ag_1 disbanded $coalition_1$ to join $coalition_2$ as a worker; and agents ag_3, ag_4 and ag_5 became independent. Note that independent agents cannot compete with coalitions, because we assume that each task is always composed of more than one subtask, and hence more than one agent is required to complete it.

So far we have focused on describing the computational model of our competitive environment. In the following sections we focus of the decision making that coalitions and agents require to participate in such an environment.

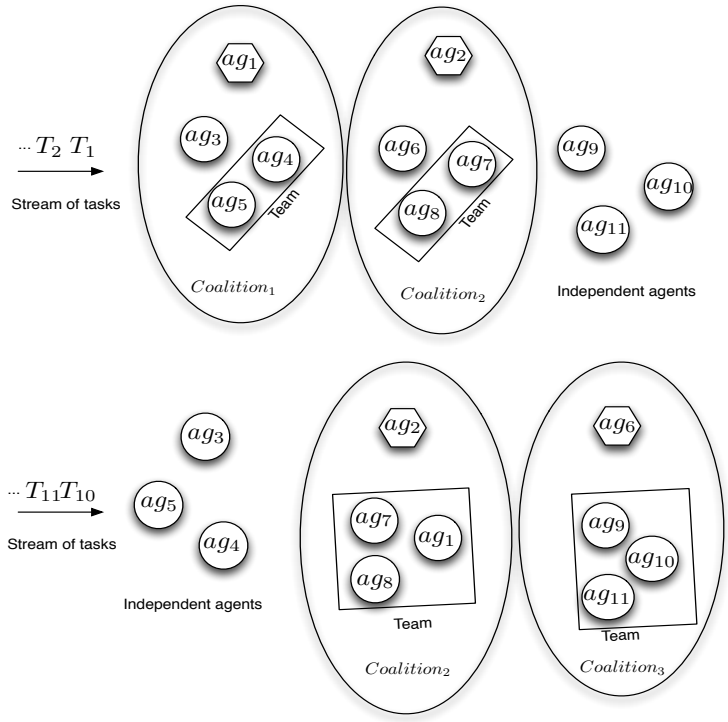


Figure 3: Possible evolution of the number and composition of coalitions over time in our competitive environment.

4. Task allocation and execution

In this section, we present the decision making of coalitions and agents involved in team formation, task assignment and execution, and evaluation (Steps 2-7, Table 1). Thus we describe how a coalition: (i) decides and gathers the most appropriate set of agents to service a task; and (ii) how a coalition evaluates its team. Moreover, we detail how an agent decides which coalition to join or, if it already belongs to one, whether it should switch. Finally, we also describe how reputation is aggregated in the reputation module from Figure 2.

4.1. Mediator decision making

Every time a task arrives, each mediator leading a coalition has three main responsibilities. First, it must form a team out of the best available agents in the coalition

to service the task (Step 2). Second, it must perform the task whenever it has been awarded to the coalition (Step 5). Third, once a task has been serviced, it must evaluate the performance of the workers in the servicing team (Step 7). In Algorithm 1, we specify the mediator's general behaviour for team formation, task assignment and execution, and performance evaluation (Steps 2 to 7 in Table 1).

Algorithm 1 Coalition formation and performance evaluation

```
1:  function RECEIVEREQUEST((T_i, d_i))
2:      Team = BroadcastRequest(myCoalition, ⟨{τ^i_1, ..., τ^i_{k_i}}, d_i⟩)
3:      if not Complete(T_i, Team) then
4:          AO = BroadcastRequest(AgentsOutsideCoalition)
5:          Team = add(Team, AO)
6:      if Complete(T_i, Team) then
7:          Assigned = send(contractor, ACK, (T_i, d_i))
8:          if Assigned then
9:              AddToCoalition(AO)
10:             ExecuteTask(T_i, Team)
11:             Receive(rwd_m)
12:             for all ag_j ∈ Team do
13:                 EvaluateWorkers(CoalitionEval)
14:             send(evaluations, contractor)
15:         Release(Team)
```

Team formation (Step 2) We focus first on team formation, i.e., we present how a coalition (represented by its mediator) decides and gathers the most appropriate set of agents to service a task. Given a task, a mediator must first find a set of workers that satisfy the skills and time constraints required by the task. There may be different ways to choose these agents, but we assume that repeated interactions with the same agents improve task performance. For instance, as result of task supervision, a mediator understands better the strengths and limitations of its workers, and continues to improve its understanding the longer they work together. Therefore, the mediator first sends a request to the workers in its coalition (line 2). A request is a tuple $\langle\{\tau^i_1, ..., \tau^i_{k_i}\}, d_i\rangle$, where $\tau_1, ..., \tau_{k_i}$ are the subtasks that compose the task, and d_i the period of time to service it. After receiving the responses from workers, if the mediator cannot find enough workers in its coalition to carry out the task, it also sends requests both to independent agents or to agents that belong to other coalitions (line 4). Notice that if a mediator was not allowed to search beyond its coalition, it would not be possible to perform adequately in a dynamic environment. This may happen for two reasons. First, since the incoming task load or task characteristics may change, there may not be agents with required capabilities within its coalition. Second, a coalition may have a sufficient number of agents to service a task, but they might already be busy servicing other tasks.

Once there are sufficient agents, the mediator must select from them. Now, recall that agents may fail to deliver their subtask on time. Thus, to make the coalition com-

petitive, a mediator uses a selection criterion based on preventing failures: choose the most reliable agents, namely those with highest reputation. In turn, to determine an agent's reputation, the mediator consults the contractor, which is in charge of assessing reputation. Notice that a worker can be part of a coalition without belonging to a team at some particular moment. Finally, note that for a newly created coalition, its reputation will depend on the performance of the coalition since the moment it is formed, but not considering the past history of the workers in other coalitions.

Task assignment and execution (Step 3-5) Now, we focus on task assignment and task execution. Once a mediator has formed its own team (line 5), it acknowledges to the contractor that it can perform the task. Note that all mediators provide the same information, stating just the task they can perform. Then, to prevent failures, the contractor assigns the task to the most competitive coalition, which we define as the one with highest reputation. Note that in order to avoid overfitting, instead of assigning all tasks to coalitions with higher reputations, there is a small random probability of selecting any of them. The coalition that obtains the task (line 8) starts servicing it (line 10), while the coalitions that do not obtain the task dissolve the teams they had formed (line 15).

In this paper, we do not employ cost to assign tasks and subtasks, since that would require that agents to bid, and therefore adopt a bidding model. We believe this adds unnecessary complexity and is not the focus of the work in this paper. By not considering cost, we implicitly assume that all agents have similar costs, and are thus differentiated only by their reputation. We therefore focus on how reputation influences interactions between agents and coalitions, as a key challenge in crowdsourcing, already highlighted by Kittur et al. in [16]. Thus, while it is an interesting area for exploration, studying extensions to our model that consider cost parameters is left as future work.

Team evaluation (Step 6-7) After workers in a team have serviced their subtasks, the mediator delivers the serviced task to the contractor. The mediator receives a reward ($rwd_m > 0$) (line 11) and evaluates its workers (line 13). To evaluate workers, each coalition uses a decay function $CoalitionEval$ that evaluates the delivery time of an agent, where a longer delay gives a larger penalty in reputation. Once the evaluation is computed, this value is sent to the contractor (line 14). Note that even if an agent changes its coalition, its reputation remains.

4.2. Assessing coalition and agent reputation

The contractor is in charge of assessing reputation (i) of the coalition, depending on how the team has performed; and (ii) of agents, depending on how they have individually performed. Using the reputation module (Figure 2), reputation for both coalitions and agents is updated by combining an evaluation of the performance in the current task with the current reputation. Thus we balance between past and current reputation, which is a simple way of calculating reputation. In Equations 1 and 2 we show how the reputation of a coalition coa_z and an agent ag_j is updated. Note that while many different reputation mechanisms are possible, that is a separate problem, and it is beyond of the scope of this paper to specify a complex reputation mechanism, as well as being unneccesary.

$$Rep(coa_z) = \alpha_1 \cdot ContractorEval(d_i, \tilde{d}(coa_z, T_i)) + (1 - \alpha_1) \cdot Rep(coa_z) \quad (1)$$

$$Rep(ag_j) = \alpha_2 \cdot CoalitionEval(ag_j)(d_i, \tilde{d}_j) + (1 - \alpha_2) \cdot Rep(ag_j) \quad (2)$$

where $\tilde{d}(coa_z, T_i)$ is the time the coalition took to finish its task; d_i is the time requested to service the task; $\tilde{d}_j$ is the time the worker ag_j took to finish its subtask; and α_1 and α_2 are factors that model the influence of current reputation. In addition, $CoalitionEval$ and $ContractorEval$ are decay functions to evaluate the delivery time of an agent and a coalition, respectively, where a longer delay gives a larger penalty in reputation. In Section 6.1 we propose a concrete evaluation function.

4.3. Worker decision making

In this section, we focus on how an agent decides the coalition to join, or if it already belongs to one, whether to switch. This is a critical decision, since the possibility of obtaining a subtask depends on two factors: (i) how competitive an agent is; and (ii) how competitive its coalition is. The latter factor arises because tasks are assigned according to the reputation of each coalition (Step 4), as explained in Section 4.1. Thus, since a worker may have several requests to perform subtasks from different coalitions, it must decide which coalition to join to be part of a team. Moreover, if the agent already belongs to a coalition, it must decide whether leave or stay (Step 2). We therefore endow each worker with a local stochastic decision making mechanism to make such decisions; i.e., worker agents do not have a global perspective of the system, and they cannot communicate with each other. Note that even if allowing agents to have a global view could be beneficial for them, we wanted our model to resemble reality as much as possible, i.e., to simulate an environment where workers do not have full perspective of the available groups to join, but only to the ones that have an offer for them.

Algorithm 2 Coalition selection

```
1: function SubtaskProposalsAndCoalition(Requests)
2:     Collect(Requests);
3:     p^j_stay = Calculate(RepCoa(ag_j), CollSynCoa(ag_j))
4:     if (SampleofBernoulli(p^j_stay) then)
5:         send(Coalition(ag_j), "accept");
6:     else
7:         send(Coalition(ag_j), "leave");
8:         send(CoalitionHighestRep, "join");
9:     send(OtherCoalitions, "reject");
```

In Algorithm 2 we specify a worker's behavior after receiving several requests to perform subtasks. First, a worker collects all the requests it has (line 2). If it already belongs to a coalition then, to decide whether it should switch to another one, it must

consider: (i) how well its current coalition is doing, i.e., how competitive its coalition is in terms of obtaining tasks; and (ii) whether it is being selected by the mediator to be part of teams, i.e., if it is doing well within the coalition. These factors correspond to the reputation of a coalition coa_i ($Rep(coa_i)$), and the *collaboration synergy* of the agent in the coalition ($CollSynCoa(ag_j)$). We introduce the concept of collaboration synergy to reflect the fact that repeated collaborations improve performance when humans interact. Thus, the collaboration synergy between a worker and the coalition it belongs to represents how well the worker is performing, in terms of obtaining subtasks in the coalition. This is assessed as the number of subtasks that a worker has performed without changing from one coalition to another, and considering the total number of subtasks it could have performed within the coalition. Note that this value is reset every time an agent changes its coalition. Thus to calculate the probability of staying in a coalition, we trade off exploration (reputation) and exploitation (collaboration synergy). Formally, we define the probability of an agent ag_j staying in its current coalition coa_i in Equation 3:

$$p^j_{stay} = \beta \cdot Rep(coa_i) + (1 - \beta) \cdot CollSynCoa(ag_j) \qquad 0 \leq \beta \leq 1 \tag{3}$$

Once a worker has computed p^j_{stay} (line 3), which follows a Bernoulli distribution, it decides whether to stay in its current coalition or not. Note that again we have opted for a simple model to avoid complexity in simulations. If the worker decides to stay, it sends its acceptance to its current coalition (line 5). If it decides to leave, it notifies its former coalition (line 7) and joins the coalition with the highest reputation (line 8). In both cases, it sends a rejection to any other coalition requests (line 9).

Finally, if the task is not assigned to the coalition (the coalition receives a "reject", line 9), the worker is released from the team. In contrast, if the task is assigned (Step 4, Table 1), the worker starts performing its subtask until it is completed (Step 5). At that point, the worker notifies the mediator, receives a reward $rwd_w > 0$ (Step 6) and is evaluated by the mediator (Step 7). After this, it is free to perform another subtask.

5. Adaptive virtual organizations

In this section we present our decision making for coalitions and agents to allow them to adapt in order to remain competitive in a dynamic environment (Step 8, Table 1). Thus we present how: (i) a coalition may disband; and (ii) an agent may start a new coalition. In Figure 4 this mechanism is shown as a stochastic automaton. Thus each agent has two possible states, either being a mediator or a worker. With probability $p_{m \to w}$, a mediator changes its role to worker; and with probability $p_{w \to m}$, a worker changes its role to mediator. Since the behavior of an agent in each state has already been specified (for mediators in Section 4.1, and for workers in Section 4.3), we are now able to focus on this policy to change state.

In the following sections we specify the local stochastic decision making mechanisms for mediators and workers, allowing them to change their roles depending on their own knowledge.

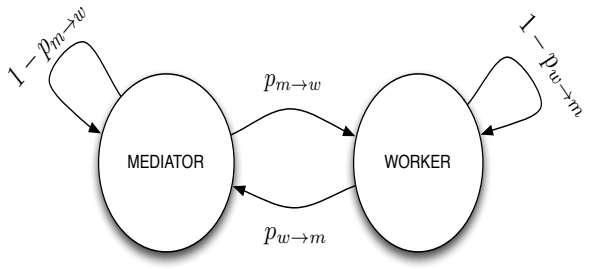


Figure 4: Change of roles modeled as a stochastic automaton.

5.1. Mediator adaptation

In this section, we specify how to determine when to make the transition from mediator to worker as shown in Figure 4, i.e., the probability $p^j_{m \to w}$. As stated previously, a mediator is in charge of forming and adapting coalitions to obtain new tasks. The higher the reputation of a coalition, the more competitive it is (since tasks are assigned according to coalition reputation), and the higher the probability of being awarded tasks. However, since there are always possible reasons for failure, especially in dynamic environments, there might come a time when a coalition is no longer valuable. To assess this, we specify the local decision making mechanism by which a mediator decides whether to change its role, and hence disband the coalition it leads.

To decide whether it pays off to remain as a mediator, we define u_m as mediator utility, which measures the actual utility of being a mediator given the number of tasks assigned and the rewards as a result of them. We also define $\tilde{u}_w$ as the estimated worker utility, which is an estimation of the number of tasks a mediator would have participated in if it had been a worker instead. To calculate $\tilde{u}_w$, a mediator uses an optimistic approach, since it assumes that every time it is not busy servicing a task, it would have been assigned a subtask if it was a worker.

Again, in order to avoid excessive complexity, we assume that recent past experience is the most useful indication of future performance. Thus each mediator only calculates its utilities for a time window Δt before the current time, in order to discard the influence of performance in the distant past (since performance changes over time). Thus, in Equations 4 and 5 we specify two utility functions for a mediator that measure how well it performed in the recent past:

$$u_m(\Delta t) = N_t(\Delta t) \cdot rwd_m \tag{4}$$

$$\tilde{u}_w(\Delta t) = N_s(\Delta t) \cdot rwd_w \tag{5}$$

where $N_t(\Delta t)$ is the number of tasks that a mediator has coordinated during the last time window; $N_s(\Delta t)$ is the number of subtasks that it could have performed as a

worker during Δt, adopting the optimistic approach mentioned above; and rwd_m and rwd_w are the rewards for being a mediator and a worker, respectively.

Once $u_m(\Delta t)$ and $\tilde{u}_w(\Delta t)$ are calculated, a mediator can determine their ratio to decide its preferred role. This is useful in deciding whether to change or not, since if the utility of being a worker is higher than that of being a mediator, then the probability of becoming worker becomes higher. Equation 6 shows the probability of a mediator becoming a worker ($p^j_{m\to w}$), which we compute by using the $u_m(\Delta t)$ and $\tilde{u}_w(\Delta t)$ ratios, and limiting the obtained value to one. Once a mediator has computed this probability, which follows a Bernoulli distribution, it decides whether to change from one state to the other (Figure 4).

$$p^j_{m\to w}(\Delta t) = \frac{\tilde{u}_w(\Delta t)}{u_m(\Delta t)} \tag{6}$$

5.2. Worker adaptation

In this section, we specify how a worker determines when to make the transition to mediator shown in Figure 4, i.e., the probability $p^j_{w\to m}$. We consider two situations in which a worker may benefit from changing its role: (i) when it is not requested to service subtasks (either because it is not performing as it should, or because there are insufficient tasks); or (ii) when it is busy servicing subtasks all the time. In the first case, a worker should try to become a mediator to obtain some reward since it is clearly not succeeding as a worker. In the second case, since it is busy all the time, it assumes the workload is high, and thus may assume that by becoming a mediator it could receive a higher reward. Note that even if more mediators mean more competition, we will see that by using our adaptive mechanism, if the workload is not high enough, a mediator becomes a worker, in this way avoiding the problem of unserviced tasks not having enough workers.

In this case, in contrast to the mediator, which can estimate its utility as a worker, a worker cannot compute its estimated utility as a mediator, since a worker is not aware of the number tasks that are assigned (a mediator has extra information as it sits between the workers and the contractor). Thus, we define a function where the probability of changing from worker to mediator ($p^j_{w\to m}$) increases with the time a worker is both idle and busy. In Figure 5 we present the basic function used to calculate $p^j_{w\to m}$, which is specified in Equation 7.

$$f_{ib}(t_p) = \begin{cases} \dfrac{2\cdot t_p^2 - 150}{650} & t_p \leq -10 \quad (7a) \\ \dfrac{1}{2}\dfrac{t_p^2}{650} & -10 < t_p \quad (7b) \end{cases}$$

where t_p represents a period of time. Negative values of t_p represent periods of time when a worker is idle, while positive values represent busy periods. As we observe in the figure, workers increase their probability of becoming mediators as they increase the period of time they are idle or busy. The reason for the different slopes is that if an agent is idle for a long period ($t_p \leq -10$), its probability of becoming a mediator

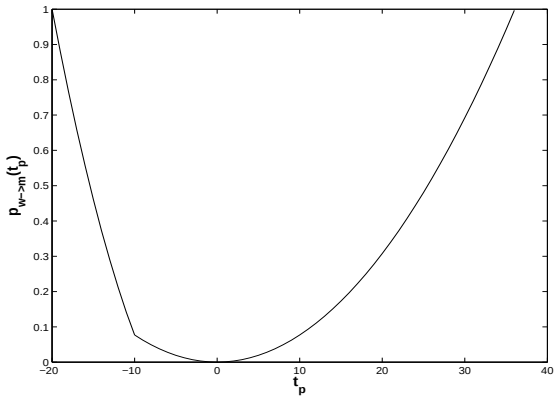


Figure 5: Probability of becoming a mediator for workers.

must be higher than if it is busy for the same period of time (positive values of t_p). The reason for this is that while it is busy, it is still obtaining some benefit (reward), and becoming a mediator could imply an unnecessary risk.

In addition, each agent applies to this probability a decay factor of q^{-wm_j}, with $q > 1$ and wm_j is the number of times a worker j has tried to become a mediator. We introduce this decay factor to model the fact that being a mediator implies more effort, since it must coordinate a coalition. Finally, once a worker has computed the probability $p^j_{w\to m}$, described in Equation 8, which follows a Bernoulli distribution, it decides whether to change from one state to the other (see Figure 4).

$$p^j_{w\to m}(t_p) = q^{-wm_j} \cdot f_{ib}(t_p) \tag{8}$$

6. Experiments

In the previous sections, we provided a decision mechanism that allows agents in a competitive environment to autonomously enact and sustain coalitions. Now, in our experiments we show how, by employing our decision mechanism, it is possible to maintain high levels of customer satisfaction, in terms of the percentage of tasks serviced on time. First, in Section 6.2, we analyze the resilience of coalitions to the failure of workers, i.e., to workers not servicing their subtasks on time. Second, in Section 6.3 we focus on how our adaptive mechanism allows coalitions to adapt to dynamic changes in task load. Before our analysis, we describe our empirical settings.

6.1. Empirical settings

For each experiment, we ran ten multi-agent simulations with 225 agents, and we present medians and variances. Unless otherwise stated, each task is composed of eight subtasks, and each subtask is managed by one agent, so eight workers are necessary to service a task. For the sake of simplicity, we assume that all the subtasks to be serviced require the same skill, so all agents are potentially eligible to service any subtask. Also,

for the sake of simplicity, we assume that all mediators have the same capacity, which we fix to one task. We define the probability to avoid overfitting (p_{nov}) to 0.01.

Agent behavior. We need to specify some parameters in order to simulate the behavior of the agents. First, a worker may finish its subtask with a certain delay. To model this we specify a probability, p_f^j, different for every agent, which is the internal probability of a worker j finishing on time.

We have also assumed that repeated interactions improve collaborative performance. To model this, we specify that the probability of a worker finishing on time depends not only on p_f^j, but also on the number of times that a worker has collaborated with a coalition (collaboration synergy, $CollSynCoa(ag_j)$). In Equation 9 we specify the combined probability as p_F^j. Once calculated, it is sampled with a Bernoulli distribution to assess whether the worker has finished on time or not.

$$p_F^j = \gamma \cdot p_f^j + (1 - \gamma) \cdot CollSynCoa(ag_j) \qquad 0 \leq \gamma \leq 1 \tag{9}$$

Thus, because a worker may not finish its subtask on time, we assess the delivery time for a worker as:

$$\tilde{d} = (1 + \delta) \cdot d \tag{10}$$

where $0 \leq \delta \leq 1$ is a parameter that models an increase of the extra time a worker needs to finish if it fails to deliver on time.

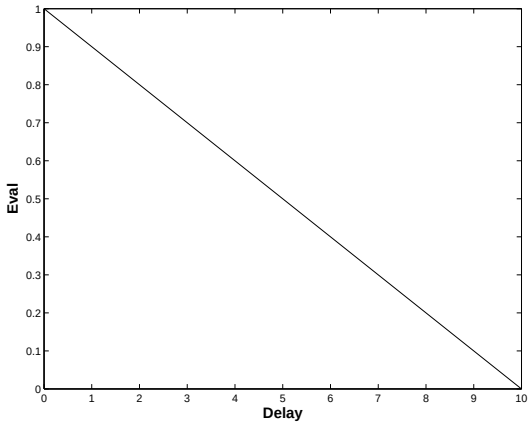


Figure 6: Decay function.

Moreover, in order to evaluate both agents and coalitions, we must instantiate the decay functions $CoalitionEval$ and $ContractorEval$. In this case, we assume that both coalitions and the contractor use the same decay function $EvalC$. We have chosen a linear decrease of reputation with delay (Figure 6), which is specified in Equation 11. Alternative decay functions are possible, but we have chosen this one for the sake of simplicity. Recall that the definition of a complex reputation mechanism is out of the scope of this paper.

$$EvalC(d, \tilde{d}) = 1 - (\tilde{d} - d) \tag{11}$$

where $(\tilde{d} - d)$ is the delay in the task and $\tilde{d} \geq d$.

Finally, in Table 2 we set the parameters from Equations 1, 2, 3, 8, 9, and 10. Notice that $rwd_m/rwd_w = 2$, since being a mediator has a greater reward than being a worker. Finally, we present our results with respect to a reference value, which is the best value that can be obtained when considering the maximum probability of finishing on time in each scenario.

Parameter	Value
γ	0.8
$\alpha_1 = \alpha_2$	0.7
β	0.5
δ	0.1
q	2
rwd_m/rwd_w	2

Table 2: Parameters.

6.2. Resilience analysis

In this section, we have three main goals. First, we study the resilience of coalitions in relation to the reliability of workers and the choice of the reputation mechanisms. Second, we study the capability of coalitions to distinguish unreliable workers. Third, we study how the collaborative synergy affects the percentage of serviced tasks on time. Notice that the task workload models the number of incoming tasks. For the sake of simplicity, we consider two binary selections for reliability: an agent j is a reliable worker if its probability of finishing a task is $p_f^j = 0.9$, while for unreliable workers it is $p_f^j = 0.1$. When we refer to the percentage of reliable workers, we mean the percentage of those agents from the overall agent population; i.e., workers of all coalitions.

6.2.1. Resilience of coalitions depending on workers reliability

In Figure 7, we present how the percentage of tasks serviced on time varies when we vary the percentage of reliable workers. In the figure, we are only interested in the results with *full reputation*, which is when both the reputation of individuals and groups is used. We observe that: (i) the percentage of tasks serviced grows as reliable workers grow; (ii) a low percentage of reliable workers ($\sim$40%) is enough to achieve more than 80% of tasks serviced on time; and (iii) when more than 50% of the workers are reliable, more than 90% of tasks are serviced on time. Therefore, we conclude that our decision making mechanism helps coalitions to achieve very high resilience: the percentage of tasks serviced on time is high despite the high percentage of unreliable workers.

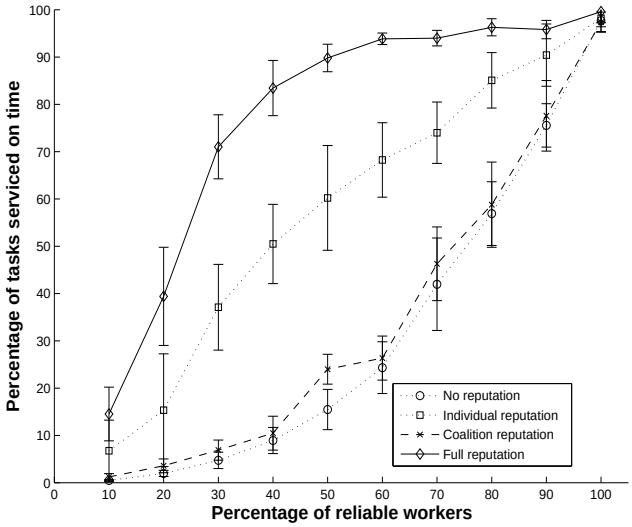


Figure 7: Percentage of tasks serviced on time, varying the percentage of reliable workers.

6.2.2. Resilience of coalitions depending on reputation mechanism

In addition, in Figure 7 we also compare the resilience that results from using individual and coalition reputation (*full reputation*), with respect to: *no reputation*, which does not use reputation for either individual selection or for group selection, but where subtasks are randomly assigned to agents; *individual reputation*, where only agents have reputation, so mediators form groups using it; and *coalition reputation*, where we only use reputation for the coalition, but not for the agents. As expected, we observe that the results when using no reputation are very poor when the percentage of reliable workers is not very high. When coalition reputation is used, the results are only improved by approximately 4%. This is because, if we use coalition reputation without individual reputation, coalitions are not formed by choosing the best workers. Thus, even if a coalition had a good reputation in the past, it can perform badly in the future. When only individual reputation is used, the percentage of tasks serviced on time increases by 30% when half of the agents are reliable, since most agents with higher reputation, agents are chosen. Finally, we observe that adding coalition reputation to individual reputation (full reputation) indeed improves the percentage of tasks serviced on time when compared to all previous reputation mechanisms. We observe that when half of the agents are reliable, using full reputation leads to 30% more tasks being serviced on time than only using individual reputation, and 70% more when compared with the other two approaches. This is because in competitive environments, there is a need to assess not only the most reliable agents, but also the most reliable coalitions to assign a task.

6.2.3. Discriminating unreliable workers

Now, we aim at understanding whether coalitions using our full reputation mechanism are able to discriminate between reliable and unreliable workers. We set 50% of all the agents in the population to be reliable ($p_f^j = 0.9$), and the remaining 50% agents to be unreliable ($p_f^j = 0.1$). Moreover, we set the incoming task load so as to keep all the reliable workers busy. Figure 8 shows the evolution of the percentage of reliable and unreliable workers that are busy. We observe that unreliable workers are promptly distinguished. Conversely, reliable workers are busy most of the time, because since they fail less, they obtain better reputation, and hence they are chosen first to service subtasks.

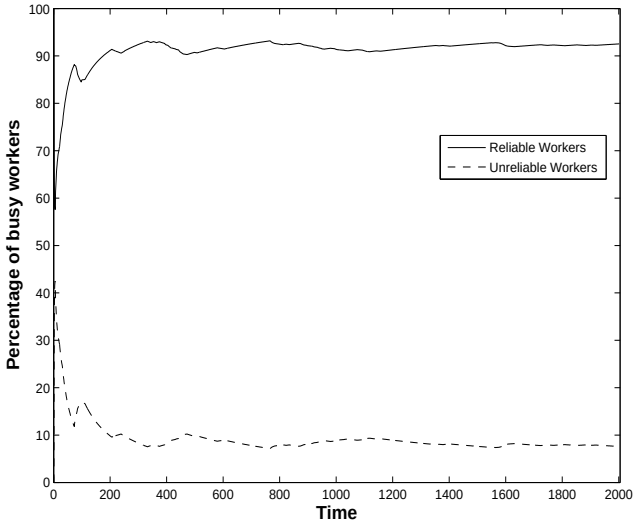


Figure 8: Discrimination of unreliable workers.

6.2.4. Collaborative synergy

In Figure 9 we show how the percentage of tasks serviced on time varies depending on the collaborative synergy. Here, instead of allowing each agent to calculate its own collaborative synergy, we fix it for all agents. We set all agents to $p_f^j = 0.9$, and $\gamma = 0.5$, in order to give a greater weight to collaborative synergy.

As can be seen, the percentage of serviced tasks increases exponentially as soon as we raise the collaborative synergy, and the high influence of this parameter is very visible in the figure. In this case, as the weight of the synergy is high, the percentage of serviced tasks does not increase until this value is approximately 0.6.

6.3. Adaptiveness analysis

In this section we analyze the adaptive capabilities of our decision mechanism. Recall from Section 5 that our adaptive decision making mechanism is aimed at: (i)

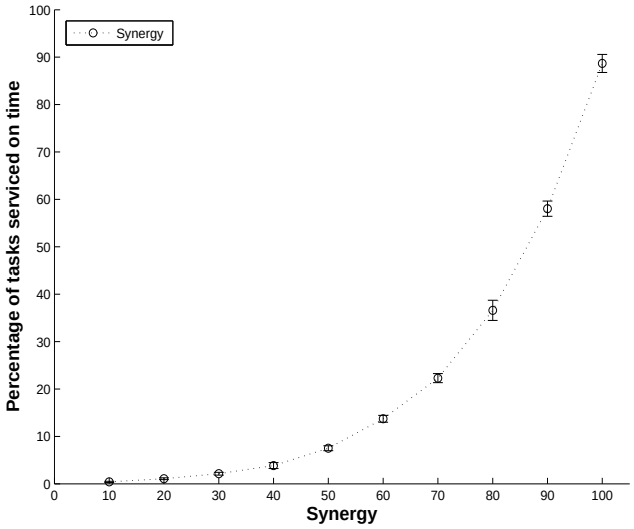


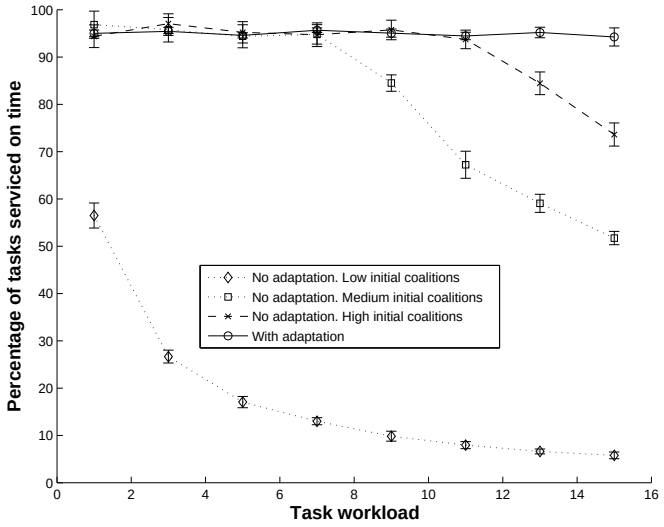
Figure 9: Serviced tasks on time increase exponentially when increasing collaborative synergy.

allowing coalitions to disband when they are no longer competitive; and (ii) allowing agents to start new coalitions. We show that such features lead to an adaptation of the number and composition of coalitions while maintaining a high percentage of tasks serviced on time. Moreover, this occurs in a dynamic environment where the task arrivals and the workload required by each task may vary. Thus, in Section 6.3.1 we consider how coalitions evolve according to the initial number of coalitions and different task workloads. In addition, in Section 6.3.2, we analyze how coalitions adapt when the task workload varies during a simulation.

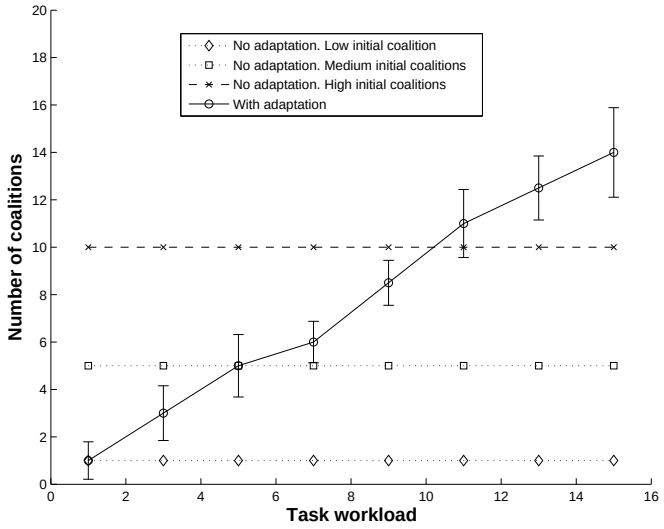
6.3.1. Adaptation to dynamic distributions of tasks

Here, we investigate how coalitions adapt when varying: (i) the initial number of coalitions; and (ii) the customers' incoming task workload. Figure 10a compares the percentage of tasks serviced on time when using our adaptive mechanism (*with adaptation* in the figure) with respect to not using it (*no adaptation*). Not adapting means that new coalitions cannot be formed nor can coalitions be disbanded, so the number of coalitions remains unaltered. Regarding the non-adaptive mechanism, we present the evolution for three different initial conditions (low: 1 coalition; medium: 5 coalitions; high: 10 coalitions). Regarding the adaptive mechanism, we present the percentage of tasks serviced on time when our adaptive mechanism starts with a low number of coalitions (1 coalition). Note that, while we also ran experiments with different initial numbers of coalitions, there are no significant differences, since our mechanism allows coalitions to adapt to the needs of the environment.

When agents use our adaptive mechanism, the percentage of tasks serviced on time remains stable even when varying the task workload. In fact, we see that 95% of tasks are serviced on time, regardless of the incoming task workload. This is because our



(a) Percentage of tasks serviced on time.



(b) Coalitions depending on the task workload.

Figure 10: Comparison without adaptation (*no adaptation*) with our adaptive mechanism (*with adaptation*).

mechanism leads to adaption of the number and composition of coalitions to different task workloads. In fact, in Figure 10b we show that as we increase the task workload, the number of coalitions also increases in order to be able to service all incoming tasks. Thus if we start with one coalition, we observe that when the task workload reaches 8, the number of coalitions increases to 8. In contrast, starting with a high number

(10 coalitions), and a task load of 5, the number of coalitions decreases to 5. Thus our adaptive mechanism allows: (i) the less competitive coalitions to disband when the incoming load is not sufficiently high, so that there are no unused coalitions; and (ii) any agent to start a new coalition when it considers it to be beneficial, so that tasks are not unserviced as a result of there being no available coalitions.

Finally, we see that, as expected, without the adaptive mechanism, as the task workload increases the percentage of tasks serviced on time decreases. As the number of coalitions is fixed, when coalitions reach their capacity, they cannot accept new incoming tasks, so these tasks are not serviced. Note that as we increase the number of initial coalitions, more tasks are serviced with the same incoming load. However, if the incoming task load is increased, then the behavior is similar to the results already shown. Furthermore, we have calculated that if we increase load to more than approximately 30, then the percentage of tasks serviced on time degrades, regardless of whether it uses our adaptive mechanism, because there are insufficient agents to service tasks.

6.3.2. Adaptation to dynamic changes

Finally, we analyze whether the decision making of coalitions and agents allows them to adapt when the incoming task workload changes over time while maintaining a high percentage of tasks serviced on time. Figure 11 presents a comparison between the results obtained with (*adaptive*) and without our adaptive mechanism (*non-adaptive*). For this experiment, we set the task workload to 7 ($L = 7$). Regarding adaptation, we set the initial number of coalitions so that all task workloads can be serviced. Here, every 500 units of time we changed the task workload, to observe its effects on the percentage of tasks serviced on time. From time step 0 to time step 500 we used a load L. Then, the load changed as follows: (1) double workload ($L \rightarrow 2L$); (2) half workload ($2L \rightarrow L$); (3) triple workload ($L \rightarrow 3L$); (4) reset workload ($3L \rightarrow L$).

We observe that when we use our adaptive mechanism: (i) the percentage of serviced tasks on time remains constant and $\sim 95\%$, independently of the task workload; and (ii) the results are independent of the initial L that we chose ($L < 30$).

Without adaptation, coalitions cannot be disbanded and agents cannot start coalitions, so the number of coalitions remains fixed. Thus, when the incoming workload is higher than L, this causes the percentage of tasks serviced on time to decrease. Moreover, having a fixed number of coalitions, even from time step 0 to time step 500, also causes the percentage of tasks serviced on time to be lower than with adaptation. This is because if a coalition has a delay, no other coalition is formed, and there is no available coalition to service tasks. Finally, after load changes, even when we set the load to L again, the percentage of tasks serviced on time does not recover, since it has degraded.

7. Conclusions

In this paper, we have focused on building a dynamic coalition formation mechanism that could be employed in collaborative scenarios like crowdsourcing, co-working, etc., where complex tasks are performed by groups of working humans, modeled as agents in this paper. Thus with the goal of improving the quality and quantity of completed tasks, in the context of a realistic scenario, we have introduced a novel decision-making mechanism that allows agents in a competitive environment to autonomously

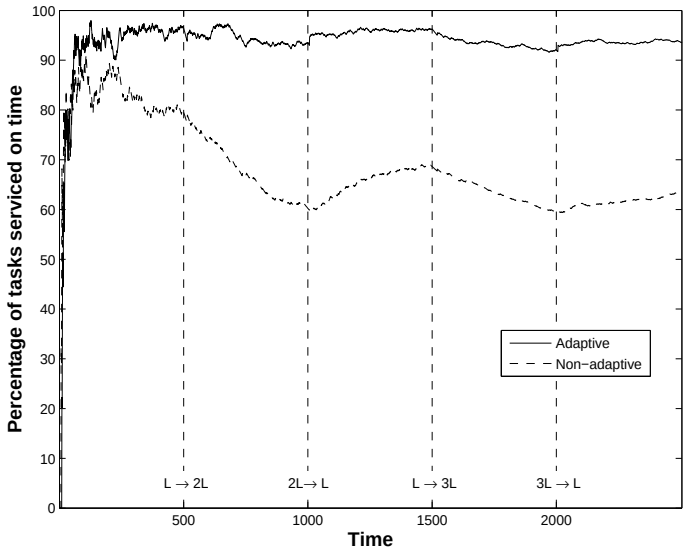


Figure 11: Percentage of tasks serviced on time over time. Adaptive vs. non-adaptive.

enact and sustain coalitions. First, our mechanism allows a coalition: (i) to assemble the most reliable team of agents to service a certain task based on agent reputation; and (ii) to decide whether the coalition must be sustained or disbanded because it is not longer beneficial. Second, our mechanism allows agents to decide whether to continue to be part of a coalition, or instead to join another coalition. In this approach, the reputation mechanisms of agents and coalitions are a fundamental aspect of evaluating individual and group quality, in order to recruit new members or to assign new tasks. In evaluating our model and mechanisms, we have provided a set of quantitative simulation results to show that the model of crowdsourcing envisioned by Kittur [16] delivers major benefits. In fact, our study empirically reinforces Kittur's model and proposals.

More specifically, we have provided empirical evidence showing that when agents employ our decision mechanism it is possible for them to maintain high levels of customer satisfaction (in terms of percentage of tasks serviced on time). First, we showed that coalitions exhibit high resilience: the percentage of tasks serviced on time is high despite a high percentage of unreliable workers. Even when the percentage of reliable agents is low ($\sim 40\%$), the percentage of serviced tasks on time is beyond 80%. Coalitions achieve high resilience through the use of a reputation mechanism that facilitates ratings about individual workers and coalitions as a whole. This mechanism helps coalitions to quickly discriminate between good and bad workers. Second, we showed that coalitions and agents successfully adapt to a varying distribution of customers' incoming tasks, both regarding the arrival rate and their characteristics. Thus, we observe that $\sim 95\%$ of tasks are serviced on time despite significant variations in the incoming distribution of tasks. This occurs because our decision-making mechanism enables: (i) coalitions to disband when they become non-competitive (particularly

in scenarios with a low demand of tasks); and (ii) individual workers detect opportunities to start a new coalition (particularly in scenarios with a high demand of tasks). Thus, we have empirically seen that it is worth engineering group formation facilities in crowdsourcing sites, using reputation in order to improve the overall performance.

As future work we plan to investigate further reputation mechanisms that take into account not only delivery time, but also further task achieving dimensions. We also plan to combine cost and reputation to create utility functions able to better discriminate among coalitions. Similarly, we plan to study the effects of allowing agents to belong to different coalitions at the same time. Although the main goal of this paper is to improve the quality and the quantity of completed tasks in order to maintain high levels of customer satisfaction, a study of agent gains and behavior is also relevant and interesting, and we will undertake such an analysis in future work. In addition, we plan to extend the model by allowing an agent to adopt more than two roles. Finally, we plan to perform experiments involving humans, to validate our simulations, in order that our approach may generate recommendations to humans regarding whether to join a coalition, abandon a coalition, change a coalition, etc., in order to assemble the most reliable teams of humans to maintain high levels of customer satisfaction.

Acknowledgments

The first author thanks the grant Formación de Profesorado Universitario (FPU), reference AP2010-1742. Arcos and Rodriguez-Aguilar thank projects COR (TIN2012-38876-C02-01/02) and Generalitat of Catalunya (2014 SGR-118). Work supported by the European Regional Development Fund (ERDF) and the Galician Regional Government under agreement for funding the Atlantic Research Center for Information and Communication Technologies (AtlantTIC).

References

[1] Abdallah, S. and Lesser, V. (2004). Organization-based cooperative coalition formation. In *Intelligent Agent Technology, 2004. (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, pages 162 – 168.

[2] Afsarmanesh, H. and Camarinha-Matos, L. M. (2005). A framework for management of virtual organization breeding environments. In *In Proceedings of IMP group Conference*, pages 35–48. Springer.

[3] Afsarmanesh, H., Camarinha-Matos, L. M., and Msanjila, S. S. (2009). On management of 2nd generation virtual organizations breeding environments. *Annual Reviews in Control*, 33(2):209–219.

[4] Aknine, S., Pinson, S., and Shakun, M. (2004). A Multi-Agent coalition formation method based on preference models. *Group Decision and Negotiation*, 13(6):513–538.

[5] Amgoud, L. (2005). Towards a formal model for task allocation via coalition formation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, AAMAS '05, pages 1185–1186, New York, NY, USA. ACM.

[6] Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., and Leonardi, S. (2012). Online team formation in social networks. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 839–848, New York, NY, USA. ACM.

[7] Anagnostopoulos, A., Castillo, C., Gionis, A., Becchetti, L., and Leonardi, S. (2010). Power in unity: Forming teams in Large-Scale community systems. In *Proc. of Conference on Information and Knowledge Management (CIKM)*, pages 599–608. ACM Press.

[8] Chen, X., Lin, Q., and Zhou, D. (2013). Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 64–72. JMLR Workshop and Conference Proceedings.

[9] Decker, K., Sycara, K., and Williamson, M. (1997). Middle-Agents for the internet. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*.

[10] Ho, C.-J. and Vaughan, J. W. (2012). Online task assignment in crowdsourcing markets. In *AAAI*.

[11] Ho, C.-J., Zhang, Y., Vaughan, J., and van der Schaar, M. (2012). Towards social norm design for crowdsourcing markets. In *AAAI Workshops*.

[12] Ipeirotis, P., Provost, F., Sheng, V., and Wang, J. (2013). Repeated labeling using multiple noisy labelers. *Data Mining and Knowledge Discovery*, pages 1–40.

[13] Ipeirotis, P. G. (2010). Analyzing the amazon mechanical turk marketplace. *XRDS*, 17(2):16–21.

[14] Karger, D. R., Oh, S., and Shah, D. (2011a). Budget-optimal task allocation for reliable crowdsourcing systems. *CoRR*, abs/1110.3564.

[15] Karger, D. R., Oh, S., and Shah, D. (2011b). Iterative learning for reliable crowdsourcing systems. *Advances in Neural Information Processing Systems 24*, pages 1953–1961.

[16] Kittur, A., Nickerson, J. V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., and Horton, J. (2013). The future of crowd work. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1301–1318, New York, NY, USA. ACM.

[17] Klusch, M. and Gerber, A. (2002). Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, 17(3):42–47.

[18] Kraus, S., Shehory, O., and Taase, G. (2003). Coalition formation with uncertain heterogeneous information. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS '03, pages 1–8, New York, NY, USA. ACM.

[19] Lappas, T., Liu, K., and Terzi, E. (2009). Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 467–476, New York, NY, USA. ACM.

[20] Lau, H. C. and Zhang, L. (2003). Task allocation via multi-agent coalition formation: taxonomy, algorithms and complexity. In *15th IEEE International Conference on Tools with Artificial Intelligence, 2003.*, pages 346 – 350.

[21] Maturana, F., Shen, W., and Norrie, D. H. (1999). Metamorph: An adaptive agent-based architecture for intelligent manufacturing. *International Journal of Production Research*, 37:2159–2174.

[22] Mérida-Campos, C. and Willmott, S. (2004). Modelling coalition formation over time for iterative coalition games. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '04, pages 572–579, Washington, DC, USA. IEEE Computer Society.

[23] Mérida-Campos, C. and Willmott, S. (2006a). Agent compatibility and coalition formation: Investigating two interacting negotiation strategies. In Fasli, M. and Shehory, O., editors, *TADA/AMEC*, volume 4452 of *Lecture Notes in Computer Science*, pages 75–89. Springer.

[24] Mérida-Campos, C. and Willmott, S. (2006b). The effect of heterogeneity on coalition formation in iterated request for proposal scenarios. In Dunin-Keplicz, B., Omicini, A., and Padget, J. A., editors, *EUMAS*, volume 223 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[oDesk] oDesk. https://www.odesk.com/.

[26] Sandholm, T., Larson, K., Andersson, M., Shehory, O., and Tohmé, F. (1999). Coalition structure generation with worst case guarantees. *Artif. Intell.*, 111(1-2):209–238.

[27] Shehory, O. and Kraus, S. (1995). Task allocation via coalition formation among autonomous agents. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, pages 655–661, San Francisco, CA, USA.

[28] Shehory, O. and Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1):165–200.

[29] Shehory, O. and Kraus, S. (1999). Feasible formation of coalitions among autonomous agents in non-super-additive environments. *Computational Intelligence*, 15(3).

[30] Slivkins, A. and Vaughan, J. W. (2013). Online decision making in crowdsourcing markets: Theoretical challenges (position paper). *CoRR*, abs/1308.1746, abs/1308.1746.

[31] Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.*, 29(12):1104–1113.

[32] Soh, L.-K. and Li, X. (2003). An integrated multilevel learning approach to multiagent coalition formation. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 619–624, San Francisco, CA, USA.

[33] Sycara, K. P. and Vaculín, R. (2008). Process mediation, execution monitoring and recovery for semantic web services. *IEEE Data Engineering Bulletin*, 31(3):13–17.

[34] The Economist (2011). The rise of co-working. http://www.economist.com/node/21542190?fsrc=scn/fb/wl/ar/anotheralternativetotheoffice.

[35] Ye, D., Zhang, M., and Sutanto, D. (2012). Integrating self-organisation into dynamic coalition formation. In van der Hoek, W., Padgham, L., Conitzer, V., and Winikoff, M., editors, *AAMAS*, pages 1253–1254. IFAAMAS.

[36] Zheng, X. and Koenig, S. (2008). Greedy approaches for solving task-allocation problems with coalitions. In *AAMAS 2008 Workshop on Formal Models and Methods for Multi-Robot Systems*.