Information Sharing and Searching via Collaborative Reinforcement Learning

George A. Vouros

Department of Digital Systems, University of Piraeus, Piraeus Greece
georgev@unipi.gr

Abstract. This paper proposes a method for computing a routing policy-value function for effective information sharing and searching in arbitrary networks of agents through collaborative reinforcement learning. This is done by means of local computations performed by agents and payoff propagation. The aim is to 'tune' a network of agents for efficient and effective information searching and sharing, without altering the topology or imposing an overlay structure.

1 Introduction

'*Tuning*' networks of agents to perform information searching and sharing requires agents to gather the *necessary* knowledge so as to decide their routing policies and propagate requests to the *right* agents, minimizing the searching effort, thus increasing the efficiency (i.e. speed/cost ratio) and the efficacy (i.e. retrieved results) of the task.

Considering to be a decentralized control problem, information searching and sharing in large-scale systems of cooperative agents is a hard problem in the general case: The computation of an optimal policy, when each agent possesses an approximate partial view of the state of the environment and when agents' observations and activities are interdependent (i.e. one agent's actions affect the observations and the state of an other) [2], is hard. This fact, has resulted to efforts that either require agents to have a global view of the system [14], to heuristics [3], to the pre-computation of agents' information needs and information provision capabilities for proactive communication [15], to localized reasoning processes built on incoming information [11,12,13], to mathematical frameworks for coordination, whose optimal policies can be approximated for small (sub-) networks of associated agents [10], and to reinforcement learning algorithms for hierarchical peer-to-peer information retrieval systems [16]. On the other hand, there is a lot of research on semantic peer to peer search networks and social networks many of which deal with tuning a network of peers for effective information searching and sharing. They do it mostly by imposing logical and semantic overlay structures.

To address the limitations (related to scalability and the assumptions made) of the above-mentioned approaches, the information sharing and searching algorithms reported in [8,9] use routing indices and agents' profiles. There are three important issues regarding these methods: (a) Given that random information sharing and searching is a hard competitor for any such task [7], the effectiveness of these methods concerning the number of information items retrieved is not that high, even

I. Maglogiannis, V. Plagianakos, and I. Vlahavas (Eds.): SETN 2012, LNAI 7297, pp. 132–140, 2012.
© Springer-Verlag Berlin Heidelberg 2012

if they manage to gradually increase the efficiency of the searching task. (b) Agents do not consider the cost or delay of their task even if the overall benefit increases. For instance, requests may be propagated via specific agents whose load (thus throughput time) increases; and finally, (c) the convergence of the method cannot be guaranteed.

To address the limitations of previous works, this paper proposes using the agent-based update of the edge-based decomposition Sparse Cooperative Q-learning method proposed in [5], so as agents to estimate the value of their routing policies in a distributed and scalable way, by exploiting the dependencies of their routing actions. The proposed method, to a greater extent than other proposals concerning the use of reinforcement learning techniques for information retrieval [16] exploits the structure of the problem by means of coordination graphs [4] and thus, can result to very effective tuning, even to non-hierarchical (arbitrarily organized) systems. Also, the proposed approach generalizes approaches that exploit routing indices [8,9] in two directions: (a) It estimates routing policy-value functions via payoff propagation, and (b) it has the potential to incorporate any problem/context specific parameter concerning routing actions value.

2 Problem Specification

The setting: Let $N=\{A_1, A_2, \ldots, A_n\}$ be the set of agents in the system. The network of agents is modelled as an acquaintances graph $G=(N,E)$, where N is the set of agents and E is a set of bidirectional edges denoted as non-ordered pairs (A_i,A_j). The neighbourhood of an agent A_i (denoted by $N(A_i)$) includes all the acquaintances of A_j such that $(A_i,A_j)\in E$. Each edge is associated with a communication cost, $Cost(A_i,A_j)$ and each agent has a specific service rate $SR(A_i)$ that specifies its maximum throughput rate (i.e. the number of queries that it processes in the unit of time). The (mean) delay of each agent A_i to process a query, is determined as follows: $Delay(A_i)=$ *Number_of_Waiting_Queries/*$SR(A_i)$. We assume that the cost and the delay are associated to comparable values. It must be noticed that in contrast to costs, delays change over time.

The agents in the network share the same set of information categories C. Each agent assigns a value (*i-Value*) to each information category. This specifies the reward one gets by obtaining an information item from that category. Also, any agent has an *expertise,* which is represented by a specific information category, and it possesses a unique *information item* (e.g. a document) of that category. Additionally, we consider a set of k queries $\{q_1,\ldots, q_k\}$. Each query asks for an information item under a specific category c in C, and is represented by a triple $<id,c,TTL>$, where *id* is the identity of the query, and *TTL* (Time To Live) is a positive integer that specifies the *maximum* number of hops that *any* query can reach.

Definitions: Considering a specific query $q=<id,c,TTL>$, a *search session* for this query is associated with a specific path in the network and starts when the query is generated by a specific agent and finishes either when the query has been answered (*served*) or when it has traversed a path in the network whose length is equal to the specified Time To Live (*TTL*) without being served (*unfulfilled*).

At a specific point of time, the *state of a search session* for a specific query comprises the variables *path (*which initially is empty, and each agent adds its id in

the path before propagating the query), *ttl* (which is initially equal to *TTL* and it is reduced by one for each hop), and the variable *agent* (specifying the agent currently processing the query). Thus, given a session at state $<path,ttl,A_i>$, and in case A_i routes the query to A_j, then the new state for this session is $<A_j \oplus path, (ttl-1), A_j>$, where $(A_j \oplus path)$ is the extension of *path* with A_j. The *joint state* comprises the variables of all search sessions.

The finite set of routing actions available to each agent A_i with respect to the query q correspond to its *G*-neighbors $N(A_i)$: More precisely, given the state of the search session $<path,ttl,A_i>$ for $q=<id,c,TTL>$, A_i may send that query to any of its *G*-neighbors that has the "best" potential among neighbours to route a query for c so as to be served with a minimum cost and delay (compared to the other *G*-neighbors) in $ttl' \leq (ttl-1)$ hops (i.e. in less hops than those required).

Coordination Graphs: In the following we consider that agents are organized in *coordination graphs* [4]. Generally, these are structured according to the dependencies between agents' actions, revealing the structure of the coordination problem: Coordination graphs allow decomposing a global value function for any joint action of agents, into a sum of local functions: Each local function depends on the combination of actions of the involved agents. In contrast to the acquaintance graph $G=(N,E)$, we denote coordination graphs as $CG=(N_{CG},E_{CG})$. Also, when it is not clear, we distinguish between *G*-neighbors (i.e. neighbors in the acquaintance graph) and *CG*-neighbors (i.e. neighbors in the coordination graph).

Given the acquaintance graph *G*, the coordination graph $CG=(N_{CG},E_{CG})$ is as follows: Each node in N_{CG} corresponds to an agent A_i in the acquaintance graph with two additional attributes: The type of the requested information and the *ttl* of the request (we denote such an agent by $A_{i,c,ttl}$, where $c \in C$ and $ttl=1...TTL$). Therefore, for each agent in *G* there are $|C| \times TTL$ nodes in *CG*. Two nodes $A_{i,c,ttl}$ $A_{j,c',ttl'}$ with $i \neq j$ are connected with a directed edge $(A_{i,c,ttl}\, A_{j,c',ttl'})$ iff A_j is in $N(A_i)$, $c=c'$ and $ttl'<ttl$. In this case $A_{j,c',ttl'}$ (resp. $A_{i,c,ttl}$) is a *CG*-neighbor (resp. *inverse CG*-neighbor) of $A_{i,c,ttl}$ (resp. of $A_{j,c',ttl'}$). Indeed, the routing action of A_i given the state $<path,ttl,A_i>$ of a search session for $q=<id,c,TTL>$, denoted by $act_{i,c,ttl}$, depends on the routing action $act_{j,c,ttl'}$, $ttl' \leq (ttl-1)$, of any G-neighbor A_j, given that, the return received by A_i, in case it propagates the query to A_j, depends on the routing decision of A_j.

It must be noticed that *CG* is not constructed in an explicit way by the proposed method. *CG* provides a structure to the information searching and sharing task and differs substantially from *G*: (a) While routing actions concern propagating queries in the "actual network", i.e. in *G,* the necessary information for valuating routing actions is gathered and propagated in *CG;* (b) *CG* has $|C| \times TTL$ more nodes than *G;* (c) if A_j is in $N(A_i)$, then not any pair of nodes $A_{i,c,ttl}$ $A_{j,c',ttl'}$ is connected (e.g. there is not any edge from $A_{i,c,3}$ to $A_{j,c,5}$); (d) *CG* is a directed and acyclic graph: Indeed, there can not be any path $(A_{i,c,ttl}, \ldots, A_{i,c,ttl})$ since a pair of nodes $A_{i,c,ttl}\, A_{j,c,ttl'}$ is connected iff $ttl'<ttl$.

The problem: Each agent A_i has a finite set of routing actions $\mathbf{Act_i}$[1]. This set comprises all the routing options an agent has for any information category and state of a search

[1] To distinguish symbols, we have used lowercase letters for atomic states/actions/policies/rewards, uppercase letters for joint states/actions/policies/rewards and uppercase-bold letters for sets of states/actions/policies.

session. The joint action is the combination of agents' individual actions, and thus a member of $\boldsymbol{Act}=\boldsymbol{Act_1}\times\ldots\times\boldsymbol{Act_n}$. Given a set of discrete state variables $\boldsymbol{S_i}$ (whose values depend on the state of specific query sessions), the joined state of the system at a specific time point t is defined to be a member of $\boldsymbol{S}=\boldsymbol{S_1}\times\ldots\times\boldsymbol{S_m}$. A state transition function $T:\boldsymbol{S}\times\boldsymbol{Act}\times\boldsymbol{S} \rightarrow [0,1]$ gives the transition probability $p(S^{t+1}|S^t,Act^t)$ that the system will reach state S^{t+1} when the joint action Act is performed at the time point t, when the system is at state S. A reward function $r_i:\boldsymbol{S}\times\boldsymbol{Act} \rightarrow \mathbb{R}$ provides each agent A_i with an individual reward, depending on the joint action Act performed in state S.

Given that the model assumes the Markov property and that the reward and transition probabilities are independent of the time t, the successor of a state S (or s) given an action Act (resp. act), is denoted by S' (resp. s').

A policy $\boldsymbol{\Pi}:\boldsymbol{S}\rightarrow\boldsymbol{Act}$ specifies a joint action Act for each joint state S.

The objective of tuning the information sharing and searching task is to find an optimal policy $\boldsymbol{\Pi}^*$ that maximizes the expected discounted future return $V^*(s) = max_{\boldsymbol{\Pi}} \mathrm{E}[\Sigma_t \gamma^t R(S^t, \Pi(S^t))|\Pi, S^0 = S]$ for each state S. The expectation operator E[.] averages over stochastic transitions, R is the global reward and $\boldsymbol{\gamma}\in[0,1)$ is the discount factor.

3 The Proposed Approach

Briefly, the overall process is as follows: Each agent gets payoff updates from its *CG*-neighbors, it propagates updated local payoffs to its *inverse CG*-neighbors, uses computed payoffs to estimate routing action values, and processes/ routes own queries. The estimation of the routing action values happens through collaborative reinforcement learning, while the estimation of payoffs – which are exploited by the learning method- via the max-plus algorithm.

The objective of collaborative reinforcement learning is to support agents to select a joint policy that provides them the highest possible reward. Agents have no prior knowledge about the effect of their actions, but this information has to be learned based on the received rewards. We use the collaborative multiagent Markov decision process (collaborative multiagent MDP) model [4], also used in [5]. In this model each agent selects an individual action, given a particular state. Based on the resulting joint action the system transitions to a new state and the agents receive an individual reward. The global reward is the sum of all individual rewards. No agent can observe the global reward. In a collaborative MDP, the goal of the agents is to optimize the global reward. The individually received rewards allow for solution techniques that take advantage of the problem structure revealed by coordination graphs.

Recall that the objective is to find an optimal policy $\boldsymbol{\Pi}^*$ that maximizes the expected discounted future return. Q-functions (action-value functions) represent the expected future discount reward for a state S when selecting an action Act and behaving optimally from then on. To approximate the global Q-function we use the agent-based update of the edge-based decomposition Sparse Cooperative Q-learning method proposed in [5].

Therefore, given a coordination graph of agents, each edge $(A_{i,c,ttl},A_{j,c,ttl'})$ in E_{CG} corresponds to a local Q-function Q_{ij}. The global Q-function is the sum of all local functions

$$Q(S,Act) = \sum_{(A_{i,c,ttl},A_{j,c,ttl'})} Q_{ij}(S_{ij},act_{i,c,ttl},act_{j,c,ttl'}) \quad (1)$$

where $\boldsymbol{S}_{ij} \subseteq \boldsymbol{S}_i \cup \boldsymbol{S}_j$ is the subset of state variables that are relevant to the dependency between agents A_i and A_j with respect to c. The local Q-function Q_i of agent A_i is defined as the summation of half the value of all Q-functions Q_{ij}, where $A_{j,c,ttl'} \in N_{CG}(A_{i,c,ttl})$:

$$Q_i(S_i,act_{i,c,ttl}) = \sum_{(A_{i,c,ttl},A_{j,c,ttl'})} Q_{ij}(S_{ij},act_{i,c,ttl},act_{j,c,ttl'}) \quad (2)$$

The agent update (edge decomposition) method computes the temporal difference error per agent and divides this value over the edges. The Q-function of an edge incorporates the information from *all* edges of *each of* the involved agents:

$$Q_{ij}(S_{ij},act_{i,c,ttl},act_{j,c,ttl'}) = Q_{ij}(S_{ij},act_{i,c,ttl},act_{j,c,ttl'}) + a \sum_{A_{k,c,t} \in \{A_{i,c,ttl},A_{j,c,ttl'}\}} \frac{r_k(S,Act) + \gamma Q_k(s'_k,act^*_{k,c,t}) - Q_k(s_k,act_{k,c,t})}{|N_{CG}(A_{k,c,t})|} \quad (3)$$

The discount factor γ is set to 0.3 and the learning rate α to 0.2. act^*_k is the maximizing action of agent A_k in the state s'_k. The reward $r_k(S,Act)$ for each of the agents A_k in an edge is set to be equal to the i-*Value* (i$Value(A_k,c)$) of the searched information category c for the agent A_k. The value of the maximizing action of agent A_k for a search session at state s'_k=($path,ttl,A_k$) is estimated as follows: $Q_k(s'_k,act^*_{k,c,ttl})=max(g_k(act_{k,c,ttl}))$, where g_k depends on the payoffs propagated via the max-plus algorithm (explained in the next paragraphs). $Q_k(s_k,act_k)$ is computed as specified by equation (2).

Thus, each local Q-function Q_{ij} is updated with a proportional part of the received reward of the two agents it is related to and with the contribution of this edge to the maximizing joint action $act_{k,c,ttl}$* in the state s_k. As already said, this is approximated by the max-plus algorithm. Using this combination of methods, as it is shown in [8], the edge-based decomposition scales linearly in the number of CG-neighbors. The update of the action-value for an agent is based on the current Q-value and the local contribution of this agent to the global return.

In order to compute the optimal joint action that maximizes the sum of agents' local payoffs (i.e. the global payoff), we use message passing algorithms [6], and particularly the max-plus algorithm. According to this algorithm, given a query <id,c,TTL> a session of which is at state <$path,ttl, A_i$>, each agent sends a message μ_{ij} to each of its CG-neighbors. Allowing only payoff functions defined over at most two agents in CG the computation of payoffs is as follows:

$$\mu_{ij}(act_{j,c,ttl}) = \max_{act_{i,c,ttl'}} \left\{ F_{ij}(act_{i,c,ttl'},act_{j,c,ttl}) + \sum_{A_k \in N_{CG}(A_i)-\{A_j\}} \sum_{t=1}^{ttl'} \mu_{ki}(act_{i,c,t}) \right\} + c_{ij} \quad (4)$$

The sums concern the local payoffs of all CG-neighbors of $A_{i,c,ttl'}$, except those related to A_j. Given that $F_{ij}(act_i,act_j) = f_i(act_i) + f_i^j(act_i,act_j)$, f_i specifies the payoff contribution of $A_{i,c,ttl}$. Formally, $f_i(act_{i,c,ttl})$= $iValue(A_i,c)$-$Delay(A_i)$.

Also, f_i^j is the payoff contribution of the pair of neighbors $A_{i,c,ttl}$ and $A_{j,c,ttl'}$, given their actions $act_{i,c,ttl}$, $act_{j,c,ttl'}$. Formally,

$$f_i^j(act_{i,c,ttl}, act_{j,c,ttl'}) = \begin{cases} -Cost(A_i, A_j) & if \quad act_{i,c,ttl} \; routes \quad to \; A_j \\ -COST & otherwise \end{cases}$$

where *COST* is a very large number[2]. It must be pointed out that these functions specifying payoff contributions are rather simple and independent of the *ttl* parameter. Future work concerns elaborating them further by taking into account further attributes of the problem and of the setting.

Given a query <*id,c,TTL*> a session of which is at state <*path,ttl, A_i*>, the agent A_i may at any time step compute

$$g_i(act_{i,c,ttl}) = f_i(act_{i,c,ttl}) + \sum_{A_k \in N(A_i)} \mu_{ki}(act_{i,c,ttl}) \tag{5}$$

which equals the contribution to the global payoff function achieved via A_i's subtrees. Thus, each agent can form a decision by approximating its optimal choice regarding *c* and *ttl:*

$$act^*_{i,c,ttl} = \arg\max_{act_{i,c,ttl}} g_i(act_{i,c,ttl}) \tag{6}$$

Although the max-plus algorithm converges to fixed message values, Q-learning does not converges to the optimal Q* values for multiple, independent learners, since the decisions of one agent affect in a dynamic way the actions of the others [5].

Overall, the task is as follows: Given an agent A_i in *G*, and a query <*id,c,TTL*> whose session is at state <*path,ttl, A_i*>, A_i routes the query to a percentage of its *G*-neighbors (let that be *AP – AcquaintancesPercentage*). Routing decisions are formed by exploiting routing action values estimated by means of equation (3), or, initially, purely randomly: Initially, agents have no estimation about the information provision abilities and costs associated to any of their neighbors. Also, even in the case that an agent has an estimation of routing actions' values, it explores further possibilities by propagating the queries to randomly chosen *G*-neighbors, as well. When the query reaches an agent A_k that possesses an item in category *c*, then this agent sends the answer directly to the originator of the query. Given that the search session for the corresponding query is at state <*path',ttl', A_k*>, A_k calculates and propagates its payoff to the *inverse CG*-neighbors of $A_{k,c,ttl'}$. Then, payoff propagation due to this update proceeds concurrently to the routing of other queries and, of course, to the estimation of agents' routing action values. Payoffs are getting propagated until they converge to fixed values: This is always the case, given that the constructed *CG*s are acyclic.

4 Experimental Results

To validate the proposed approach we have built a prototype that simulates information sharing in networks of agents. To test our approach we have run several experiments with random and small-word networks. Due to space reasons, we present results for random networks only. Networks comprise 50 agents (|*N*|=50). Results are

[2] The rationale behind this, is that, when A_i considers selecting a routing action to A_j, then the payoffs of the other neighbors become irrelevant.

representative for cases with larger networks of agents. The average number of acquaintances per agent in our experiments is 13. Each experiment ran 5 times for 40 rounds at each round. At each round a constant number of 330 queries are being generated. Each query is randomly assigned to an originator agent and is set to request one information item of a randomly chosen category. The *TTL* for every query is set to be equal to 7. In such a setting, the demand for information items is high, given agents' information provision abilities and the *TTL* of queries. To end a round, all query sessions must have been ended, and all payoff messages must have been converged to fixed values. Information used in the experiments is synthetic and is being classified in 15 distinct categories: Agents' expertise and information values for each of the categories is determined randomly. The percentage of acquaintances (*AP*) to which a query can be propagated is set to 1,10 or 20, so as to show the efficacy of the method, even if agents perform restricted exploration. To demonstrate the advantages of our method we provide results for different configurations: (a) Agents propagate queries according to the best routing action estimated using only the max-plus algorithm (equation (6)). This configuration is indicated by "MS"; (b) Agents propagate their payoff estimations *only* to the inverse-*CG* neighbors that show high interested to each corresponding category of information *c* (indicated by "P"). The estimation of agents' interests is done as proposed in [8,9]. (c) We also provide results from a baseline method where agents select their routing actions randomly (indicated by "R"). The experimental settings are denoted by X-Y-AP-W, where X denotes the type of network, Y the number of agents, *AP* the percentage of acquaintances to which queries are propagated, and W is either "MS", "R", or unspecified for the "standard" cooperative Q-learning setting. For instance, Rand-50-10-MS denotes a setting with a random network of 50 agents where each query is being propagated to at most 10% of an agent's acquaintances (*AP=10*), and agents decide on their routing actions using equation (6). Results computed in each experiment show the total number of query-propagation messages (*q-messages*), the total number of messages for the propagating payoffs (*p-messages*), the *benefit* of the system, i.e. the percentage of served queries, and the *message gain,* i.e. the ratio of benefit to the total number of messages. Experimental results are depicted in Figure 1.

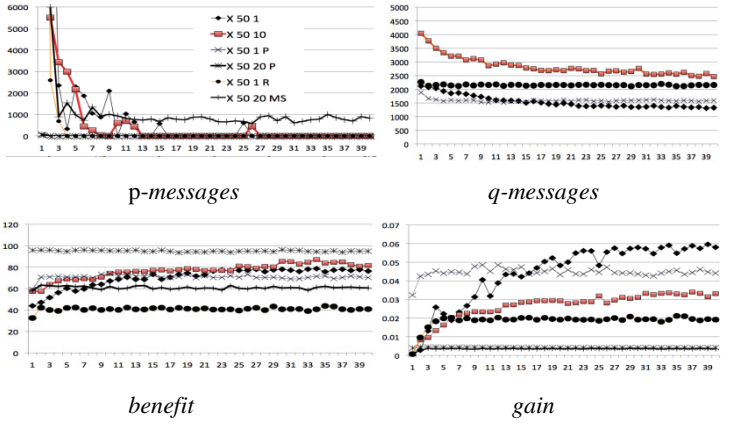

Fig. 1. Results for X=Rand. The horizontal axis shows the round number.

Results lead us to the following conclusions: (a) In any configuration of the method, the number of *p-messages* in the network is drastically reduced to 0. Actually, random routing achieves the greater reduction of *p-messages* in the first 5 rounds. As *AP* increases, in all settings, the number of *p-messages* decreases more quickly. In the "MS" configuration, the number of p-messages reaches a plateau far above zero. On the contrary, in the "P" settings, where agents exploit profiles of acquaintances, the number of *p-messages* is very small and they reach zero very fast. (b) The number of *q-messages* is also gradually reduced: Greater *AP* results to a greater number of messages. Also, the "R" and "P" configurations do not manage to decrease the number of *q-messages* (these are constantly very large - not shown in the diagram). This makes us concern about "when and how profiles should be used?". (c) Things are different for the benefit achieved: Random routing proves to be competitive, while routing taking into account agents' profiles is very effective even from the first rounds: It achieves nearly 100% benefit when messages are propagated to 20% of agents' acquaintances. However, when agents use the cooperative Q-learning method proposed they increase the benefit considerably: Increasing *AP* results to slightly larger benefit. (d) Concerning gain, in all settings, it increases as *AP* reduces: Configurations exploiting 1% of acquaintances are more effective given that they achieve high benefit, with a small number of messages.

5 Concluding Remarks

Aiming to tuning the information searching task in arbitrary networks of agents, we compute an approximation of the global routing policy-value function through collaborative reinforcement learning. Specifically we use the agent-based update of the edge-based decomposition Sparse Cooperative Q-learning method proposed in [5]. The proposed method exploits dependencies between agents' actions and thus, can result to very effective tuning, even to non-hierarchical (arbitrarily organized) systems. This is demonstrated by the results in a number of experimental settings discussed: The method is very effective even if we restrict exploration or payoff propagation. Future work aims to study optimality in various settings, studying deeply the exploitation of profiles.

References

1. Cooper, B.F., Garcia-Molina, H.: Ad hoc, self-supervising peer-to-peer search networks. ACM Transactions on Information Systems (TOIS)archive 23(2), 169–200 (2005)
2. Goldman, C., Zilberstein, S.: Decentralized Control of Cooperative Systems: Categorization and Complexity Analysis. JAIR 22, 143–174 (2004)
3. Goldman, C., Zilberstein, S.: Optimizing Information Exchange in Cooperative Multi-agent Systems. In: Proc. of AAMAS 2003 (July 2003)
4. Guestrin, C., Koller, D., Parr, R.: Multiagent planning with factored MDPs. In: NIPS, vol. 14 (2002)
5. Kok, J.R., Vlassis, N.: Collaborative Reinforcement Learning by Payoff Propagation. JMLR 7, 1789–1828 (2006)

6. Kschischang, F.R., Frey, B.J., Loeliger, H.-A.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47(2) (2001)
7. Velagapudi, P., Prokopyev, O., Scerri, P., Sycara, P.: Analyzing the Performance of Randomized Information Sharing. In: Proc. of AAMAS 2009, pp. 821–828 (2009)
8. Vouros, G.: Information Searching and Sharing in Large-Scale Dynamic Networks. In: Proc. of AAMAS 2007, pp. 235–242 (2007)
9. Vouros, G.: Searching and Sharing Information in Networks of Heterogeneous Agents. In: Proc. of AAMAS 2008, Poster (2008)
10. Xu, Y., Scerri, P., Yu, B., Lewis, M., Sycara, K.: A POMDP Approach to Token-Based Team Coordination. In: Proc. of AAMAS 2005, Utrecht, July 25-29. ACM Press (2005)
11. Xu, Y., Lewis, M., Sycara, K., Scerri, P.: Information Sharing in Large Scale Teams. In: Proc. of Workshop on Challenges in Coordination of Large Scale MultiAgent Systems (2004)
12. Xu, Y., Liao, E., Scerri, P., Yu, B., Lewis, M., Sycara, K.: Towards Flexible Coordination of Large Scale Multi-Agent Systems. In: Challenges of Large Scale Coordination (2005)
13. Xu, Y., Scerri, P., Yu, B., Okamoto, S., Lewis, M., Sycara, K.: An Integrated Token Based Algorithm for Scalable Coordination. In: Proc. of AAMAS 2005, pp. 407–414 (2005)
14. Xuan, P., Lesser, V., Zilberstein, S.: Communication Decisions in Multi-agent Cooperation: Model and Experiments. In: Proc. of AGENTS 2001, pp. 616–623 (2001)
15. Zhang, Y., Volz, R., Ioeger, T.R., Yen, J.: A Decision Theoretic Approach for Designing Proactive Communication in Multi-Agent Teamwork. In: SAC 2004, pp. 64–71 (2004)
16. Zhang, H., Lesser, V.: A Reinforcement Learning based Distributed Search Algorithm For Hierarchical Peer-to-Peer Information Retrieval Systems. In: Proc. of AAMAS 2007, pp. 231–238 (2007)