

Maximal Clique Based Distributed Coalition Formation for Task Allocation in Large-Scale Multi-agent Systems

Predrag T. Tošić and Gul A. Agha

Open Systems Laboratory, Department of Computer Science,
University of Illinois at Urbana-Champaign,
Mailing address: Siebel Center for Computer Science,
201 N. Goodwin Ave., Urbana, IL 61801, USA
{p-tosic, agha}@cs.uiuc.edu

Abstract. We present a fully distributed algorithm for coalition formation among autonomous agents. The algorithm is based on two main ideas. One is a distributed computation of maximal cliques (of bounded sizes) in the underlying graph that captures the interconnection communication topology of the agents. Hence, given the current configuration of the agents, the coalitions that are formed are characterized by a high degree of connectivity, and therefore a high fault tolerance with respect to the subsequent node and/or link failures. The second idea is that each agent chooses its most preferable coalition based on how highly the agent values each such coalition in terms of the coalition members' combined resources or capabilities. Coalitions with sufficient resources for fulfilling highly desirable tasks are preferable to the coalitions with resources that suffice only for completing less valuable tasks. We envision variants of our distributed algorithm presented herein to prove themselves useful coordination subroutines in many massively multi-agent system applications where the agents may repeatedly need to form temporary groups or coalitions of modest sizes in an efficient, online and fully distributed manner.

Keywords: distributed algorithms, large-scale multi-agent systems, distributed group formation, agent coalitions.

1 Introduction and Motivation

Agent coordination poses a number of challenges to a designer of a large-scale *multi-agent system (MAS)*. In particular, in order to be able to effectively coordinate, agents need to be able to *reach consensus* on various matters of common interest. The two particularly prominent distributed consensus problems that often arise in *MAS* applications are those of *leader election* (e.g., [7, 18]) and *coalition formation*. Group or coalition formation is an important issue in distributed systems in general (e.g., [7]), and *MAS* in particular (e.g., [29, 25]). Given a collection of communicating agents, the goal in distributed coalition formation is that these agents, based on their local knowledge only, decide how to effectively self-organize into coalitions, so that each agent knows which coalition(s) it belongs to.

There are several critical issues that the *MAS* designer needs to address in the context of distributed coalition formation. First, what is the desired notion of a coalition

in a given setting? Second, a distributed coalition formation mechanism - that is, a distributed algorithm that enables the agents to effectively form coalitions - needs to be provided. Third, coalitions and each agent's knowledge about its coalition membership need to be maintained and, when needed, appropriately updated. Fourth, are the coalitions to be allowed to overlap, so that an agent may simultaneously belong to two or more coalitions? These and other challenges related to autonomous agents forming coalitions have been extensively studied in the literature on multi-agent systems, e.g., [8, 10, 11, 14, 15, 29]. They have also arisen in our own recent work on parametric models and a scalable simulation of the large scale ($10^3 - 10^4$ agents) ensembles of autonomous unmanned vehicles on a multi-task mission [5, 6, 19, 20].

Herein, we restrict our attention to the second issue above. We propose a particular mechanism (distributed algorithm) for an effective coalition formation that ensembles of autonomous agents can use as one of their basic coordination subroutines. A need for a dynamic, fully distributed, efficient and online coalition formation may arise due to a number of different factors, such as geographical dispersion of the agents, heterogeneity of tasks and their resource requirements, heterogeneity of agents' capabilities, and so on [20]. While for the small- and medium-scale systems of, e.g., robots or unmanned vehicles, a fully or partially centralized approach to coalition formation and maintenance may be feasible, the large scale systems (with the number of agents of orders of magnitude $10^3 - 10^4$ or higher) appear to necessitate a genuinely distributed approach.

The proposed algorithm is a graph algorithm. The underlying undirected graph captures the communication network topology among the agents. Each agent is a node in the graph. As for the edges, the necessary requirement for an edge between two nodes to exist is that the two nodes be able to directly communicate with one another. That is, an unordered pair of nodes $\{A, B\}$ is an edge of the underlying graph if and only if A can communicate messages to B , or B can communicate messages to A , or both.

The basic idea is to efficiently and in a fully decentralized manner partition this graph into (preferably, *maximal cliques*) of nodes. These coalitions are then maintained until they are no longer useful or meaningful. For instance, the coalitions should be transformed, or else simply dissolved, when the interconnection topology considerably changes, either due to the agents' mobility, or because many old links have died out and perhaps many new, different links have formed, and the like. Another possible reason to abandon the existing coalition structure is when the agents determine that the coalitions have accomplished the set of tasks that these coalitions were formed to address. Thus, in an actual *MAS* application, the proposed coalition formation algorithm may need to be invoked a number of times as a coordination subroutine.

The rest of the paper is organized as follows. The preliminaries are covered in Section 2. We include in this section some examples of large-scale multi-agent systems that are characterized by the *sparse communication network topologies*, as well as a brief overview of the most relevant related work. In Section 3, we succinctly state the problem addressed, the approach taken, and the critical assumptions that need to hold in order for our approach to be applicable. Section 4 is the central part of the paper: we first outline our *Maximal Clique-based Distributed Coalition Formation (MCDCF)* algorithm, and then provide a simple yet illustrative toy-size example of how

the algorithm works. Section 5 includes an outline of the cost analysis of the algorithm, and some discussion. Finally, we summarize in Section 6.

2 Coalition Formation in Large-Scale Multi-agent Systems

Large ensembles of autonomous agents provide an important class of examples where the agents' capability to coordinate and, in particular, to self-organize into groups or coalitions, is often of utmost importance.

We propose herewith a distributed coalition formation algorithm based on the idea that, in peer-to-peer (in particular, *leaderless*) *MAS*, an agent would prefer to form a coalition with those agents that it can communicate with directly, and, moreover, where every member of such a potential coalition can communicate with any other member directly. That is, the preferable coalitions are (*maximal*) *cliques*. It is well-known that finding a maximal clique in an arbitrary graph is **NP**-hard in the centralized setting [3, 4]. This implies the computational hardness that, in general, each node faces when trying to determine the maximal clique(s) it belongs to. However, if the degree of a node is sufficiently small, then finding all maximal cliques this node belongs to may become computationally feasible. If one cannot guarantee that, or *a priori* does not know if, all the nodes in a given underlying *MAS* interconnection topology are of a small degree, then one has to impose additional constraints in order to ensure that the agents are not attempting to solve an infeasible problem.

We describe our distributed maximal clique based coalition formation algorithm in Section 4. Variations of this basic algorithm can be designed to meet the needs of various types of agents, such as, to give some examples, the following:

- the classical cooperative *distributed problem solving* (*DPS*) agents [8, 9, 14, 15];
- different kinds of self-interested, strictly competing or competing-and-cooperating agents [11, 20] where concepts, paradigms and tools from the *N-person game theory* have found many applications; and, more generally,
- various bounded-resource, imperfect-knowledge agents acting in complex environments [20, 26] that are only *partially accessible* to any agent; such autonomous agents are thus characterized by the *bounded rationality* [16].

One may ask, why would this, maximal-clique based approach be promising for the very large scale (or *massive*) multi-agent systems (*MMAS*) that may contain ensembles of anywhere from thousands to millions of agents? The underlying network of such *MMAS* is bound to be very large, thus rendering even many typically feasible (i.e., polynomial-time in the number of agents) graph algorithms obsolete due to their prohibitive cost - let alone allowing distributed coordination strategies that are based on the graph theoretic algorithms that are, in the centralized setting, **NP**-hard in general.

However, there is one critical observation that saves the day of our approach: even if the underlying graph is indeed very large, in many important *MMAS* applications this graph will also tend to be *very sparse*. That is, a typical node in such a graph will tend to have only a handful of neighbors. Therefore, a distributed algorithm where agents reason and communicate strictly locally, where no *flooding* of the network is

ever performed (or needed), and where each agent needs to store and work with only the data pertaining to its near-by agents, can still be designed to be sufficiently efficient.

Some examples of the engineering, socio-technical and socio-economic systems and infrastructures that can be modeled as *MMAS* and that are also characterized by the aforementioned *sparseness* of the underlying network topology, include the following:

(i) *Large-scale* ($10^3 - 10^4$) *ensembles of micro-UAVs* or other similar autonomous unmanned vehicles deployed, for example, in a surveillance or a search-and-rescue mission over a sizable geographic area. Unlike the scenarios where dozens of *macro UAVs* are deployed, where a centralized control and/or one human operator per UAV are affordable and perhaps the most efficient and robust way of deployment, in a very large scale system of autonomous *micro UAVs* no central control is feasible or even possible, and the run-time human intervention is either minimal or nonexistent. Such micro-UAV ensembles need to be able to coordinate, self-organize, and self-adapt to the changing environments in a truly decentralized, dynamic and autonomous manner. For more on the design and simulation challenges of such large-scale ensembles of micro-UAVs, see [5, 19, 20].

(ii) *Smart sensor networks* that include anywhere from thousands to millions of tiny sensors, each of which often of only a few millimeters in size, and of a rather limited computational power and communication range and bandwidth. Such smart sensors usually communicate via *local broadcasts* with very limited ranges. The main “communication mode” of the agents in our algorithm in *Section 4* are precisely the local broadcasts. Due to their small memory capacities and low power consumption requirements, smart sensors need to simultaneously minimize both the amount of local processing, and the amount of communication.

(iii) Various *social networks*, and, in particular, various variants of the ‘*small-world*’ *networks* where, in addition to the strictly local connectivity in the communication network topology, a relatively few long-range connections are also present [23, 24]. A typical node in such a network will have only a handful of neighbors it can directly communicate with, and, moreover, most or nearly all of these neighbors in the network will also tend to be the neighbors in the usual, physical proximity sense.

(iv) Various *socio-technical infrastructures*, such as, e.g., various transportation systems, power grids, etc. An ambitious project on realistic, large-scale modeling and simulation of infrastructures such as the city traffic systems, called *TRANSIMS*, is described at [22] and in the documents found therein.

2.1 Related Work

A variety of coalition formation mechanisms have been proposed in the *MAS* literature both in the context of DPS agents that are all sharing the same goal (as, e.g., in [15]) and in the context of self-interested agents where each agent has its own individual agenda (as, e.g., in [14, 29]). In particular, the problem of distributed task or resource allocation, and how is this task allocation coupled to what coalition structures are most desirable in a given scenario [8, 15], are also of central importance in our own work on a concrete *MAS* application, namely, a scalable parametric model and software simulation of unmanned aerial vehicles (UAVs) that are residing and acting in bounded resource multi-task environments [5, 19, 20].

Another body of MAS literature highly relevant to the distributed coalition formation and task and/or resource allocation, casts the distributed resource allocation problems into the distributed constraint satisfaction and/or optimization (DCS/DCO) terms [8, 9, 10]. Of a particular relevance to our work herein and other possible extensions of the original maximal clique based coalition formation algorithm presented in [21] are the references [15] and [9]. While Modi et al. in [9] offer the most complete formalization of various distributed resource and/or task allocation problems, as well as general mappings to the appropriate types of (*dynamic*) *distributed constraint satisfaction or optimization* problems, their approach is not suitable for a direct application to our modeling framework of massively multi-agent systems in general (see, e.g., [20]), and the application domains we had in mind when devising the algorithm presented herein, in particular. Namely, the agents in [9] are strictly cooperative, share the same goals, and, as such, are not endowed with any notion of individual utilities or preferences. While we have studied cooperative MAS in [20] and elsewhere, as well, one of our main assumptions is that, due to a large scale of the system and a high dynamism and unpredictability of the changes in the environment, no shared or global knowledge about the environment is maintained. In particular, each agent has its own individual preferences over the possible (local) states of the world. The collaboration is then achieved through “encoding” incentives into the individual agents’ *individual behavior functions* [19], and thus using the *incentive engineering* approach [2] to enable the agents to cooperatively coordinate with one another.

The importance of DCS in MAS in general is discussed, e.g., in [28]. However, further discussion of DCS based approaches to distributed resource or task allocation and coalition formation is beyond the scope of this paper.

3 Problem Statement and Main Assumptions

The main purpose of this work is a fully distributed, scalable and efficient algorithm for ensembles of autonomous agents to use as a subroutine within their coordination strategy, with the purpose of efficiently forming temporary coalitions of modest sizes.

Each agent is equipped with a tuple of its internal resources or *capabilities* [15]. Each entry in the capability tuple of any agent is a nonnegative real number. Likewise, each task requires a certain nonnegative amount of each of the individual resources from this tuple in order to be serviced. A single agent, or a coalition of two or more agents, can service a given task if and only if their joint capabilities suffice with respect to that task’s resource consumption requirements. That is, for each component of the capability vector, the sum of the corresponding values taken over all the agents in the coalition has to be greater than, or equal to, the value of the corresponding component of the chosen task’s resource consumption vector.

Our distributed maximal clique based coalition formation algorithm is described in the next section. For this algorithm to be applicable, the following basic assumptions need to hold:

- Agents communicate with one another by exchanging messages either via local broadcasts, or in a peer-to-peer fashion.
- Communication bandwidth availability is assumed sufficient.
- Each agent has a sufficient local memory (including the message buffers) for storing all the information received from other agents.
- Communication is reliable during the coalition formation, in the following sense: if an agent, A , sends a message to another agent B , then either agent B gets *exactly* the same message that A has sent, or else the communication has failed, so that B does not receive anything from A at all. In particular, we assume no *scrambled* or otherwise modified messages are ever received by any agent. Of course, once the coalitions have been already formed, the above assumption on communication reliability need no longer hold¹.
- Each agent has a unique global identifier, 'UID', and the agent knows its UID.
- There is a total ordering, $<$, on the set of UIDs, and each agent knows this ordering.
- Each agent has, or else can efficiently obtain, a reliable knowledge of which other agents are within its communication range.
- The *veracity* assumption holds, i.e., an agent can trust the information received from the neighboring agents.

On the other hand, an agent need not *a priori* know the UIDs of any of the other agents, or, indeed, how many other agents are present in the system at any time.

4 MCDCF Algorithm

After the preliminaries and the clear statement of the problem addressed and the assumptions made, we now present, analyze and discuss our distributed coalition formation algorithm. The *Maximal Clique based Distributed Coalition Formation (MCDCF)* algorithm will be presented in subsection 4.1. An example of a simple network of agents, and how the algorithm works when applied to this network, is given in subsection 4.2. An outline of the algorithm's cost analysis will follow in Section 5.

4.1 Algorithm Description

We approach distributed coalition formation for task allocation as follows. The candidate coalitions are required to be cliques of uniformly bounded sizes. That is, the system designer, based on the application at hand and the available system resources (local computational capabilities of each agent, bandwidth of the agent-to-agent communication links, etc.), *a priori* chooses a threshold, $K = K(n)$, such that only the coalitions of sizes up to K are considered. Agents themselves subsequently form coalitions in a fully distributed and online manner, as follows. Each agent (i) first learns of

¹ As this requirement is still restrictive, and considerably limits the robustness of our algorithm, we will try to relax this assumption in our future work, and enable the agents to effectively form coalitions even in the presence of some limited amount of communication noise during the coalition formation process itself.

who are its neighbors, then (ii) determines the appropriate *candidate coalitions*, that the agent hopes are (preferably maximal, but certainly of sizes bounded by K) cliques that it belongs to, then (iii) evaluates the utility value of each such candidate coalition, measured in terms of the joint resources of all the potential coalition members, then (iv) chooses the most desirable candidate coalition, and, finally, (v) sends this choice to all its neighbors. This basic procedure is then repeated, together with all agents updating their knowledge of (a) what are the preferred coalitions of their neighbors, and (b) what coalitions have already been formed.

In addition to its globally unique identifier UID, which we assume is a positive integer, and the vector of capabilities, each agent also has two local flags that it uses in communication with other agents. One of the flags is the binary “decision flag”, which indicates whether or not this agent has already joined some coalition. Namely, $decision \in \{0, 1\}$, and the value of this flag is 0 as long as the agent still has not irrevocably committed to what coalition it is joining. The second flag is the “choice flag”, which is used to indicate to other agents, how happy is the agent with its current tentative choice or proposal of the coalition to be formed. That is, the choice flag indicates the level of an agent’s urgency that its proposal for a particular coalition to be formed be accepted by the neighbors to whom this proposal is being sent. If an agent v_i sends to its neighbors the choice flag value $choice(i) = 0$, that means that this agent has no satisfactory alternatives to its currently proposed coalition. The choice flag value of 1 indicates that an agent can afford to change its coalition choice, but that each of the available alternative coalitions is strictly less preferred than the current proposal. Finally, $choice(i) = 2$ indicates that agent v_i has alternative choices that are of equal preference as the currently proposed coalition.

We remark that any *candidate coalition*, that is, a subset of the set of all neighbors of an agent, such that the agent currently considers this subset to be a possible choice of the coalition this agent would like to form, need not be a clique, let alone a *maximal clique*. Indeed, based on its strictly local knowledge, the agent in general does not know which of its candidate coalitions are cliques, if any. However, only those candidate coalitions that indeed *are cliques* will ever be agreed upon by the participating agents, and therefore have a chance of possibly becoming the *actual* coalitions. This observation justifies the name of our algorithm.

We split the *MCDCF* algorithm into six stages. Four of these six are iteratively repeated until the consensus on coalition formation is reached. We point out, however, that *each agent executes these stages asynchronously and in parallel with the other agents*. The only assumption about the synchronization among the agents is that an agent does not begin another iteration of *Stages 2 - 5* before its neighbors are done with the previous iteration. Should an agent fail to receive the update from one of its neighbors within the pre-specified time slot, the agent assumes that its neighbor is no longer available for the coalition formation, and deletes this neighbor’s UID from all the appropriate lists. In the sequel, we won’t bother distinguishing between an agent or a communication network node, v_i , and this agent’s (alternatively, node’s) UID, i ; the intended meaning in any given situation will be clear from the context.

Stage 0: Each agent, asynchronously and in parallel with all other agents, broadcasts a four-tuple to all its immediate neighbors. The entries in this tuple are (i) the agent’s

UID, (ii) the agent's list of immediate neighbors, $L(i)$, that includes i , (iii) the value of the choice flag that indicates that the list sent is the neighborhood list, and (iv) the value of the decision flag. Each agent also receives the corresponding tuples from all of its neighbors. Those neighbors whose messages have not been received within the allotted time are discarded from the future coalition considerations.

Stage 1: Each agent i locally computes the overlaps of its neighborhood list with the neighborhood lists that it has received from its neighbors, $C(i, j) \leftarrow L(i) \cap L(j)$. These list intersections are then ordered with respect to the list size.

Each agent repeats *Stages 2 - 5* until it either reaches a consensus on what coalition it is joining, or else is left with no choice but to form the trivial single-member coalition.

Stage 2: Agent i looks for information from its neighbors, whether they have joined a coalition "for good" during the previous round. Those neighbors that have are deleted from the neighborhood list $L(i)$; the intersection lists $C(i, j)$ and the candidate coalition lists $C(i)$ are also updated accordingly, and those $C(i, k)$ for which k is deleted from the neighborhood list $L(i)$ are also deleted. Likewise, the coalition values $val[C(i, k)]$ are updated as appropriate.

Stage 3: Agent i picks one of the most preferable lists $C(i, j)$; let $C(i) \leftarrow chosen [C(i, j)]$. If the group or coalition size is the main criterion, then one of the lists of maximal length is chosen. If the combined *capabilities* of each tentative coalition for servicing various tasks are the main criterion, then each agent first evaluates or estimates the *coalition value* with respect to its local knowledge of the existing tasks and their demands in terms of the coalitional capabilities. To evaluate these coalition values, agent i needs to obtain information about other, near-by agents' capabilities. The agent then orders possible future coalitions based on these estimated coalition values, and picks as its current coalition proposal one of the possible coalitions with the highest coalition value. Since the assumption is that the capability vector of each agent has all entries nonnegative, this *monotonicity property* ensures that no proper subset of a candidate maximal clique coalition is ever chosen - except in the cases when the clique size exceeds the threshold, $K(n)$.

Stage 4: Each agent sends its tuple with its UID, the tentatively chosen list $C(i)$, the value of the choice flag, and the value of the decision flag, to all its neighbors. Likewise, each agent receives the corresponding 4-tuples from its current neighbors.

Stage 5: Agent i compares its chosen list $C(i)$ with the lists $C(j)$ received from its neighbors. If a satisfactory clique that includes the node i exists, and all members of this clique have selected it at this iteration as their current coalition of choice (that is, if $C(i) = C(j)$ for all $j \in C(i)$), this will be efficiently recognized by the agents that are forming this particular clique. The decision flag of each agent $j : j \in C(i)$ is then set to 1, the coalition is formed, and this information is broadcast to all of the neighbors. In particular, agent i locally broadcasts its agreed-upon coalition, and decision flag $decision(i) = 1$, to all of its still remaining neighbors. Else, if no such agreement is reached, then agent i , based on its UID and priority, and its current value of the *choice flag*, either does nothing, or else changes its mind about its current coalition of choice, $C(i)$.

Each agent uses *time-outs* in order to place an upper bound on for how long it may be waiting to hear from any other agent during any stage of the algorithm. If agent v_p has sent a message to agent v_q , and the latter is not responding, then there are four possibilities: (i) agent v_q has failed; (ii) the communication link from v_q to v_p has failed; (iii) while v_q is in v_p 's communication range, the converse does not hold (but v_p may not know it), and (iv) either the agent v_q , or the communication link from v_q to v_p , is too slow. In each case, once v_p has waited sufficiently long to hear from v_q , v_p will simply consider v_q unavailable for the joint coalition formation, and will delete v_q from its candidate coalition lists. Thus, case (iv) will be treated by agent v_p in exactly the same way as the other three cases.

In order to ensure that the algorithm avoids cycling in every possible scenario, once an agent, v_i , changes its mind about the preferred coalition $C(i)$, it is not allowed through the remaining rounds of the algorithm to go back to its old choice(s). Once no other choices are left, this particular agent sticks to its current choice, and waits for other agents to settle to their choices. This requirement ensures the ultimate convergence to a coalition structure that all agents (locally) agree on. That is, under the assumptions stated in the previous section, the agents will reach consensus on the coalition structure after a finite number of iterations through the *Stages 2 - 5*.

Once all the agents exit the iterated execution of the *Stages 2 - 5*, each formed coalition will indeed be a clique. Moreover, those agent coalitions whose sizes do not exceed the pre-specified threshold, K , are also maximal in a sense that, given such a coalition C , no agent(s) outside of this coalition can be added to it, so that the following requirements simultaneously all hold: (i) each of the new agents is already adjacent to all the "old" coalition members of C , (ii) if more than one new agent is added, then all the added agents are also pairwise neighbors to each other, (iii) the newly added agent(s) do not already belong to a coalition (or coalitions) that have already been formed, and (iv) the new size of the augmented coalition C is still at most K .

However, it is easy to construct examples of the underlying graphs and the particular runs of the algorithm such that, once every agent joins a coalition and the algorithm terminates, several agents end up in trivial coalitions. It is therefore reasonable, in many application contexts, to introduce an *optional Stage 6* of the algorithm during which some of these small and, therefore, potentially not sufficiently useful coalitions, may be merged together. Thus, if some two small coalitions, or one small and one bigger coalition, are adjacent to each other², they can be merged together. For an illustration, we refer the reader to the worked out example in subsection 4.2.

4.2 How MCDCF Algorithm Works: A Simple Example

To show how the *MCDCF* algorithm works, we use a simple example. The interconnection topology of a group of agents is given in *Figure 1*. For simplicity, we assume that the only value associated with each coalition is the coalition's size. We also assume that no agent falls behind others by too much, i.e., that all agents complete each iteration of *Stages 2 - 5* within the allotted time.

² That is, if there exist node x in the first coalition and node y in the second such that x and y are adjacent in the underlying graph.

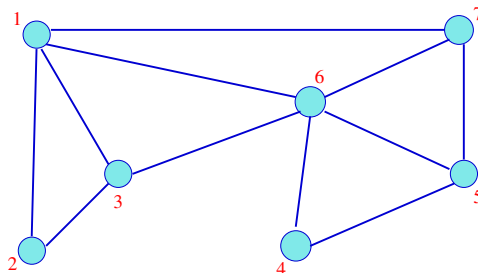


Fig. 1. An example: the agents' starting communication topology

We notice that the largest clique that any agent in this example belongs to is a 3-clique. However, several agents belong to multiple triangles, so this toy example is instructive insofar as how the agents break ties, avoid deadlocks, and reach consensus on the coalition structure.

First, the initialization stage takes place, during which each agent locally broadcasts its list of neighbors to each of its neighboring agents. Then each agent, asynchronously and in parallel with the other agents, forms the initial candidate coalitions by computing the pairwise neighborhood list intersections. However, not all of thereby formed coalitions are *reachable*: if $C(i, k) \subsetneq C(i, j)$ then there exists a (maximal reachable) candidate coalition $C(i, \cdot)$, $C(i, k) \subseteq C(i, \cdot) \subsetneq C(i, j)$, such that for some agent $v_p : p \in C(i, \cdot)$, there exists an agent v_q such that $q \in C(i, j)$, and v_p and v_q are not adjacent to each other. Clearly, this agent v_p will never agree on forming the bigger coalition $C(i, j)$, since this coalition would include at least one agent that v_p cannot directly communicate with. Hence, all such sets $C(i, j)$ that *properly include* other prospective coalitions can be safely deleted from the list of candidate coalitions.

In our example, agent v_1 can safely delete the set $\{1, 3, 6, 7\}$, as this coalition can never be agreed upon by all four agents. Concretely, v_3 will never agree to form a coalition that includes v_7 (and vice versa). Furthermore, based on the information received from v_3 and v_7 , agents v_1 and v_6 can readily and safely *infer* that the coalition $\{1, 3, 6, 7\}$ cannot be agreed on.

Once all such *unreachable coalitions* are deleted, the column denoted “*candidate coalitions* $C(i, j)$ ” in *Table 1* is obtained. After that, each agent picks one of the available choices. Under the *monotonicity assumption* discussed earlier, each agent will select one of the *maximal sets (potential cliques)* that it belongs to. Each agent also appropriately sets the value of its *choice flag*. When the coalition size is the criterion of that coalition's value, agent v_1 , for example, would set its *choice flag* to 2, since it has more choices that are just as preferable as the selected candidate coalition, the set $\{1, 2, 3\}$. Once this step is completed by all agents, the overall configuration is reached as depicted in *Table 1*, where each row represents the corresponding agent's current local knowledge of the neighborhood structure and of its choice of a tentative coalition.

For simplicity, we have assumed in this example that, in case of a tie, each agent picks the lexicographically lowest coalition. Hence, the agents v_1, v_2 and v_3 immediately reach the agreement on forming the coalition $\{1, 2, 3\}$. The other four agents, however, do not reach an agreement after the first iteration. Let us assume that, among sev-

Table 1. State of each agent v_i at the end of first iteration

Node	neighborhood $L(i)$	nbhd. overlaps $L(i) \cap L(j)$	candidate coalitions $C(i, j)$	chosen $C(i)$	choice flag
v_1	{1, 2, 3, 6, 7}	{1, 2, 3}, {1, 2, 3, 6}, {1, 3, 6, 7}, {1, 6, 7}	{1, 2, 3}, {1, 3, 6}, {1, 6, 7}	{1, 2, 3}	2
v_2	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	0
v_3	{1, 2, 3, 6}	{1, 2, 3, 6}, {1, 2, 3}, {1, 3, 6}	{1, 2, 3}, {1, 3, 6}	{1, 2, 3}	2
v_4	{4, 5, 6}	{4, 5, 6}	{4, 5, 6}	{4, 5, 6}	0
v_5	{4, 5, 6, 7}	{4, 5, 6}, {4, 5, 6, 7}, {5, 6, 7}	{4, 5, 6}, {5, 6, 7}	{4, 5, 6}	2
v_6	{1, 3, 4, 5, 6, 7}	{1, 3, 6}, {1, 3, 6, 7}, {4, 5, 6}, {1, 5, 6, 7}, {4, 5, 6, 7},	{1, 3, 6}, {1, 6, 7}, {4, 5, 6}, {5, 6, 7}	{1, 3, 6}	2
v_7	{1, 5, 6, 7}	{1, 6, 7}, {1, 5, 6, 7}, {5, 6, 7}	{1, 6, 7}, {5, 6, 7}	{1, 6, 7}	2

eral agents with the same value of the choice flag, $choice > 0$, the ones with the lowest index among their neighbors are required to change their proposed coalitions. In our example, this means that, in the second iteration, v_5 drops {4, 5, 6} and selects {5, 6, 7} instead. Upon doing so, agent v_5 also adjusts its value of the choice flag, and broadcasts these changes to agents v_4, v_6 and v_7 . Also, since agents v_1 and v_3 are no longer available, agents v_6 and v_7 delete v_1, v_3 from their neighborhood lists, and update the candidate coalitions accordingly.

After each of the agents v_5, v_6 and v_7 performs the described updates, and then locally broadcasts its new configuration to all of the remaining neighbors, the overall configuration at the end of the second round is as in *Table 2*.

Table 2. Coalition configuration at the end of the second iteration

Node	candidate coalitions $C(i, j)$	chosen $C(i)$	choice flag	decision status
v_1	{1, 2, 3}, {1, 3, 6}, {1, 6, 7}	{ 1, 2, 3 }	...	done
v_2	{1, 2, 3}	{ 1, 2, 3 }	...	done
v_3	{1, 2, 3}, {1, 3, 6}	{ 1, 2, 3 }	...	done
v_4	{4, 5, 6}	{4, 5, 6}	0	busy
v_5	{5, 6, 7}	{5, 6, 7}	0	busy
v_6	{4, 5, 6}, {5, 6, 7}	{4, 5, 6}	2	busy
v_7	{5, 6, 7}	{5, 6, 7}	0	busy

After the second round is completed, the only agent that still has some “maneuvering room” is v_6 ; since there is still no agreement, and $choice(v_6) > 0$, whereas $choice(i) = 0$ for all $i \neq 6$ such that v_i is not done yet, in the next round v_6 changes its coalition proposal to {5, 6, 7}, thereby reaching the consensus with v_5 and v_7 , and yielding the final configuration as in *Table 3*.

Table 3. The final coalition configuration; the only unhappy node is v_4

Node	<i>candidate coalitions $C(i, j)$ at the time of agreement</i>	<i>chosen $C(i)$</i>	<i>choice flag</i>	<i>decision status</i>
v_1	$\{1, 2, 3\}, \{1, 3, 6\}, \{1, 6, 7\}$	$\{1, 2, 3\}$...	done
v_2	$\{1, 2, 3\}$	$\{1, 2, 3\}$...	done
v_3	$\{1, 2, 3\}, \{1, 3, 6\}$	$\{1, 2, 3\}$...	done
v_4	$\{4\}$	$\{4\}$	0	doomed
v_5	$\{5, 6, 7\}$	$\{5, 6, 7\}$	0	done
v_6	$\{5, 6, 7\}$	$\{5, 6, 7\}$	0	done
v_7	$\{5, 6, 7\}$	$\{5, 6, 7\}$	0	done

Thus, in the end, each of the agents, except for v_4 , has joined (one of) the coalition(s) optimal for it. Since agent v_4 has ended up left out, and since it is adjacent to the coalition $\{v_5, v_6, v_7\}$, it can be merged with this coalition to form a larger coalition $\{v_4, v_5, v_6, v_7\}$ in the optional *Stage 6* (see discussion in subsection 4.1). Given the assumed *monotonicity* and (locally constrained) *super-additivity* of the multi-agent environment, any coalition arising from such a merger clearly cannot be a clique.

5 Algorithm Cost Analysis and Discussion

We now outline the cost analysis of the *MCDCF* algorithm. We will focus on the main resource requirements *per agent*. We analyze under what assumptions will the required amounts of computation time, memory storage and communication be feasible so that an agent would want to venture into participating in the coalition formation based on our algorithm. However, we do not address the issues of, e.g., communication reliability, network delays, and similar.

The first and foremost cost requirement is that the algorithm be of a feasible computational complexity when it comes to its overall worst-case running time. At the very least, this running time needs to be *polynomial in the number of agents*, n . Moreover, for *MMAS* of up to 10^6 agents, the upper bound on the total number of elementary computational steps better be a polynomial of a low degree. We show that, under a restrictive yet reasonable assumption on the sparseness of the underlying network of agents, this goal can be attained. Due to the space constraints, we limit our complexity analysis to that pertaining to *the execution time*, i.e., to the number of elementary computation steps carried out by each agent. We will assume that the time to send and receive messages is not increasing the asymptotic upper bounds on local computations. Since the amount of data that the agents locally broadcast in the algorithm is fairly small (namely, *linear* in the size of the largest list used by an agent), our assumption boils down to assuming a sufficient available bandwidth, sufficiently big buffers for the arriving messages, and no excessive network delays.

We shall split the time complexity analysis into two parts. First, we will estimate the maximum number of rounds, i.e., how many times an agent may need to iterate

through the *Stages 2 - 5* (see subsection 4.1). Second, we will show that the amount of computation per agent within a single iteration is relatively small.

Let $K = K(n)$ be a nonnegative, monotonically nondecreasing function of n , and let the class of the underlying graphs of agents be such that, for any positive integer n (except possibly for the first $O(1)$ of them), the size of any clique in the graph is bounded by $K(n)$. To show under what conditions is our algorithm going to iterate at most polynomially many times, we establish the following result:

Proposition 1. *Let an undirected graph with n nodes be such that the bound on the maximum node degree in this graph is given by $K(n) \leq c \cdot \log n$, for some positive constant c . Let's assume each node in the graph is an agent with sufficient computational and communication resources. Then the maximum number of iterations of the algorithm described in Section 4, when executed by the agents, will be polynomial in the number of agents, n .*

Proof. Let $K(n) \leq c \cdot \log n$. Let v_i be an arbitrary agent. Since v_i has at most $K(n)$ neighbors, the maximum number of candidate coalitions that would include agent v_i is at most $2^{K(n)+1}$, which is bounded from above by $2 \cdot 2^{c \log n} = 2n^c$. Since, during a single round of the algorithm, at least one agent has to change its current choice of the proposed coalition, and since the old choice is permanently discarded, at each round the total number of the remaining candidate coalitions in the entire system is reduced by at least one. Since there are at most $n \cdot 2^{K(n)+1} \leq 2n \cdot n^c$ possible coalitions at the beginning, the total number of rounds is $O(n^{c+1})$.

Hence, if the underlying network topology of a *MAS* is such that $K(n) = O(\log n)$, then our algorithm will run in time polynomial in n . Moreover, if $K(n) \leq c \log n$ holds for $c = 1$, then the number of rounds is at most quadratic in n . We shall assume that the bound $K(n) \leq c \log n$ holds for some positive, real constant c close to 1. We shall also assume that, whatever the criteria of “goodness” for these clique-based coalitions may be, given the necessary data about its neighbors, an agent can efficiently compute the candidate coalition’s value, $val[C(i)]$, for any such potential coalition $C(i)$. In particular, we shall assume in the sequel that, for any agent v_i , a single evaluation or value estimation of any coalition $C(i)$ of size at most m takes $O(m^2)$ steps.

Proposition 2. *Let the assumptions from the discussion above hold. Then the amount of local computation that any agent has to perform during any iteration of our MCDCF algorithm is polynomial in the size of the data structures involved (cf. lists $L(i)$, $C(i)$ and $C(i, j)$). In particular, assuming that the encoding of all information about a single agent (its *UID*, list of available resources, etc.) is bounded by $O(\log n)$, the total number of elementary bitwise operations of our algorithm is bounded by $O((\log n)^4)$.*

Proof. Under the stated assumptions, $K(n) \leq c \cdot \log n$, and therefore for any agent v_i , any of the lists $C(i)$, $L(i)$, $C(i, j)$ that this agent operates with are also of sizes bounded by $O(\log n)$. Since *Stages 0 - 1* are executed only once, and *Stage 4* includes communication only, we just need to estimate the amounts of local computation during *Stages 2, 3 and 5*. During *Stages 2 - 3*, a number of operations on individual lists and pairs of lists are performed. Sorting a list with K elements takes time $O(K \log K)$.

Finding an element and deleting it from a list with at most K elements takes time $O(K)$. When both lists are of size $O(K)$, comparing two sorted lists, computing their intersection, or testing if one sorted list is a subset of the other, each takes at most $O(K \cdot \log K)$ operations. An agent v_i may need to perform up to K such pairwise list comparisons, intersections and similar list operations - one for each $j \in L(i)$, where $|L(i)| \leq K$. Each of these operations is done at the granularity level of a single list element, which we have assumed is encoded by $O(\log n)$ bits. Hence, the total number of list operations at the bit level is bounded by $O(K^2 \log K \log n)$, which, given our assumptions, is just $O((\log n)^3 (\log \log n))$.

Let's assume that the time to evaluate any $val[C(i)]$ is bounded by some function $T(m)$, where $m = |C(i)|$ is the size of coalition $C(i)$. Let's also assume that, at each node, the candidate coalitions are *sorted* in a non-increasing order with respect to their values $val[C(i)]$. Then evaluating the coalitional values takes at most $O(\log n) \cdot (T(c \cdot \log n) + c \cdot \log(c \log n) + O(1))$ elementary steps. When $T(m) = O(m^2)$, this simplifies to $O((\log n)^3)$. Since each step is assumed to require no more than $O(\log n)$ bit operations, we arrive at $O((\log n)^4)$ bitwise operations overall. As updating and maintaining the coalition values during the subsequent iterations is no costlier than originally computing them from the scratch at the first iteration, the upper bound of $O((\log n)^4)$ remains valid.

Similar analysis applies to *Stage 5*, where the costliest operations are the pairwise list comparisons. Thus *Stages 2 - 5* together take the number of elementary bitwise steps per iteration that is $O((\log n)^4)$.

5.1 Discussion

The proposed distributed coalition formation algorithm is based on two main ideas. One idea, familiar from the literature (see, e.g., [15] and references therein), is to formulate a distributed task and/or resource allocation problem as a (*distributed*) *set covering problem*, (*D*)*SC*, in those scenarios where the coalition overlaps are allowed, or a (*distributed*) *set partitioning problem*, (*D*)*SP*, when no coalition overlaps are allowed. Two (or more) coalitions overlap if there exists an element that belongs to both (all) of them. It is well-known that the decision versions of the classical, centralized versions of the *SC* and *SP* problems are **NP**-complete [4]. Consequently, what is needed are efficient distributed heuristics so that the agents can effectively apply *DSC*- or *DSP*-based strategies for coalition formation. Fortunately, some such efficient heuristics are already readily available [15].

The second main idea is to ensure that the formed coalitions of agents meet the robustness and fault tolerance criteria, which are particularly important in applications where there is a high probability of the subsequent node and/or communication link failures. The most robust coalitions of agents of a particular size are those that correspond to *maximal cliques* in the underlying interconnection topology of the agents' communication network. Alas, the *Maximal Clique* problem is also well-known to be **NP**-hard [3, 4]. This hardness stems from the fact that an agent may need to test for the "cliqueness" exponentially many candidate subsets that it belongs to. However, in those graphs where the maximum degree of each node is bounded by $c \log n$, the number of subsets that each node belongs to is $O(n^c)$, i.e., *polynomial in the total number*

of agents. Moreover, in those graphs where the node degrees are uniformly bounded by some (small) constant, $K = O(1)$, since 2^K is presumably still sufficiently small, finding maximal cliques becomes not only tractable in theory (i.e., solvable in polynomial time) but also practically feasible in the online, real-time and bounded-resource scenarios that are of the main interest in many *MMAS* applications.

What are the main properties of the coalition structures likely to arise when our algorithm is invoked on an arbitrary large but sufficiently sparse graph that satisfies the aforementioned assumptions? Once the coalitions are formed according to the algorithm, these coalitions of agents will be tight (as everyone in the coalition can communicate with everyone else *directly*), and therefore as robust and fault-tolerant as possible. This is a highly desirable property involving the coalitions or teams of agents operating in the environments where both the agent failures and the agent-to-agent communication link failures can be expected once these agent coalitions are deployed to service their appropriate tasks. One example of such a *MMAS* application domain and, in particular, some coordination strategies in that domain, are studied in [5, 6, 19, 20].

The proposed algorithm can be used as a subroutine in many multi-agent system scenarios where, at various points in time, the system needs to reconfigure itself, and the agents need to form new coalitions (or transform the existing ones) in a fully distributed manner, where each agent has to reason, act and coordinate with other agents *strictly locally*, and where it is important for the agents to be able reach consensus on these coalitions efficiently.

Our final observation is that the proposed algorithm can be expected to be useful only when the time scale of significant changes in the inter-agent communication topology is much coarser than the time scale for the coalitions of agents, first, to form according to the algorithm, and, second, once formed, to accomplish something useful in terms of the agents' ultimate goals (see, e.g., [20, 21]).

6 Conclusion

We propose in this paper an algorithm for distributed coalition formation based on a distributed computation of (maximal) cliques of modest sizes in the underlying communication network of agents. We hope that this algorithm, or its appropriately fine-tuned variants, will turn out to be a potentially useful subroutine in many multi-agent system applications, where the interconnection topology of the agents may often change, so that the system needs to dynamically reconfigure itself *repeatedly*, yet where these topology changes are at a time scale that allows agents to (i) form their coalitions, and (ii) do something useful while participating in such coalitions, before the underlying communication topology of the system changes so much as to render the formed coalitions either obsolete or ineffective.

As for the future work, we plan a detailed comparative analysis of the approach presented herein on one, and the well-known coalition formation approaches from the MAS literature, on the other hand. In particular, we would like to compare and contrast the purely peer-to-peer, genuinely “democratic” approaches to multi-agent coordination, where *all agents are made equal* (except possibly for the different capability vectors), with the asymmetric, less democratic and more leader-based coordination ap-

proaches (such as, e.g., various automated dynamic auctions). Intuitively, the genuinely leaderless mechanisms for coalition formation, such as our maximal clique based approach, are less prone to “bottlenecks” and single points of failure than the coordination strategies where certain agents are given special roles or the “leader” status. However, this intuition needs to be both further theoretically investigated and experimentally validated via appropriate comparative simulations and performance measurements.

Acknowledgment. Many thanks to Myeong-wuk Jang, Nirman Kumar and Reza Ziaei for many useful discussions. This work was supported in part by the *DARPA IPTO TASK Program*, contract # *F30602-00-2-0586*. The first author would also like to acknowledge the travel grant from the MMAS’04 conference organizers.

References

1. N. M. Avouris, L. Gasser (eds.), “Distributed Artificial Intelligence: Theory and Praxis”, Euro Courses Comp. & Info. Sci. vol. 5, Kluwer Academic Publ., 1992
2. D. H. Cansever, “Incentive Control Strategies For Decision Problems With Parametric Uncertainties”, Ph.D. thesis, Univ. of Illinois Urbana-Champaign, 1985
3. T. H. Cormen, C. E. Leiserson, R. L. Rivest, “Introduction to Algorithms”, MIT Press, 1990
4. M. R. Garey, D. S. Johnson, “Computers and Intractability: a Guide to the Theory of NP-completeness”, W. H. Freedman & Co., New York, 1979
5. M. Jang, S. Reddy, P. Tosic, L. Chen, G. Agha, “An Actor-based Simulation for Studying UAV Coordination”, Proc. 15th Euro. Symp. Simul. (ESS ’03), Delft, Holland, 2003
6. M. Jang, G. Agha, “On Efficient Communication and Service Agent Discovery in Multi-agent Systems,” 3rd Int’l Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS ’04), pp. 27-33, May 24-25, Edinburgh, Scotland, 2004
7. N. Lynch, “Distributed Algorithms”, Morgan Kaufmann Publ., Wonderland, 1996
8. P. J. Modi, H. Jung, W. Shen, M. Tambe, S. Kulkarni, “A dynamic distributed constraint satisfaction approach to resource allocation”, in Proc. 7th Int’l Conf. on Principles & Practice of Constraint Programming, 2001
9. P. J. Modi, H. Jung, W. Shen, “Distributed Resource Allocation: Formalization, Complexity Results and Mappings to Distributed CSPs”, technical report, Nov. 2002
10. P. J. Modi, W. Shen, M. Tambe, M. Yokoo, “An asynchronous complete method for distributed constraint optimization”, Proc. 2nd AAMAS ’03, Melbourne, Australia, 2003
11. J. Rosenschein, G. Zlotkin, “Rules of Encounter: Designing Conventions for Automated Negotiations among Computers”, The MIT Press, Cambridge, Massachusetts, 1994
12. T. Sandholm and V. Lesser, “Issues in automated negotiation and electronic commerce: Extending the contract net framework”, in 1st Int’l Conf. on Multiagent Systems, pp. 328-335, San Francisco, 1995
13. T. Sandholm, V. Lesser, “Coalitions among Computationally Bounded Agents”, *Artificial Intelligence*, spec. issue on “Principles of MAS”, 1997
14. O. Shehory, S. Kraus, “Coalition formation among autonomous agents: Strategies and complexity”, Proc. MAAMAW’93, Neuchatel, Switzerland, 1993
15. O. Shehory, S. Kraus, “Task allocation via coalition formation among autonomous agents”, Proc. 14th IJCAI-95, Montreal, August 1995
16. H. A. Simon, “Models of Man”, J. Willey & Sons, New York, 1957
17. R. G. Smith, “The contract net protocol: high-level communication and control in a distributed problem solver”, IEEE Trans. on Computers, 29 (12), 1980

18. G. Tel, "Introduction to Distributed Algorithms", 2nd ed., Cambridge Univ. Press, 2000
19. P. Tosić, M. Jang, S. Reddy, J. Chia, L. Chen, G. Agha, "Modeling a System of UAVs on a Mission", Proc. SCI '03 (invited session), Orlando, Florida, 2003
20. P. Tosić, G. Agha, "Modeling Agents' Autonomous Decision Making in Multiagent, Multi-task Environments", Proc. 1st Euro. Workshop on MAS (EUMAS '03), Oxford, 2003
21. P. Tosić, G. Agha, "Maximal Clique Based Distributed Group Formation Algorithm for Autonomous Agent Coalitions", Proc. Workshop on Coalitions & Teams, within AAMAS '04, New York City, July 19-23, 2004
22. For more on the *TRANSIMS* project at the Los Alamos National Laboratory, go to <http://www-transims.tsasa.lanl.gov/> (The '*Documents*' link includes a number of papers and technical reports for the period 1995 - 2001)
23. D. J. Watts, "Small Worlds: The Dynamics of Networks Between Order and Randomness", Princeton Univ. Press, Princeton, N. Jersey, 1999
24. D. J. Watts, S. H. Strogatz, "Collective dynamics of 'small-world' networks", *Nature* 393, 1998
25. G. Weiss (ed.), "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence", The MIT Press, Cambridge, Massachusetts, 1999
26. M. Wooldridge, N. Jennings, "Intelligent Agents: Theory and Practice", Knowledge Engin. Rev., 1995
27. M. Yokoo, K. Hirayama, "Algorithms for Distributed Constraint Satisfaction: A review", AAMAS, Vol. 3, No. 2, 2000
28. M. Yokoo, "Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems", Springer, 2001
29. G. Zlotkin, J.S. Rosenschein, "Coalition, cryptography and stability: Mechanisms for coalition formation in task oriented domains", Proc. AAAI'94, Seattle, Washington, 1994