Finding Better Teammates in a Semi-cooperative Multi-agent System

Sara Amini
Department of Computer Science and Information Technology
Institute for Advanced Studies in Basic Sciences
Zanjan, Iran 45137-66731
Email: s-amini@iasbs.ac.ir

Mohsen Afsharchi
Department of Electrical and Computer Engineering
University of Zanjan
Zanjan, Iran 45371-38791
Email: afsharchim@znu.ac.ir

***Abstract*—Although in semi-cooperative systems, agents are self-interested, they have to help others to get their help in the requirement time. However, designing a distributed method that encourages agents to be truthful and cooperative is challenging. If each agent is able to find useful co-workers to team up with, they are inspired to cooperate with each other, which leads to desirable results for both individuals and the system as a whole.**

In this paper, a distributed mechanism on the basis of reinforcement learning (RL) is proposed which guides agents to find better teammates i.e. agents which are cooperative and useful for a long run. A model-based RL is used to model agents' beliefs toward others as transition probabilities where agents try to influence these probabilities in a way they get benefit from. We clarify properties of our system such as Nash equilibrium by some theorems and test the mechanism by applying it to a distributed sensor network designed for target tracking. The simulation results show effectiveness of the method since RL agents gain more in comparison to selfish and random-policy agents. Experiments also indicate that mixed-strategy RL agents benefit more by taking advantage of further synergy produced by forming larger teams.

I. Introduction

In a multi-agent system (MAS) with selfish agents, encouraging agents to be truthful and cooperative is not a simple task. Rational agents by definition have one goal; maximizing their own utilities, and in many cases they do not tend to help others to avoid paying any costs. However, in some situations, agents cannot carry out their desired tasks alone as doing some tasks requires more resources than an agent can supply by itself. In these systems (called semi-cooperative) the performance of each agent is highly dependent on the performance of others and the system as a whole [1]. Thus, a method is required to inspire agents to be truthful and help others.

The first idea might be a method in which each agent gives scores to its co-workers, and a central unit controls resources of the system [2]. A strategy-proof mechanism makes sure agents behave truthfully [3], but there is still a problem with this plan due to existence of the main controller which decreases scalability of the system. Reducing the central controller's role in these systems is our main motivation in this work, and to do that, we let agents to negotiate for resources, and to exhort them to cooperate with each other, we give them a guideline which is reinforcement learning (RL).

In real world systems, requesting help from an agent may be rejected since the requested agent being busy at the requirement time. Learning helps agents to monitor others' behaviors in a long run process and prevent them to be affected by unintentional oscillations of their co-workers' behaviors. Learning also has the merit of restraining agents to be influenced by intentional instant changes of selfish agents' behaviors which want to deceive cooperative agents and persuade them to trust them by offering a good contract in the need time while having a history of untruthful behavior. By RL, agents try to find better teammates in the system; i.e. agents which can be useful, not only for a short period of time, but also for a long time. This means they seem to have high quality resources and also be loyal and cooperative.

A problem in distributed sensing (managing sensors spatially distributed in an area) is that each sensor has a local view of the environment and their obtained data, sometimes are incompatible. Tasks of the system can be done more accurate if sensors cooperate with each other to fuse their data [4].

If each sensor possessed by a different owner, the situation becomes more complex since individual sensors might have intentions to satisfy just their own stakeholder and use others' resources, but be unwilling to give theirs to others to pay as less cost as possible. In addition, sensors' goals might be in conflict with each other [2].

Distributed sensor networks (DSNs) with different stakeholders have applications in Pico-satellite projects, multinational multiplatform military exercises, and traffic control [5].

The goals of target tracking systems are detecting moving objects in an area and determining their features like their path [6]. To test our method in practice, a DSN (with different owners) for target tracking is implemented in our work.

The rest of this paper is organized as follows; in the next section, a short review of the previous related works is presented. In Section III the problem is formally defined. In Section IV the system model is explained. System properties and experimental results are presented in the next two sections. Finally, Section VII concludes this paper and gives some directions for future researches.

II. Related Work

Team or coalition formation has a key role in MASs since it is highly related to distributed problem solving which is a main goal in designing a MAS. One famous application of distributed problem solving is distributed task (or resource) allocation among both cooperative and self-interested agents. In this section, we review some of these works.

Gaston and desJardins introduce a distributed on-line learning mechanism called agent-organized network (AON) as a structure-based dynamic team formation tool for cooperative agents [7]. The researchers in [8] propose a hierarchical model by using Q-learning in a cooperative context. Ahn, et al [9] present a multi-dimensional trust (MDT) model in an environment with self-interested agents. Agents use MDT to recognize beneficial teammates by employing a weighted sum of MDT components, and in order to learn these weights, Q-learning is used.

Pippin and Christensen [10] use some auctions for task allocation in a multi-agent system. They employ an observation-based trust model and a shared reputation mechanism to determine appropriate agents for auctions. The researchers use a Beta distribution to build their trust model. Although [9], [10] have some similarities to our work, there are some big differences too. In both [9] and [10], agents learn about other agents' behaviors via some probabilities but they never try to affect these probabilities. In contrast, in our work, agents recognize useful co-workers and they attempt to make good beliefs in these agents. The beliefs are modeled as transition probabilities of a model-based RL (Certainty Equivalence [11]) and agents consciously try to increase probability of forming teams with beneficial co-workers.

Learning to form coalitions has been subject of many researches in the DSNs area, especially by MASs scientists. Generally, these projects can be divided into two categories: networks with sensors owned by just one stakeholder, and those with different owners. We start by the first type.

Glinton et al [12] use a graph to model coalition structure of the network and try to improve quality of coalitions by modifying this graph. In [13] researchers use a hierarchical structure to have a self-organized DSN. The authors of [14] define some features (like P_1, ..., P_6 in our work) that by using a weighted sum of them, agents determine suitable co-workers. Then agents start negotiations with these candidates and on the basis of obtained results, update the weights (a form of a model-free RL). The authors use this approach in a cooperative context and they do not need to consider the effect of selecting one agent instead of another. However, in our work with self-interested agents, sensors need to be more careful about the effect of these selections and that is why we use a model-based RL as mentioned before. In [15], the authors present a self-organizing resource allocation (SORA) method in a DSN which defines a virtual market in the network. The work in [16] presents an agent-based model (with two types of agents; sensor-agents and task-agents which negotiate with each other) to manage resources of a DSN.

In this paragraph we summarize some of the works about DSNs with selfish agents. A centralized mechanism on the basis of VCG (where agents value their obtained information by Fisher information) is used in [2]. Dash uses a second-price sealed bid auction for resource allocation in a DSN [17]. In his work, the sensors which want to give information to others assumed to be sellers and the demanding sensors are buyers. Each seller acts as an auctioneer and auctions are executed simultaneously. We use negotiations rather than simultaneous auctions in our work because they can be performed independently whenever a demanding sensor needs and by learning trusted agents, it can be changed to a simple message passing, so agents waste less time in this highly dynamic environment. The work in [18] presents a centralized trust-based mechanism on the basis of VCG which is efficient, individually rational and incentive compatible.

III. Problem Definition

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$ and $\mathcal{T} = \{t_1, t_2, \ldots, t_M\}$ be the set of agents and tasks respectively, and assume each agent is owned by a different stakeholder.

Each agent in every moment is in one of these two cases: Busy or Not-busy. In Busy, they are using their resources for doing their own tasks or helping others to perform their desired tasks, so they incur a cost. They cannot accept any help request as they cannot afford it due to their limited resources and also because this behavior gives some agents opportunities for malicious behaviors (getting resources of the system and never trying to make up for that). Offering a systematic method which records truthful behaviors (without a central controller) leads to desired results for the system.

There are several factors to be considered in finding an appropriate teammate; the first is the quality of doing a task t_i by some agent a_j, as different agents might have different capabilities. The second is the cost of persuading some agent to do a task, and the last but not least is the belief of an agent in how much the potential helper really is apt to help; If there exists another demanding agent a_k which offers a better contract, does a_j still like to free up its resources for a_i? Agents are given RL as a guideline to assist them making the best decisions in this complicated environment.

IV. System Model

In this section, we present our RL model and the way agents can use its results to learn which agent they must help in the system and expect to get help from to maximize their gains.

A. Reinforcement Learning Modeling

Equation 1, called Bellman equation [19], is the base of RL where, $V(s)$ is the value (long-term usefulness) of state s, $R(s)$ is the immediate reward of s, S and A are the set of states and actions respectively, γ $(0 \leq \gamma < 1)$ is the discount factor, and $P_{sa}(s')$ is the transition model implying the probability of reaching state s' by taking action a in s.

$$\forall s \in S: \quad V(s) = R(s) + \gamma \max_{a \in A} \sum_{s' \in S} P_{sa}(s') V(s') \qquad (1)$$

Neighbors of a_i are agents which are known by it and are represented by $N(a_i)$ in our work. Assume agent a_i requires k resources to do a task t_j which is equivalent to needing a k-member team. If $|N(a_i)| = n$, a_i should seek the best team in a $\binom{n}{k}$-member search space. With this kind of modeling,

each RL state would be one of the teams a_i can join. To get rid of this big search space, we model each RL state as an a_i's neighbor, but assume a_i chooses k states rather than one[1].

In a semi-cooperative environment, agents have to help others because they need to get help from them. They try to find, help and also get help from some agents which seem truthful and cooperative: if agents help them, they can expect the help receivers compensate in the future. Agents cannot just rely on the immediate offered utility when deciding to help others or needing to get help, since this gives selfish agents the opportunity of deceiving them and devouring resources of the system. If the network is able to record cooperative behaviors in a systematic way, agents can trust some limited set of agents, help them and expect to get help from in the need time.

In order to select the best team candidates, agents should consider some factors. First, they need a way to measure the quality of performing a task by some agent which is a context-specific quantity and for instance in a DSN, Fisher information [2] is a good choice to determine how much a sensor's information is valuable. In our work, the quality provided by a_j's resources to a_i is shown by $Q(a_i, a_j)$.

The second is the cost of doing a task which can be seen as the exchanged money in the negotiation, but its functionality can be broadened. Credit is defined as a measurement in the negotiation phase (before team formation) which is a more general term than cost or money, as we do not necessarily need agents transfer money in the system. Credit is a quantity that agents can save gradually to understand how much others value their help; the more valuable they seem to an agent, the better to team up with it. We use $C(a_i, a_j)$ as a notation for amount of credit that a_j gives to a_i to get its resources in return.

The third factor is the probability of making a team with some agent. Due to uncertainty in the environment, agents try to learn others' behavior patterns via some probabilities. These probabilities are the probability of coming to an agreement to get help from some agent such as a_j, shown by $P_Q(a_j)$, and the probability of persuading a_j to give some amounts of credit (in exchange for the hired resources), shown by $P_C(a_j)$. Agents want to recognize useful teammates in the system, and try to increase P_Q and P_C of them. They attempt to estimate P_Q and P_C and also the future values of these quantities (after accepting or rejecting the current offers). By using Bellman equation, agents try to keep a balance between these factors (credit, quality and probabilities P_Q and P_C). In RL literature, we ought to distinguish between the long-term desirability of of each state and its immediate goodness. Let $V_Q(a_j)$ and $V_C(a_j)$ denote the (long-term) value of agent a_j from quality and credit perspective when some agent decides to form a team with it (i.e. in the future, how much it is worth teaming up with a_j from quality and credit points of view).

[1]If $k \ll n$, finding the best k elements of an n-member state-space has time complexity $O(n)$ which is better than finding the best of a state-space with teams as states which needs $O(\binom{n}{k})$. If k approaches n but still $k < n$, in the worst case we have to sort an n-member array which needs $O(n \log n)$ and this is again better than $O(\binom{n}{k})$. The only case $O(\binom{n}{k})$ surpasses $O(n \log n)$ is when $k = n$, but in this case agents do not need to learn anything! If $k = 1$; $O(\binom{n}{k}) = O(n)$, so both models result the same complexity.

In order to model agents' beliefs as probability distributions, a model-based RL is needed, in which two types of parameters must be learned: transition probabilities and state rewards. Certainty Equivalence [11] is a simple model-based RL where agents save the results of taking different actions at different states and by using them, they gradually approximate transition probabilities and state rewards. In our work, transition probabilities are P_C and P_Q (assume we want to solve two RL problems in parallel, one for quality and the other for credit), and rewards of a state are C and Q (depending on agents want to find a state for getting credit or quality from).

At first, imagine a_i needs to find the best agent (state in RL literature) to get help from. By applying Bellman equation to a_i's neighbors, $N(a_i)$, we define

$$V_Q(a_j) := P_1(a_j)\overline{Q}(a_j) + \gamma \max_{a_k \in N(a_i)} P_Q^f(a_k|a_j)V_Q(a_k), \quad (2)$$

where $P_1(a_j)$ is the probability of getting a_j's resources in the negotiation, $\overline{Q}(a_j)$ is the average amount of quality a_j can provide, and $P_Q^f(a_k|a_j)$ equals $P_Q(a_k)$ in the next round, while choosing a_j in the current one ($P_Q^f(a_k|a_j)$ shows the belief of a_k after forming a team with a_j).

By using $\text{Neg}_r(a_i, a_j)$ as the number of negotiations a_i opened with a_j, and $\text{Neg}_s(a_i, a_j)$ as the number of successful negotiations a_i conducted with a_j [14], we have

$$P_1(a_j) := \frac{\text{Neg}_s(a_i, a_j)}{\text{Neg}_r(a_i, a_j)}.$$

$P_1(a_j)$ can be seen as an approximation of $P_Q(a_j)$, but we can use some parameters similar to features of [14] as a more precise approximation. $P_1(a_j)$ shows the percentage of times negotiations with a_j (to get its resources) were successful. It does not imply these successful negotiations make what percentage of all successful negotiations a_i ever experienced. For example if a_i asks for a_j's and a_k's resources once and ten times respectively, and they both agree to give their resources in all of these times, $P_1(a_j)$ and $P_1(a_k)$ are equal, even though obviously, a_k is more reliable than a_j. By modifying features of [14] to be more appropriate to our semi-cooperative context, we use $\text{Neg}_{tr}(a_i)$ as the total number of times a_i opened a negotiation with its neighbors, and $\text{Neg}_{ts}(a_i)$ as the total number of times a_i conducted a successful negotiation with its neighbors, and by applying them a_i calculates

$$P_2(a_j) := \frac{\text{Neg}_s(a_i, a_j)}{\text{Neg}_{tr}(a_i)},$$

$$P_3(a_j) := \frac{\text{Neg}_s(a_i, a_j)}{\text{Neg}_{ts}(a_i)},$$

$$P_Q(a_j) := w_1 P_1(a_j) + w_2 P_2(a_j) + w_3 P_3(a_j),$$

where w_is are normalizing factors.

$P_Q^f(a_k|a_j)$ is the probability of getting a_k's resources after teaming up with a_j. Sometimes it may be worth ignoring some utilities to make a good belief in a more valuable agent.

Remark 1: The reason why in Equation 2 a sigma (in the second term) is dropped (compare 1 to 2) is that the summation in Equation 1 is done on all states an agent can reach by taking action a. In our context, taking an action means deciding to open a negotiation with an agent which may lead to get some resources (if the negotiation finishes by success) or failure. Clearly, the probability of a successful negotiation is complement to the probability of a failed one. If a_i experiences a failed negotiation it gets nothing, thus

$$V_Q(a_j) = P_1(a_j)\overline{Q}(a_j)$$

$$+\gamma \max_{a_k \in N(a_i)} [P_Q^f(a_k|a_j)V_Q(a_k) + (1 - P_Q^f(a_k|a_j)) \times 0]$$
$$= P_1(a_j)\overline{Q}(a_j) + \gamma \max_{a_k \in N(a_i)} P_Q^f(a_k|a_j)V_Q(a_k)$$

The first term of Equation 2 is the immediate reward about which agents do not have any prior knowledge, and they use average obtained reward as an approximation. Because reward is received after negotiation and agents have uncertainty about it, they use mathematical expectation as an estimate:

$$E(Q(a_j)) = P_s(a_j)\overline{Q}(a_j) + P_f(a_j) \times 0 = P_s(a_j)\overline{Q}(a_j)$$

$P_s(a_j)$ is the probability of a successful negotiation with a_j when an agent needs its resource, which equals $P_1(a_j)$ in our work, and $P_f(a_j)$ is the probability of a failed negotiation.

Thus far, we have just explained about quality, but all of the mentioned discussions can be applied to the credit concept too; at first P_4, P_5 and P_6 are defined similar to P_1, P_2 and P_3 respectively:

$$P_4(a_j) := \frac{\text{Neg}_s(a_j, a_i)}{\text{Neg}_r(a_j, a_i)},$$

$$P_5(a_j) := \frac{\text{Neg}_s(a_j, a_i)}{\text{Neg}_{tr}(a_j)},$$

$$P_6(a_j) := \frac{\text{Neg}_s(a_j, a_i)}{\text{Neg}_{ts}(a_j)},$$

and by using them P_C is defined as

$$P_C(a_j) := w_4P_4(a_j) + w_5P_5(a_j) + w_6P_6(a_j),$$

and finally, Bellman equation for credit is defined as

$$V_C(a_j) := P_4(a_j)\overline{C}(a_j) + \gamma \max_{a_k \in N(a_i)} P_C^f(a_k|a_j)V_C(a_k),$$

where its parameters can be interpreted like Equation 2, except that here quality changes to credit.

B. Learning to Find Better Co-workers

So far, a_i has solved two RL problems in parallel; one for credit (to understand which agent it must help) and another for quality (to understand from which agent it must get help). In this section, we explain how a_i can make a relation between them to find better co-workers.

Total value of each of a_i's neighbors (state in RL literature) at first glance may be defined as

$$V(a_j) := w_QV_Q(a_j) + w_CV_C(a_j), \tag{3}$$

where w_Q and w_C are weights agents can learn, but we set $w_Q + w_C = 1$ for convenience. Needing to form a k-member team, a_i's long-term utility can be defined as

$$V(a_i) := \sum_{\substack{a_{i_l} \in N(a_i) \\ l=1,2,3,\ldots,k}} V(a_{i_l}).$$

There is a problem with Equation 3 because selfish agents can only use resources of the system without trying to compensate for that and increase the second term of Equation 3 and consequently V in an unfair way. Although reducing w_C might seem as a solution, adding a term that maintains a balance between these two terms would be a better solution. We do not want to let this new term become biased by credit to prevent untruthful agents benefiting from offering a high amount of credit when needing help, while not helping others effectively. So we define Balance factor as

$$\forall a_j \in N(a_i): \quad B(a_j) := \frac{p_1(a_j) + p_2(a_j) + p_3(a_j)}{p_4(a_j) + p_5(a_j) + p_6(a_j)},$$

and therefore Equation 3 turns out to be

$$V(a_j) := w_QV_Q(a_j) + w_CV_C(a_j) + w_BB(a_j), \tag{4}$$

where $w_Q + w_C + w_B = 1$[2].

the numerator of $B(a_j)$is the probability of a_j helping a_i and its denominator approximates the probability of a_i helping a_j (from a_j's point of view). Thus, B indicates the balance between a_i and a_j; a high B means a_j helped a_i more, and a_i has a commitment to make up for that. Having B near to 1 implies that there is a balance between a_i and a_j, and a very low B shows a_j potentially has a tendency to just use a_i's resources and is disinclined to help when a_i needs.

By using $B(a_j)$, a_i has two purposes. First, if a_j just receives a_i's resources and never tries to compensate, a_i can recognize it even though a_j's V_C and consequently V are high. Second, if a_j has helped it much in the past, but a_i could not make up for that, a_i can recognize it. The reason is that although a_i has a good belief about a_j, a_j does not; it may be the case that a_j is trying to replace a_i with another more helpful agent, and a_i must be prepared for that.

C. Trust Factor

By applying a pure strategy, RL agents try to form k-member teams when needing k resources. Generally, an RL agent should spend some time to explore the environment and after knowing it properly, start to benefit from what has learned (exploitation step). Exploitation in our work means each agent starts to make a k-member team with a fixed set of agents on the basis of RL results. Switching from working with n members to k ones, especially when the difference between these two numbers is huge, can reduce the social welfare severely. To prevent this situation, we let agents to use a concept called Trust factor which is defined as

$$\forall a_j \in N(a_i): \quad T(a_j) := \frac{V_Q(a_j)}{V_C(a_j)}.$$

Agents use T between exploration and exploitation steps to smooth effects of this change in their strategies. In fact, between these two steps, agents can eliminate some agents which have very lower T than their other neighbors. In this way, they omit some (not useful) states from their search spaces and the smaller the search space, the faster RL converges and the cost of performing each RL step decreases, but more important than these two advantages, agents can smooth switching from working with n members to k ones.

D. RL in a Semi-cooperative Sensor Network

In a DSN for target tracking, sensors usually have to team up to estimate path of a target with an acceptable precision.

Assume a_j and a_i are neighbors in a network. They are not aware of each other's status. A truthful behavior means if one is in Not-busy status and the other asks for help, it reveals its status honestly and helps the requester. However, due to a need to spend some costs, they might not tend to tell the truth.

[2]By running our model several times, we understood it is better to set $w_Q > w_C \gg w_B$ at the beginning, but when the system approaches the balance (convergence in RL literature), we gradually let $w_Q = w_C = w_B$. Another point is that, by setting upper and lower bands for C and then scaling Q and B accordingly, we would have the same domains for all of them (C, Q and B), which makes comparison more fair.

We believe RL can be seen as a distributed solution to this problem since by using it, agents (i.e. sensors in our work) learn about others' behaviors gradually and the system is able to store truthful behaviors' information in a distributed manner.

In order to use the proposed RL method, we need an evaluation metric to compute the quality of a team candidates' resources. Quality in our model (Equation 2) is a context-specific concept and in our work (a DSN), we use Fisher information as an evaluation metric [2].

We cannot just use Fisher information as the quality measurement; there must be a balance between Fisher information and the paid credit, and sensors should themselves be able to decide about the value which one may be worth to them. By defining the quality like Equation 5 sensors have flexibility to decide about the value of these parameters.

$$Q(a_j) = w_f I(a_j) + w_c(MC - C(a_j)) \tag{5}$$

In Equation 5, $I(a_j)$ is Fisher information provided by sensor a_j's data for the current target, *MC* is the maximum amount of credit an agent can afford for a contract and $C(a_j)$ is the credit a_j demands to give its resources to the requester. Two factors are important for a demanding sensor: first, value of the supplier's information, which is determined by Fisher information, and second, cost of the supplier's resource that the fewer the better, so appears with a negative sign in Equation 5. Coefficients w_f and w_c show the importance of these two terms. For instance, the demanding sensor might value the first term more than the second for a target which is very important for it and increases w_f in comparison to w_c, and vice versa when the target is less important or when it has spent too much credit in the past. However, $w_f + w_c = 1$ in all situations.

Finally, $0 < I(a_j) < MC$; domain of Fisher information is changed to domain of *MC* to make comparison easier and more fair.

V. System Properties

After running some steps to explore the environment, agents start to choose agents with the highest V to make teams with them. If their team candidates agree to join them, this greedy behavior in selecting k members leads to a pure strategy which changes the belief probability distribution of each agent such as a_i over its neighbors from a uniform distribution

$$P(a_{i_1}) = \frac{1}{n}, P(a_{i_2}) = \frac{1}{n}, P(a_{i_3}) = \frac{1}{n}, \ldots, P(a_{i_n}) = \frac{1}{n}$$

where $a_{i_j} \in N(a_i)$ and $|N(a_i)| = n$, to

$$\begin{aligned} &P(a_{i_1}) = \frac{1}{k} - \epsilon_1, P(a_{i_2}) = \frac{1}{k} - \epsilon_2, ..., P(a_{i_k}) = \frac{1}{k} - \epsilon_k, \\ &P(a_{i_{k+1}}) = \delta_1, \ldots, P(a_{i_n}) = \delta_{n-k} \end{aligned} \tag{6}$$

where $\sum_{l=1}^{k} \epsilon_l = \sum_{m=1}^{n-k} \delta_m$.

If a_i is successful in forming its desired team, it starts to give and get help from a fixed set of agents. So, it begins to increase $P_1, \ldots, P_6$ of this set, and consequently their functions P_Q and P_C. This leads a_i to differentiate between its own team members' P_Qs and P_Cs from others'. This situation happens in probabilities of a_i's teammates too, at least about a_i. Thus, after exploring enough in the environment, a_i is able to find a set which is responsible to satisfy, and also has a right to demand its required resources. If a_i needs k resources, it starts to make a k-member team, so it should satisfy k agents in the system to modify their beliefs in favor of itself to get benefit from their help in the future.

This behavior leads to a Nash equilibrium (NE) as shown in the following theorem.

Theorem 2: When an agent's probability distribution over its neighbors converges to Distribution 6, using a (pure) strategy that selects the first k elements (with the highest P) is a Nash equilibrium.

Proof: If a_i reaches Distribution 6, there are k members which recognize it as a useful element in the system. Thus, if a_i asks help from these members, they agree with probability $1/k$ which is higher than any other belief probability values (Note that a_i needs k probability values to be maximum at the same time and uniform distribution $1/k$ is what it requires). Hence agents by selecting the best k members which have higher probability values get more than selecting at least one element outside of this k-member set. So, by changing its strategy unilaterally, a_i does not get something more, and as a result, this strategy is a Nash equilibrium. ■

Having a system which converges to the NE does not always guarantee that we have an appropriate model. If in the NE, a social choice function be maximized, and by moving toward NE and having truthful and cooperative behaviors, agents gain more, the system will present desirable results in both levels: individuals and the system as a whole. We prove in the following theorems that the proposed method has these desirable properties in the NE.

Theorem 3: The proposed method encourages agents to be truthful and cooperative.

Proof: If a_i does not tell the truth about its status (does not help others), and insists on this behavior, it gradually is recognized by its neighbors: assume a_j is one of a_i's neighbors that has helped it a lot, but a_i never compensate for that. So, in a_j's probabilities, $P_1(a_i)$, $P_2(a_i)$ and $P_3(a_i)$ and also their functions $B(a_i)$, $P_Q(a_i)$, and $V_Q(a_i)$ have such low values that potentially a_i is not placed in the top k co-workers of a_j. This situation happens in other a_i's neighbors too, and if a_i does not change its dishonest behavior, it incrementally becomes omitted from the network.

If none of agents help others, no one is able to do its tasks.

Now assume a_i helps others in a random way. In this case, its belief distribution over its neighbors does not converge to Distribution 6. If it is not placed in the top k elements of RL agents, it is neglected by them and gradually becomes omitted from their search space. Random agents are not suitable substitutions for lost RLs, by their random and changing behavior. So, agents' utilities decrease in this case too. ■

Theorem 4: If all agents employ the proposed method, total information of the system is maximized in the ex-ante Nash equilibrium.

Proof: The information provided by the system to a_i is

$$I_i = \sum_{\substack{a_{i_j} \in Tm(a_i) \\ j=1,\ldots,k}} P_Q(a_i, a_{i_j}) I(a_i, a_{i_j}),$$

where $P_Q(a_i, a_{i_j})$ is the probability of a_i receiving a_{i_j}'s resources, $Tm(a_i)$ is a_i's team, and $I(a_i, a_{i_j})$ is the value of information (Fisher information) a_{i_j} can provide for a_i.

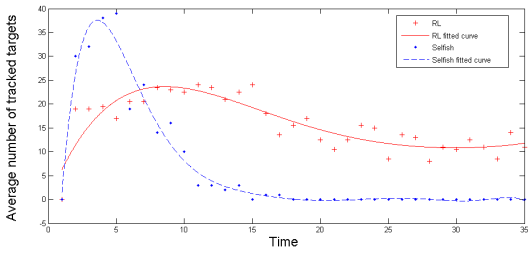

Fig. 1. Comparison between number of tracked targets by RL and Selfish agents

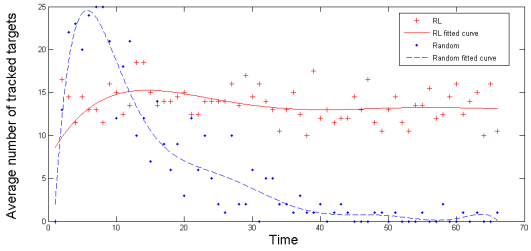

Fig. 2. Comparison between number of tracked targets by RL and Random agents

Gained information of the system is defined as a summation on all agents' obtained information

$$I = \sum_{a_i \in \mathcal{A}} \sum_{\substack{a_{i_j} \in Tm(a_i) \\ j=1,\ldots,k}} P_Q(a_i, a_{i_j}) I(a_i, a_{i_j}). \quad (7)$$

There are two cases about $I(a_i, a_{i_j})$; Case1: Targets have a pattern in the environment; so, each sensor a_i finds the best sensors on the basis of I and tries to increase P_Q of them (as much as it can on the basis of its budget) and this leads to the best allocation provided that all of the sensors behave truthfully and cooperate with each other.

Case 2: Targets move in a random way in the environment, thus

$$E(I(a_i, a_{i_1})) \simeq E(I(a_i, a_{i_2})) \simeq \ldots \simeq E(I(a_i, a_{i_n})) \simeq \mu$$

So, here

$$\sum_{a_i \in \mathcal{A}} \sum_{\substack{a_{i_j} \in Tm(a_i) \\ j=1,\ldots,k}} P_Q(a_i, a_{i_j}) I(a_i, a_{i_j})$$
$$\simeq \mu \sum_{a_i \in \mathcal{A}} \sum_{\substack{a_{i_j} \in Tm(a_i) \\ j=1,\ldots,k}} P_Q(a_i, a_{i_j}) \simeq \mu$$

In Case 2, the precision is not important as all of the neighbors have the same value in a long run. However, reaching μ is dependent on P_Q and by increasing k elements of the set $\{P_Q(a_i, a_{i_j}) | a_{i_j} \in Tm(a_i)\}$, a_i is able to have the maximum probability of obtaining its needed resources and this makes Summation 7 maximized. ■

VI. Experiments

To evaluate our model, a DSN simulator is created whose goal is object tracking in an area. Our aim is to distribute the role of the main controller in a semi-cooperative DSN and present a method on the basis of RL that helps agents to find better teammates. In showing effectiveness of the model, we design different scenarios and evaluate RL agents' performance versus other types.

TABLE I. Comparison among Random agents' benefits, RLs' benefits and social welfare

NRA	NRL	RATT	RLTT	SW
0	64	0	**21.05**	1347
8	56	15.00	20.66	1277
16	48	12.69	21.12	1217
24	40	10.92	21.42	1119
32	32	10.00	22.16	1029
40	24	9.10	22.42	902
48	16	9.79	20.87	804
56	8	10.18	10.87	657
64	0	**7.64**	0	489

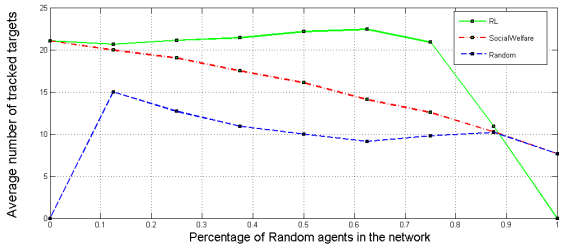

Fig. 3. Effect of number of Random agents on number of tracked targets by RLs and Random agents and social welfare

Our simulator is written in JADE [20]; an agent-based framework implemented in Java [21]. Each sensor and target is designed as an independent thread, so we have a multi-threaded environment which is essential in simulating a multi-agent system [4]. Targets move in a random way in the environment. On detecting a target in its vicinity, a sensor starts to make a team in order to track it. As the environment is semi-cooperative it should negotiate with its team candidates (the negotiation protocol is monotonic concession [22]). If it selects these candidates wisely, the negotiation process finishes in a few rounds. On the other hand, RL agents set their concession policy during the negotiation on the basis of RL results, and give more concessions to the more valuable agents.

To test our theories in practice, we try various scenarios, with different number of sensors (36, 49, 64 and 100) and network configurations. In the first one, RL agents are put in front of selfish (SF) agents. By SFs, we mean agents which get help from others but never try to compensate. Figure 1 shows how RLs can recognize these agents and eliminate them from the network where, vertical and horizontal axis are the average number of tracked targets and time respectively (polynomial curves are fitted to both types' numbers of tracked targets in order to make comparison more easily). Every 20 RL rounds (in the whole system) is considered as a time unit. At the beginning, SFs' tracked targets are a lot, since they just get help from others while RLs are in exploration step and help the system honestly. When RLs start to use Trust factor, SFs are becoming known by them and gradually omitted from the network, so their tracked targets decrease (time≈4) and in contrast, RLs' tracked targets increase. When RLs start to make their final teams (time≈10) their tracked targets decrease[3] to some extent, but remains fixed afterward.

In the second experiment, competition of RLs versus Random (RA) agents is tried. RAs are agents which want to help the system as much as RLs, but their behavior in choosing which agent they help or get help from is fully random. Figure

[3]This is due to using a pure strategy which reduces agents' flexibility in choosing suitable resources.

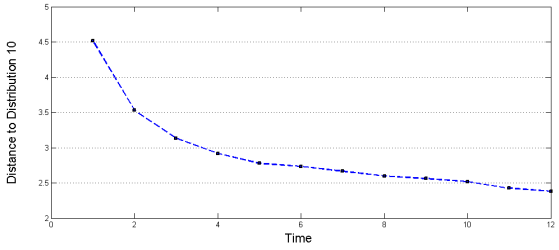

Fig. 4. Reduction of Euclidean distance of RL's probability distributions to Distribution 6 (moving toward Nash Equilibrium)

2 shows results of this scenario, where the analysis is similar to Figure 1: RAs are eliminated from the system by RLs[4].

In another experiment, number of RAs from 0% of the agents rises to 100% (total number of agents remains fixed) and the average number of tracked targets by both RAs and RLs (shown by RATT and RLTT respectively) and also the social welfare[5] (SW) are monitored. Table I and Figure 3 show the results[6], where NRA and NRL represent the number of RAs and RLs respectively. In Figure 3, vertical axis is the average number of tracked targets and the horizontal axis is the percentage of RAs in the network. As NRA rises, SW decreases since RAs' random behavior prevent them to be good replacements for deleted RLs, and the network, even after detecting many targets, cannot monitor them (needed teams are not formed on time in many cases). When the NRL decreases to 25%, RLTT starts to reduce severely, because RAs are not good substitutions for their omitted RL teammates.

The most surprising point is about the difference between RLTT and RATT when the network contains just one type, as RLTT is greater than RATT almost three times! Note that in this case RAs are not isolated by RLs, and they decide to help each other as much as RLs (the mathematical expectation of the number of times they decide to help the system is the same for both types) and this huge difference in utilities shows effectiveness of our method.

Another point is about RATT which decreases as NRA increases, and we can tell two reasons for that: first, by increasing the number of RAs, RLs' population decreases. This is not a good situation for RAs since they lose some agents' help, before being known by them (when RLs are in exploration step), as RLs help them more than their own type by their steady behavior in these moments. Second, by increasing NRA, in average, RAs detect less targets in the environment and this leads RATT reduces (We put mean density of targets fixed in all of the runs). However, when NRA reaches more than 60% of the agents, RATT starts to increase since they have enough power to not being isolated by RLs (RLs are in the minority in this situation).

In another experiment, the network contains just RLs (with pure strategies). Our aim is to test whether or not RLs' belief distributions over their neighbors approach to Probability Distribution 6 (which shows they move toward NE). Figure 4 demonstrates that Euclidean distance of RLs' distributions to

[4]In this scenario, the number of RAs is 1/3 of all agents and k=5

[5]Here social welfare is defined as the total number of tracked targets by the system. We want to use more complicated functions for the future.

[6]In order to track the trend more easily, we divide social welfare by the total number of agents in Figure 3.

TABLE II. COMPARISON BETWEEN MIXED-STRATEGY AND PURE-STRATEGY RLS' BENEFITS

	Rejection	Tracking Targets	Helping Others
Pure	13390	10.80	43.31
M7	10162	26.36	77.22
M6	10845	11.85	55.03
M6&7	10422	20.92	68.89

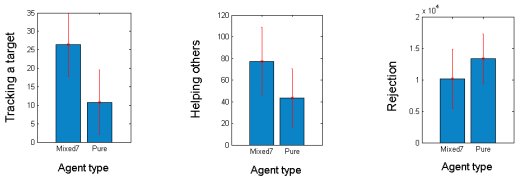

Fig. 5. Comparison between Mixed-strategy and Pure-strategy agents' gains from the system

Distribution 6 decrease as the time passes and this indicates the system moves toward stability (NE)[7].

The last experiment is about a network with pure-strategy RL agents (PRLs) and mixed-strategy RL agents (MRLs). By MRLs we mean some agents which make teams with $k+r$ agents while needing k resources, but choosing k elements from this set in a fully random manner. Although, having $k+r$ friends makes MRLs more powerful than PRLs, it forces them to use more resources, and thus, this strategy adapts with stronger agents (which have enough resources). Our goal by this scenario is to offer a mechanism by employing it, agents which spend more resources to help others, can get more in return and the results show effectiveness of this plan. In our scenario, we set $k = 4$ and $r = 3$. However, some agents cannot find 7 members to make teams with them (they have not been accepted by 7 agents), and have to acquiesce to 6-member teams. Table II shows the results of running this scenario several times and getting an average between these runs (the numbers of both types are equal in the network). By M7, M6 and M6&7 we mean MRLs which make 7, 6 and 6 or 7-member teams respectively. As you can see, M7s help the system 19% more than PRLs, but get help from the system 32% more, and the difference between these two numbers (i.e. 13%) can be regarded as the pure revenue of MRLs from applying a mixed strategy policy rather than a pure one (this can be seen as the synergy of making bigger groups too). In Table II, Rejection shows the number of times each type gets rejection from its neighbors (in the whole time of system) when needing their help, and again M7s win PRLs. Figure 5 displays a better comparison between just M7s and PRLs. The standard deviation is another factor that again in tracking targets (getting help from the system) M7s surpass PRLs since they have a standard deviation equal to 1.00 rather than PRLs whose standard deviation equals 1.63, and this shows the system is more reliable for M7s.

VII. CONCLUSION AND FUTURE WORK

In this paper, a distributed method on the basis of reinforcement learning (RL) is presented which helps agents to find better teammates and encourages truthful and cooperative behaviors in the system.

To test our mechanism in practice, it is applied to a distributed sensor network designed for target tracking. Sim-

[7]We set $P = \frac{1}{2}(P_Q + P_C)$ and then normalize it in this experiment.

ulation results show effectiveness of the mechanism since RL agents surpass selfish and random-policy agents. In addition, it is shown that by using a mixed-strategy, RL agents can get more resources from the system in the need time and enjoy further synergy produced by forming larger teams.

for the future, we aim to use mixed-strategy RL agents to receive more accuracy from the system. We also want to try more complicated social welfare, utility and cost functions, and design experiments for evaluating roles of Balance and Trust factor independently in the system.

REFERENCES

[1] X. S. Zhang and V. Lesser, "Meta-level coordination for solving distributed negotiation chains in semi-cooperative multi-agent systems," *Group Decision and Negotiation*, vol. 22, pp. 681–713, mar 2012.

[2] A. Rogers, R. K. Dash, N. Jennings, S. Reece, and S. Roberts, "Computational mechanism design for information fusion within sensor networks," in *Proc. IEEE International Conference on Information Fusion*, Florence, Italy, jul 2006, pp. 1–7.

[3] R. K. Dash, S. D. Ramchurn, and N. R. Jennings, "Trust-based mechanism design," in *Proc. IEEE International Joint Conference on Autonomous Agents and Multiagent Systems*, vol. 2, New York, NY, USA, jul 2004, pp. 748–755.

[4] M. Wooldridge, *An Introduction to Multiagent Systems*. UK: Wiley, 2008.

[5] R. Dash, A. Rogers, N. Jennings, S. Reece, and S. Roberts, "Constrained bandwidth allocation in multi-sensor information fusion: A mechanism design approach," in *Proc. IEEE International Conference on Information Fusion*, vol. 2, Philadelphia, PA, USA, jul 2005, pp. 1185–1192.

[6] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm Computing Surveys (CSUR)*, vol. 38, pp. 1–45, dec 2006.

[7] M. E. Gaston and M. desJardins, "Agent-organized networks for dynamic team formation," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, Utrecht, Netherlands, jul 2005, pp. 230–237.

[8] R. Katayanagi and T. Sugawara, "Efficient team formation based on learning and reorganization and influence of communication delay," in *Proc. IEEE International Conference on Computer and Information Technology (CIT)*, Paphos, Cyprus, aug & sep 2011, pp. 563–570.

[9] J. Ahn, X. Sui, D. DeAngelis, and K. S. Barber, "Identifying beneficial teammates using multi-dimensional trust," in *Proc. International joint conference on Autonomous agents and multiagent systems-Volume 3*, may 2008, pp. 1469–1472.

[10] C. E. Pippin and H. Christensen, "Cooperation based dynamic team formation in multi-agent auctions," in *Proc. SPIE Defense, Security, and Sensing*, Maryland, US, may 2012.

[11] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence*, vol. 4, pp. 237–285, may 1996.

[12] R. Glinton, P. Scerri, and K. Sycara, "Agent-based sensor coalition formation," in *Proc. IEEE International Conference on Information Fusion*, Cologne, Germany, jun 2008, pp. 1–7.

[13] B. Horling, R. Mailler, and V. Lesser, "A case study of organizational effects in a distributed sensor network," in *Proc. IEEE International Conference on Intelligent Agent Technology (IAT)*, Beijing, China, sep 2004, pp. 51–57.

[14] L. Soh and X. Li, "A learning-based coalition formation model for multiagent systems," in *Proc. ACM International Joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne, Australia, jul 2003, pp. 1120–1121.

[15] G. Mainland, D. Parkes, and M. Welsh, "Decentralized, adaptive resource allocation for sensor networks," in *Proc. USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, vol. 2, Boston, MA, USA, may 2005, pp. 315–328.

[16] T. Le, T. Norman, and W. Vasconcelos, "Adaptive negotiation in managing wireless sensor networks," in *Proceedings of the 13th International Conference on Principles and Practice of Multi-Agent Systems*, Kuching, Sarawak, Malaysia, sep 2012, pp. 121–136.

[17] R. Dash, "Distributed mechanisms for multi-agent systems: Analysis and design," Ph.D. dissertation, Univ. of Southampton, Southampton, jun 2006. [Online]. Available: http://eprints.soton.ac.uk/id/eprint/262727

[18] S. Ramchurn, C. Mezzetti, A. Giovannucci, J. A. Rodriguez, R. K. Dash, and N. R. Jennings, "Trust-based mechanisms for robust and efficient task allocation in the presence of execution uncertainty," *Journal of Artificial Intelligence Research*, vol. 35, pp. 119–159, jun 2009.

[19] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[20] TelecomItalia. (2012) Java agent development framework (jade). [Online]. Available: http://jade.tilab.com/

[21] F. Bellifemine, A. Poggi, and G. Rimassa, *Developing Multi-agent Systems with JADE*. Chichester, England: John Wiley & Sons, 2001.

[22] J. S. Rosenschein and G. Zlotkin, *Rules of Encounter: designing conventions for automated negotiation among computers*. Cambridge, MA: The MIT Press, 1994.