Maximal Clique Based Distributed Group Formation For Task Allocation in Large-Scale Multi-Agent Systems

Predrag T. Tosic 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@cs.uiuc.edu, agha@cs.uiuc.edu

Abstract. *We present a fully distributed algorithm for group or coalition formation among autonomous agents. The algorithm is based on two main ideas. One is a distributed computation of maximal cliques (of up to pre-specified size) in the underlying graph that captures the interconnection topology of the agents. Hence, given the current configuration of the agents, the groups that are formed are characterized by a high degree of connectivity, and therefore high fault tolerance with respect to node and 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 particular highly desirable task(s) are preferable to coalitions with either insufficient resources, or with resources that suffice only for completing less valuable tasks. We envision variants of our distributed algorithm 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

Autonomous agents and *multi-agent systems (MAS)* [1, 4, 29, 30] are characterized, among other properties, by (i) a considerable degree of autonomy of individual computing entities or processes (agents) and, at the same time, the fact that (ii) each agent has a local, that is, in general, incomplete and imperfect "picture of the world". Since in MAS there is either no central control, or at best only a limited central control, and the individual agents have to both think and act locally, genuinely distributed algorithms are needed for the agents to effectively coordinate with one another.

MAS pose a number of challenges to a distributed algorithm designer. Many of the major challenges are related to various aspects of the agent coordination problem (e.g., [1, 29]). In order to be able to effectively coordinate, agents need to be able to *reach consensus (or agreement)* on various matters of common interest. Two particularly prominent distributed consensus problems are those of *leader election* (e.g., [8, 22]) and *group formation.* Group or coalition formation is an important issue in distributed systems in general [8], and MAS in particular [33]. Given a collection of communicating agents, the goal in distributed group formation is that these agents, based on their local knowledge only, decide how to effectively split up into several groups, so that each agent knows which group(s) it belongs to.

There are several critical issues that a MAS designer needs to address in the context of (distributed) group formation. First, what is the right notion of a group in a given setting? Second, a distributed group formation mechanism - that is, a distributed algorithm that enables agents to effectively form groups or coalitions - needs to be provided. Third, groups and each agent's knowledge about its group membership need to be maintained and, when needed, appropriately updated. Fourth, are the groups to be allowed to overlap, so that an agent may simultaneously belong to two or more groups? These and other challenges related to autonomous agents forming coalitions have been extensively studied in the literature on

multi-agent systems, e.g., [9, 11, 14, 18, 19, 33]. They have also arisen in our own recent and ongoing work on parametric models and a scalable simulation of large scale ($10^3 - 10^4$ agents) ensembles of autonomous unmanned vehicles on a multi-task mission [6, 7, 23, 24].

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 group formation may arise due to a number of different factors, such as the geographical dispersion of the agents, heterogeneity of tasks and their resource requirements, heterogeneity of agents' capabilities, and so on [24]. While for small- and medium-scale systems of robots or unmanned vehicles a fully or partially centralized approach to group formation and maintenance may be feasible or even optimal, large scale systems (with the number of agents of orders of magnitude $10^3 - 10^4$ or higher) appear to necessitate a fully distributed approach. That is, the agents need to be able to self-organize into coalitions, and quickly reach a consensus on who is forming a coalition with whom in a fully decentralized manner [25].

2 Group or Coalition Formation in 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 for such systems to be able to accomplish their tasks.

One can distinguish between two general, broad classes of such autonomous agents. One is the class of agents deployed in the context of *distributed problem solving.* The agents encountered in distributed problem solving (DPS) typically share their goal(s). For instance, DPS agents most often have a joint utility function that they all wish to maximize as a team, without any regard to (or perhaps even a notion of) individual payoffs. This joint utility or, more generally, the goal or set of goals assigned to DPS agents, is usually provided by their designer. However, it may not be always feasible - or even possible - that the designer always explicitly specify, for instance, how are the agents to divide-and-conquer their tasks and resources, how are they to form groups and elect leaders of those groups, etc. Due to scalability, incomplete *a priori* knowledge of the environments these agents may encounter, and possibly other considerations, instead of "hard-wiring" into his DPS agents explicitly how are the agents to be coordinated, the system designer may choose only to enable the autonomous agents with the basic coordination primitives, and leave to the agents to self-organize and coordinate as the situation may demand. Hence, in many situations the DPS agents may be required to be able to effectively form groups or coalitions in a fully distributed manner.

The second basic type of agents, the *self-interested agents*, are a kind of agents that do not share their goals (and, indeed, need not share their designer). In contrast to the DPS agents, each self-interested agent has its own agenda (e.g., an individual utility function it is striving to maximize), and no altruistic incentives to cooperate with other agents. However, even such self-interested, goal-driven or individual-utility-driven agents, while in essence selfish, may nonetheless still need to cooperatively coordinate and collaborate with each other in many situations. One class of examples are those agents (such as, e.g., autonomous unmanned vehicles) that, if they do not coordinate in order to resolve possible conflicts, they risk mutual annihilation. Another class of examples are the agents with bounded resources: individually, an agent may lack resources to accomplish any of its desired tasks - yet, if this agent forms a coalition with one or more other agents, the combined resources and joint effort of all agents in such a coalition may provide utility benefits to everyone involved. An example are heterogeneous types of agents deployed to a rescue mission in a disaster area: while there is an overarching, global goal at the system level, each agent typically also has its local, individual notion of task(s) and its own goal(s).

For these reasons, group and coalition formation are of considerable interest for many different kinds of autonomous agents and multi-agent systems, and, among other, even in those multi-agent systems where the agents do not share a global utility function, and where each agent generally acts selfishly. In particular, efficient fully distributed algorithms for effective group formation are needed. Such algorithms should use only a few communication rounds among the agents,

place a very modest computational burden on each agent, and ensure that a distributed consensus among the agents on who is forming a group with whom is effectively, reliably and efficiently reached.

We propose herein one such class of distributed algorithms. We describe the generic, distributed group (coalition) formation algorithm in *Section 4* and give the pseudo-code in the *Appendix*. Variations of this basic max-clique-based group formation algorithm can be designed to meet the needs of various types of agents, such as, to give some examples, the following:

- the classical cooperative DPS agents [9, 10, 18, 19];

- different kinds of self-interested, either strictly competing or competing-and-cooperating agents [14, 24] where concepts, paradigms and tools from *N-person game theory* have found many applications; and, more generally,

- various bounded-resource, imperfect-knowledge agents acting in complex environments (e.g., [24, 30]) that are typically only *partially accessible* to any agent; such autonomous agents are thus characterized by *bounded rationality* [20].

We propose herewith a generic distributed group formation algorithm based on the idea that, in peer-to-peer (in particular, *leaderless*) *MAS*, an agent (node) would prefer to form a group with those agents that it can communicate with directly, and, moreover, where every member of such a potential group can communicate with any other member directly. That is, the preferable groups (coalitions) are actually (maximal) cliques. It is well-known that finding a maximal clique in an arbitrary graph is **NP**-complete in the centralized setting [3, 5]. This implies the computational hardness that, in general, each node faces when trying to determine maximal clique(s) it belongs to. However, if the degree of a node (that is, its number of neighbors in the graph) is small, in particular, if it is $O(1)$, then finding all maximal cliques this node belongs to is 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 small degree, then one has to impose additional constraints in order to ensure that the agents are not attempting to solve an infeasible problem. In particular, we shall additionally require herein that the possible coalitions to be formed, be of up to a certain pre-specified maximum size.

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 graph (network topology) of such *MMAS* is bound to be very large, thus, one may argue, rendering even many typically efficient (i.e., polynomial-time in the number of agents) algorithms obsolete due to their prohibitive cost, let alone allowing distributed coordination strategies that are based on the graph theoretic algorithms that are themselves, in the classical, centralized setting, **NP**-complete in general. However, there is one critical observation that saves the day of the approach that we propose herewith: even if the underlying graph of an *MMAS* is indeed very large (possibly millions of nodes), in many important applications this graph will also tend to be *very sparse*. That is, a typical, average node (agent) will tend to have only a handful, and almost certainly only $O(1)$ neighbors. Therefore, a distributed algorithm where agents act, reason and communicate strictly locally, where no *flooding* of the network is ever performed (or needed), and where each node needs to store and work with only the data pertaining to its near-by nodes, 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 employed, 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 UAVs no central control is feasible or even possible, and the run-time human intervention is either minimal or nonexistent. Such micro-UAV ensembles, therefore, 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 design and simulation challenges of such large-scale ensembles of (micro-)UAVs, see, e.g., [6, 23, 24].

(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 and communication power. In particular, the RAM memory of such sensors is currently typically of the order of Kilobytes, the flash memory range is about $1MB$, the communication bandwidth is $10^2 - 10^3$ Kilobits/second, and a typical battery life span is anywhere from a few hours up to a week. Such smart sensors usually communicate via essentially *local broadcasts* with very limited ranges. The main "communication mode" of the agents in our algorithm in *Section 4* will be precisely the local broadcasts. Also, due to small memory capacities and low power consumption requirements, smart sensors, in order to be effective, have to simultaneously minimize both the amount of local processing, and how much (and how often) they communicate messages to other sensors. Sensor networks, although on a much smaller scale, have already been used as an example to which *dynamic distributed constraint satisfaction/optimization* formalisms can be fruitfully applied [10].

(iii) Various *social networks*, and, in particular, various variants of *'small-world' networks* where, in addition to strictly local connectivity in the communication network topology, a relatively few long-range connections are randomly added [27, 28]. 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., traffic and transportation systems, power grids, etc. For a simple concrete example, consider a car driver (an autonomous agent) participating in the traffic in a large city. While there may be millions of such drivers on the road at the same time, the decision-making of a driver-agent is based, for the most part[1], on a few properties of its local environment, i.e., the actions of near-by agents; likewise, his own actions will usually directly affect only a handful of other agents in the system (e.g., the driver right behind him, or the pedestrian trying to cross the street right ahead of him). Thus, when modeling and simulating such an infrastructure with an appropriate large-scale network and a *MMAS*, the underlying network, while of a very large size, will be in general fairly sparse, and a typical node adjacent to (meaning, in this context, capable of directly affecting and/or perceiving) only a small number of other nodes. An ambitious project on realistic, large-scale modeling and simulation of infrastructures such as city traffic systems, called *TRANSIMS*, is described at [26] and in the documents found therein.

We now return to the dynamic, distributed group or coalition formation in massively multi-agent systems, and the mathematical formalisms and algorithmic solutions proposed for this challenging problem. 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 [19]) and in the context of self-interested agents where each agent has its own individual agenda (as, e.g., in [18, 33]). 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 [9, 19], are also of central importance in our own work on a concrete *MAS* application with a particular type of robotic agents, namely, unmanned aerial vehicles (UAVs), that are residing and acting in bounded resource multi-task environments [6, 23, 24].

Another body of MAS literature highly relevant to our central problem herein, namely 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 [9, 10, 11].

Of the particular relevance to our work herein and other possible extensions of the original maximal clique based group formation algorithm presented in [25] are references [19] and [10]. While Modi et al. in [10] offer the most complete formalization of various distributed resource and/or task allocation problems and general mappings to appropriate types or subclasses of (dynamic) distributed constraint problems, two characteristics of their approach make it unsuitable for a direct application to our modeling framework of massively multi-agent systems in general (see, e.g., [24]), and the application domains we had in mind when devising the algorithm presented herein, in particular.

One, the agents in [10] are strictly cooperative, share the same goals, and have no notion of individual utilities or preferences. While we have studied cooperative *MAS* in [24] and elsewhere, as well, one of our main assumptions is that,

[1] There are exceptions of course; for instance, when an imminently approaching tornado is announced on the radio.

due to a large scale of the system and a high dynamism and unpredictability of the changes in the environment and the goals, no shared or global knowledge of the environment or the goals is maintained, and, 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* [23], and thus using the *incentive engineering* approach [2] to enable the agents to cooperatively coordinate even though each agent is, strictly speaking, *self-interested.*

Two, we address the issue of which agents will select which tasks to serve [23, 24], or, as in this paper, which groups of agents will form in order to serve some particular set of tasks. To that end, what is critical is an agent's or agent coalition's capabilities for serving the desired task(s). Each agent possesses a tuple of capabilities or available resources; likewise, each task has a tuple of resource requirements. An agent coalition can serve a particular task if and only if each component (i.e., individual capability or resource) of its joint tuple of capabilities is greater than or equal to the corresponding requirement vector entry of that task. An agent can only belong to one coalition at a time, and work with its fellow coalition members on one task at a time. Thus, while the agent operations in [10] are mutually exclusive with respect to time (an agent can only execute a single operation at any time), our agent capabilities are mutually exclusive with respect to space or, equivalently, task allocation. Clearly, a complete model of resource and task allocation in *MMAS* should incorporate both aspects. Since we are presently only interested in coalition formation for the purpose of coalition-to-task mapping, but we are not concerned herein with *how* are then these agent coalitions exactly going to perform the tasks they have been mapped to, our framework as described in [24] and outlined herein suffices for the current purposes.

The importance of DCS in *MAS* in general is discussed, e.g., in [32]. 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 generic, fully distributed, scalable and efficient algorithm for ensembles of autonomous agents to use as a subroutine - that is, as a part of their coordination strategy - with a purpose of efficiently forming temporary groups or coalitions.

The proposed algorithm is a graph algorithm. The underlying undirected graph[2] captures the communication (ad hoc) network topology among the agents, as follows. 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 at the time our distributed group formation subroutine is called. 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.[3].

The basic idea is to efficiently partition this graph into (preferably, *maximal*) *cliques* of nodes. These maximal cliques would usually also need to satisfy some additional criteria in order to form temporary coalitions of desired quality. 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 of the underlying graph considerably changes, either due to the agents' mobility, or because many old links have died out and perhaps many new, different links have formed,

[2] For simplicity, we assume the graph is undirected, even though the communication from one node to another is clearly directional, and the communication links need not be symmetric. However, since the nodes eventually need to reach a mutual consensus, which requires that *all* members of a coalition agree to the same coalition and *notify* all other coalition members of the agreement, the assumption about the edge (non-)directionality is inconsequential, in a sense that only those coalitions that indeed are cliques in the corresponding *directed graph* will ever be agreed upon by the agents.

[3] We point out that this definition of the graph edges can be made tighter by imposing additional requirements, such as, e.g., that the two agents (that is, graph nodes), if they are to be connected by an edge, also need to be compatible, for instance, in terms of their capabilities, that they each provide some resource(s) that the other agent needs, and/or the like.

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 group formation algorithm may need to be invoked a number of times as a coordination subroutine.

We assume that each agent has a locally accurate (see, e.g., discussion in [24]) picture of (i) who are its neighboring agents, and (ii) what are the near-by tasks and, in particular, what are these tasks' resource requirements. Each agent is equipped with a tuple of its internal resources, or *capabilities* [19]. Each task requires certain amount of each of the individual resources in this tuple in order to be serviced. A single agent, or a coalition of two or more agents, can serve a particular task if and only if their joint capabilities suffice with respect to the task's resource consumption requirements. That is, the sum of each component of the capability vector taken over all the agents in the coalition has to be greater than, or equal to, the corresponding component of the task's resource consumption vector.

Our distributed max-clique based group formation algorithm is sketched 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 not to be an issue.

- Each agent has a sufficient local memory for storing all the information received from other agents.

- Communication is reliable during the group formation, in the following sense: if an agent, A, sends a message to another agent B (either via a local broadcast where it is assumed that B is within the *communication range* of A, or in a direct, peer-to-peer manner), either agent B gets *exactly* the same message that A has sent, or else the communication link has completely failed and so B does not receive anything from A at all. In particular, we assume no *scrambled* or otherwise modified messages are ever received by any receiving agent. Also, once the groups are formed, the above assumption on communication reliability need no longer hold[4].

- Each agent has (or else can efficiently obtain) a reliable knowledge of which other agents are within its communication range.

- Each agent, i.e., each network node, has a unique global identifier, 'UID', and the agent knows its UID.

- There is a total ordering, $\prec$, on the set of UIDs, and each agent knows this ordering $\prec$.

- 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. If agent A has sent a message (e.g., a coalition proposal - see *Section 4*) to agent B, and the latter is not responding at all, there are three possibilities: (i) agent B has failed; (ii) communication link from B to A has failed; or (iii) while B is in A's communication range, *vice versa* actually does not hold (but A may not know it).

- The *veracity* assumption holds, i.e., an agent can trust the information about tasks, capabilities, etc. it receives from other agents.

To summarize, when agent A sends a message to B, then B either receives exactly what A had sent, or nothing at all - and, moreover, if B has received the message, B knows that A is telling the truth about its neighbors, capabilities, commitments and preferences in it (see *Appendix*).

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.

In addition to its globally unique identifier UID, which we assume is a positive integer, and the vector of capabilities, each agent 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 group (coalition). Namely, $decision\_flag \in \{0, 1\}$, 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" the agent is with its current tentative choice

[4] 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 groups even in the presence of some limited amount of communication noise during the group formation process.

or proposal of the group 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. For more details, we refer the reader to *Appendix* and reference [25].

4 An Outline of Max-Clique-Based Distributed Group Formation for Task Allocation

Now that the assumptions have been stated and the notation has been introduced, we outline our distributed maximal clique based coalition formation algorithm. We describe in some detail how the algorithm works below; the pseudo-code is given in *Appendix*.

The proposed distributed group or coalition formation algorithm is based on two main ideas. One idea, familiar from the literature (see, e.g., [19] 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 group overlaps are allowed, or a *(distributed) set partitioning problem, (D)SP,* when no group overlaps are allowed. Two (or more) groups overlap if there exists an element that belongs to both (all) of them. It is well-known that decision versions of the classical, centralized versions of the *SC* and *SP* problems are **NP**-complete [5]. Hence, we need efficient (distributed) heuristics so that the agents can effectively deploy *DSC-* or *DSP-based* strategies for coalition formation. Fortunately, some such efficient heuristics are already readily available [19].

The second main idea is to ensure that the formed groups of agents meet the robustness and fault tolerance criteria, which are particularly important in applications where there is a high probability of node and/or communication link failures. Indeed, we were primarily motivated by such applications when designing the algorithm proposed herewith (see [23, 24]). The most robust groups of agents of a particular size are those that correspond to *cliques* in the underlying interconnection topology of the agents' communication network. Moreover, the larger such a clique, the more robust the group of agents in the clique with respect to the node and/or link failures. Hence, appropriate maximal cliques need to be formed in a distributed fashion. However, the *Maximal Clique* problem is also well-known to be **NP**-hard [3, 5]. This hardness stems from the fact that an agent, in the general case (arbitrary underlying graph), may need to test for "cliqueness" exponentially many candidate subsets that it belongs to. However, in graphs where the maximum degree of each node is bounded by $c \log N$, where c is a constant and N is the total number of nodes in the graph, the number of subsets that each node belongs to, and therefore the number of candidate cliques, is $O(N^c)$, i.e., *polynomial in the total number of nodes*, N. In particular, in sufficiently sparse graphs, where the node degrees are bounded by some (small) constant, $K = O(1)$, the size of any maximal clique cannot exceed K. In such situations, since 2^K is presumably still sufficiently small, finding maximal cliques becomes both theoretically feasible (i.e., solvable in the time polynomial in the number of agents) and practically computable in the online, real-time scenarios that are of main interest in *MMAS* applications [5].

We approach distributed coalition or group formation for task allocation as follows. The "candidate coalitions" are going to be required (whenever possible) to be cliques of modest 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 agent-to-agent communication links, etc.), *a priori* chooses a threshold, K, such that only coalitions of sizes up to K are considered. Agents themselves subsequently form groups in a fully distributed and online manner, as follows. Each agent (i) first learns of who are its neighbors, then (ii) determines appropriate candidate coalitions, that the agent hopes are (preferably maximal, but certainly of size 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 (highest utility value to the agent) candidate coalition, and, finally, (v) sends this choice to all its neighbors.

[5] We remark that there are also other important and frequently encountered in practice special cases, in terms of the structural properties of the underlying graphs, where the *Maximal Clique* problem turns out to be computationally feasible. In particular, the sharp uniform upper bound on all node degrees is suffi cient, but not necessary, for the computational feasibility of the *Max Clique* problem.

This basic procedure is then repeated (see the *WHILE* loop in the pseudo-code in *Appendix*), 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.

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 basic information it has received, and keeps receiving, from its nearest neighbors), 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 possibly become the *actual* (as opposed to merely *potential*) coalitions. This observation justifies the name of our algorithm. Moreover, in case of the candidate coalitions of fewer than K elements (see *Appendix*) that actually end up being agreed upon by the agents, and therefore becoming the *actual coalitions*, it can be proved that these groups of nodes actually do form *maximal cliques* in a sense that these cliques can be possibly made larger in only two ways: by adding the nodes so that the threshold K is exceeded, and/or by taking away the nodes that already belong to another, equally good or better, coalition.

The algorithm proceeds in five major stages. The only assumption about *synchrony* among different nodes is that no node begins stage $n + 1$ before all the nodes have completed stage n (for $n \in \{1, 2, 3, 4, 5\}$) [6]. Within each stage, however, every node (agent) does its local computations, as well as communication (local broadcasts) independently, i.e., asynchronously and in parallel with respect to other agents. We do assume that no node failures take place during the group formation; communication links, on the other hand, are allowed to fail - but the tacit assumption is that there won't be too many such link failures, so that non-trivial clique-like groups can be formed.

The five stages of the algorithm, and a brief description of each stage, follow:

Stage 1:
Set $counter \leftarrow 1$.
Each node (in parallel) broadcasts a tuple to all its immediate neighbors. The entries in this tuple are (i) the node's UID, (ii) the node's list of (immediate) neighbors, L(i), (iii) the value of the choice flag, and (iv) the value of the decision flag.

WHILE *(not every agent has joined a group yet)* DO

Stage 2:
Each node (in parallel) computes the overlaps of its neighborhood list with the neighborhood lists that it has received from its neighbors, $C(i, j) \leftarrow L(i) \cap L(j)$. Repetitions (if any) among this neighborhood list intersections are deleted. The remaining intersections are ordered with respect to the list size (the ties, if any, are broken arbitrarily), and a new (ordered) collection of these intersection lists (heretofore referred to simply as 'lists') is then formed.

If $counter > 1$ then:
Each node looks for information from its neighbors, whether they have joined a group "for good" during the previous round. Those neighbors that have (i.e., whose *decision-flag* $= 1$), are deleted from the neighborhood list $L(i)$; the intersection lists $C(i, j)$ are also updated accordingly, and those $C(i, k)$ for which k is deleted from the neighborhood list $L(i)$ are also deleted.

Stage 3:
Each remaining node (in parallel) 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 this means, that one of the lists of maximal length is chosen. If the combined *capabilities* of each tentative coalition for servicing various tasks is the main criterion, then each agent evaluates or estimates the *coalition value* with respect to its (possibly imperfect, and generally local) knowledge of the existing tasks and their demands in terms of resources or capabilities. To evaluate these coalition values of what are as of yet only *tentative*

[6] We do hope to relax this assumption in future refi nements of the current version of the algorithm.

coalitions, the agent 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 max clique coalition is ever chosen - except in the cases when the clique size exceeds the pre-specified threshold, K.

Irrespective of the exact criteria for the coalition quality, once the agent has partially ordered all candidate coalitions whose sizes do not exceed the upper bound K, it picks the best (or, in case of a tie, one of the best) coalition(s) with respect to those criteria. Then, the value of the choice flag is set, based on whether the agent has other choices of candidate coalitions that are as preferable as the current choice, and, if not, whether there are any other still available (nontrivial) choices at all.

Stage 4:
Each node (in parallel) 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.

Stage 5:
Each node i (in parallel) compares its chosen list $C(i)$ with lists $C(j)$ received from its neighbors. If a clique that includes the node i exists, and all members of this clique have selected it at this stage as their current group or coalition of choice (that is, if $C(i) = C(j)$ for all $j \in C(i)$), this will be efficiently recognized by the nodes forming this clique. The decision flag of each node $j \in C(i)$ is set to 1, a group is formed, and this information is broadcast by each node in the newly formed group to all of its 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 group of choice, $C(i)$. The latter scenario is possible only if *choice* > 0, meaning that there are other choices of *potential* coalitions $C(i)$ that have not been tried out yet that are still available to agent i.

$counter \leftarrow counter + 1$;
END DO $[* \, end \, of \; WHILE \; loop \, *]$

If $round > 1$ then, at Stage 2, each node looks for the information from its neighbors to find out if any of them have joined a group in the previous round. For those nodes that have (i.e., whose decision flag $dec = 1$), each node neighboring any such already committed node deletes this committed node from its neighborhood list $L(i)$, updates all $C(i, j)$ that remain, and selects its choice of $C(i)$ based on the updated collection of group choices $\{C(i, j) : j \in L(i)\}$. That is, now all those nodes that have already made their commitments and formed groups are not "in the game" any more, and are therefore deleted from all remaining agents' neighborhood lists as well as the tentative choices of coalitions. (Of course, the only coalition a committed agent is *not* deleted from at this stage is *the coalition* that this agent has just joined).

It can be readily shown that, once all agents exit the *WHILE* loop (that is, a repeated execution of the *Stages 2-5*), each thereby formed group is, indeed, 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 (i) each of the new agents is already adjacent in the communication topology 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) did not already belong to a coalition (or coalitions) at least as good as C, and (iv) the new size of the augmented coalition C is still at most K.

However, it is easy to construct examples of underlying graphs and particular "legal" runs of the algorithm (cf. in terms of the "tie-breaking" when an agent has two or more equally preferred choices to choose from) such that, once every node joins a coalition and the algorithm terminates, several agents end up in trivial coalitions, such as, e.g., groups of size 1 or 2. It is therefore reasonable, in most applications, to introduce an (optional) *Stage 6* of the algorithm, where these small - and therefore potentially not sufficiently robust, or useful - coalitions, whenever possible, are merged together. That is,

if some two small coalitions are adjacent to each other[7], they can be merged together. Obviously, the connectivity of such coalitions, and therefore their tolerance to communication link failures, is in general going to be lower than that of those coalitions that are genuine cliques.

How are the non-clique coalitions to be formed, i.e., what criteria are to be used for merging together small groups into larger ones, critically depends on the nature of the underlying application and the designer's priorities when it comes to the agent coalitions' desired properties. Further discussion of this important issue, however, is beyond our current scope.

There are some more details in the algorithm that we leave out for the space constraint reasons. One important technicality is that, in order to ensure that the algorithm avoids to cycle in every possible scenario, once an agent 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 (and the only remaining) choice, and waits for other agents to converge to their choices. It can be shown that this ensures ultimate convergence to a coalition structure that all agents agree to. That is, under the assumptions stated in the previous section, the agents will reach consensus on the coalition structure after a finite number of rounds inside the WHILE loop (see also *Appendix*). Moreover, if the maximum size of any $L(i)$ is a (small) constant, then the convergence is fast.

5 Analysis and Discussion

We have outlined a fully distributed algorithm for group or coalition formation based on maximal cliques, combined with the set partition based distributed task allocation. This algorithm will be feasible when the underlying graph is relatively sparse, and, in particular, when the sizes of all maximal cliques are bounded by $c \cdot \log N$ for some small (i.e., close to 1) constant c and the number of agents N. For N very large, the proposed algorithm will be highly efficient if the maximal node degree in the underlying graph is bounded by a constant $K = O(1)$ of a modest size. When this is not the case (or when it cannot be guaranteed to always hold), appropriate restrictions can be imposed "from the outside" to ensure that the algorithm (i) converges, and (ii) is of a feasible computational complexity. Also, the coalition value for each sufficiently small subset of the set of all agents has to be efficiently computable by each agent involved.

Once the groups are formed, these groups will be tight (as everyone in the group can communicate with everyone else), and, in nontrivial cases, therefore as robust as possible for a given number of group members with respect to either node or link failures. This is a highly desirable property involving coalitions or teams of agents operating in environments where both the agent failures and the agent-to-agent communication link failures can be expected. One example of such *MAS* application domain, and in particular coordination strategies in this domain, are studied in [6, 7, 23, 24].

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 would join an appropriate (new) coalition because the agent finds this to be in its individual best interest, and where it is important for agents to agree *effi ciently* on what coalitions are to be formed, rather than spending precious resources on complex negotiation protocols.

It is important, however, to point out that, in case of the self-interested agents, some form of negotiation among the agents is typically unavoidable, irrespective of the particular mechanism employed for the purposes of generating a desired coalition structure. In particular, self-interested agents that would use a variant of our algorithm presented in the previous section, would still need to negotiate, at the very least, on how are the spoils to be divided up among them. According to T. Sandholm in *Chapter 5* of [29], a complete coalition formation process is made of three stages: (i) coalition structure generation, (ii) optimization within each coalition, and (iii) payoff (or utility) distribution; see also [17]. In case of the DPS agents, stages (i) and (ii) generally suffice, but when the agents have their individual preferences over the states of the world in general, and the desirability of different coalitions and tasks in particular, addressing (iii) is necessary for a coalition formation strategy to be complete and effective.

[7] That is, if there exist node x in the fi rst coalition and node y in the second such that x and y are adjacent in the underlying graph.

We also notice that, in our approach, the stages (i) and (ii) are not separated from each other (so that forming coalitions causally and temporally precedes the agents within each coalition then attempting to solve an appropriate optimization problem), but, instead, (i) and (ii) are *intertwined.* That is, the desired coalition structure emerges as a result of the agents solving simultaneously a coordination and a distributed constraint optimization problem. We briefly outline our view on why, in many MAS applications, stages (i) and (ii) in Sandholm's classification scheme need not be separated as in [17]. Since in most MAS situations coordination (herein, reduced to forming groups or coalitions) is not a goal in itself, but, rather, a capability needed for, or an effective approach to, solving an underlying optimization problem, such as (distributed) resource or task allocation, the good ways to coordinate are those that enable the agents to effectively solve the underlying optimization problem. Thus the "goodness" of particular coalitions and the overall coalition structure (stage (i)), often cannot be separated from "optimization within each coalition" (stage (ii)): to optimize well *precisely means* to form good coalitions with respect to the properties of the agents' tasks and resources.

An important game-theoretic consideration in the context of coalition formation in *N-person games* [13] is that of the nature of the environment, and, in particular, whether the environment is *super-additive*, *sub-additive*, or neither. The nature of the environment and its implications in the MAS context were formalized and discussed, e.g., in [17, 18]. Initially, in designing the maximal clique group formation algorithm [25], we have tacitly assumed *locally super-additive environments*, that is, super-additivity subject to the constraints stemming from the communication network topology that restricts which agents can communicate to each other *directly* (as opposed to via multiple hops). Local or constrained super-additivity still holds when the coalition quality is measured with respect to the *joint capabilities* of the coalition members, as long as either a "single shot" coalition-to-task mapping is performed, or, alternatively, if each agent's resources or capabilities are renewable after each task (or round of tasks) is completed. Hence, in this context, insofar as the coalition-to-task mapping is concerned, all that matters is that the total resources of agents in the coalition exceed the resource requirements of the desired task - but *by how much* is not considered relevant. Once the agent resources are depleteable, and an agent is expected to participate in completing several tasks (with each agent or coalition still restricted to being able to work on only one task at a time), on the other hand, these agents would be interested in long-term *planning*, in computing the *marginal utility* of servicing each task [16], etc. These considerations, however, are beyond our current scope. For our purposes herein, without further ado, the environments are assumed *locally super-additive* in the sense outlined above. Once the issue of how is the payoff for servicing the tasks to be distributed among the respective coalition members is brought to the picture (stage (iii) in Sandholm's classification), even the local (or constrained) super-additivity assumption, in general, becomes hard to justify. However, as already mentioned, in the present work we intentionally avoid addressing the issue of the payment disbursements altogether.

Last but not least, we emphasize that our algorithm as a coordination subroutine in MAS 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., [24, 25]).

6 Summary

We have proposed herewith a generic algorithm for distributed group formation based on (maximal) cliques of modest sizes. We find this algorithm, or its appropriately fine-tuned variants, to be a potentially very useful subroutine in many multi-agent system applications, where the interconnection topology of the agents often changes 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 known from the MAS literature, on the other hand. In particular, we would like to

compare and contrast the purely P2P, 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 approaches (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 (even if only temporarily) special roles or "leader" status. However, this intuition needs to be both further theoretically investigated and experimentally tested and validated via appropriate comparative simulations and performance measurements.

One possible "testing ground" for determining the practical usefulness of our approach to multi-agent coordination, and its comparative (dis)advantages with respect to other approaches known from the literature, is the scalable simulation of bounded-resource autonomous unmanned aerial vehicles (UAVs) on a complex multi-task mission, developed at Open Systems Laboratory (OSL); see *http://osl.cs.uiuc.edu/* for more details. Our short-term plans, therefore, include implementation and testing of appropriately fine-tuned variants of the algorithm presented herein in the context of the OSL's large-scale UAV simulation.

Acknowledgment: Many thanks to Myeong-wuk Jang, Nirman Kumar and Reza Ziaei (all of Open Systems Laboratory, UIUC) for many useful discussions. This work was supported in part by the *DARPA IPTO TASK Program* under the contract *F30602-00-2-0586*. The first author would also like to acknowledge and express his gratitude for 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. S. Franklin, A. Graesser, "Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents", Proc. 3rd Int'l Workshop on Agent Theories, Architectures & Languages, Springer-Verlag, 1996
5. M. R. Garey, D. S. Johnson, "Computers and Intractability: a Guide to the Theory of NP-completeness", W. H. Freedman & Co., New York, 1979
6. M. Jang, S. Reddy, P. Tosic, L. Chen, G. Agha, "An Actor-based Simulation for Studying UAV Coordination", Proc. 15th Euro. Symp. Simul. (ESS 2003), Delft, The Netherlands, 2003
7. 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
8. N. Lynch, "Distributed Algorithms", Morgan Kaufmann Publ., Wonderland, 1996
9. 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
10. P. J. Modi, H. Jung, W. Shen, "Distributed Resource Allocation: Formalization, Complexity Results and Mappings to Distributed CSPs", technical report (extended version of [9]), November 2002
11. P. J. Modi, W. Shen, M. Tambe, M. Yokoo, "An asynchronous complete method for distributed constraint optimization", Proc. 2nd AAMAS-03, Melbourne, Australia, 2003
12. J. von Neumann, O. Morgenstern, "Theory of Games and Economic Behavior", Princeton Univ. Press, 1944
13. A. Rapoport, "N-Person Game Theory", The Univ. of Michigan Press, 1970
14. J. Rosenschein, G. Zlotkin, "Rules of Encounter: Designing Conventions for Automated Negotiations among Computers", The MIT Press, Cambridge, Massachusetts, 1994
15. S. Russell, P. Norvig, "Artificial Intelligence: A Modern Approach", 2nd ed., Prentice Hall Series in AI, 2003

16. 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.
17. T. Sandholm, V. Lesser, "Coalitions among Computationally Bounded Agents", *Artificial Intelligence*, spec. issue on "Principles of MAS", 1997
18. O. Shehory, S. Kraus, "Coalition formation among autonomous agents: Strategies and complexity", Proc. MAAMAW'93, Neuchatel, Switzerland, 1993
19. O. Shehory, S. Kraus, "Task allocation via coalition formation among autonomous agents", Proc. 14th IJCAI-95, Montreal, August 1995
20. H. A. Simon, "Models of Man", J. Willey & Sons, New York, 1957
21. R. G. Smith, "The contract net protocol: high-level communication and control in a distributed problem solver", IEEE Trans. on Computers, 29 (12), 1980
22. G. Tel, "Introduction To Distributed Algorithms", 2nd ed., Cambridge Univ. Press, 2000
23. P. Tosic, M. Jang, S. Reddy, J. Chia, L. Chen, G. Agha, "Modeling a System of UAVs on a Mission", Proc. SCI 2003 (invited session), Orlando, Florida, 2003
24. P. Tosic, G. Agha, "Modeling Agents' Autonomous Decision Making in Multiagent, Multitask Environments", Proc. 1st Euro. Workshop on MAS (EUMAS 2003), Oxford, England, 2003
25. P. Tosic, G. Agha, "Maximal Clique Based Distributed Group Formation Algorithm for Autonomous Agent Coalitions", Proc. Workshop on Coalitions & Teams, AAMAS '04, New York City, New York, July 19-23, 2004
26. 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)
27. D. J. Watts, "Small Worlds: The Dynamics of Networks Between Order and Randomness", Princeton Univ. Press, Princeton, N. Jersey, 1999
28. D. J. Watts, S. H. Strogatz, "Collective dynamics of 'small-world' networks", *Nature* 393, 1998
29. G. Weiss (ed.), "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence", The MIT Press, Cambridge, Massachusetts, 1999
30. M. Wooldridge, N. Jennings, "Intelligent Agents: Theory and Practice", Knowledge Engin. Rev., 1995
31. M. Yokoo, K. Hirayama, "Algorithms for Distributed Constraint Satisfaction: A review", AAMAS, Vol. 3, No. 2, 2000
32. M. Yokoo, "Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems", Springer, 2001
33. G. Zlotkin, J.S. Rosenschein, "Coalition, cryptography and stability: Mechanisms for coalition formation in task oriented domains", Proc. AAAI'94, Seattle, Washington, 1994

7 Appendix:
Pseudo-Code for Max-Clique-Based Distributed Group Formation for MAS Task Allocation

Notation:

$i :=$ the i-th agent (node) in the system (say, $i = 1, ..., n$)
$V(i) :=$ the i-th node's UID
$N(i) :=$ the list of neighbors of node i
$L(i) :=$ the extended neighborhood list (i.e., $L(i) = N(i) \bigcup \{i\}$)
$C(i,j) = L(i) \bigcap L(j)$
$C(i) :=$ the group of choice of node i at the current stage (i.e., one of the $C(i,j)$'s)
$choice(i) :=$ the choice flag of node i
$dec(i) :=$ the decision flag of node i

Remark: For simplicity, clarity and space limitations, we focus on distributed computation of cliques, and consensus reaching on what clique-like coalitions are to be formed. We therefore assume that there is a highly efficient, readily available subroutine for each agent to evaluate (or estimate) the utility value of each potential coalition. This subroutine is called independently by each agent inside of *Stage 3* below.

Max Clique-based Distributed Coalition Formation Algorithm:

Stage 1:
DOALL $i = 1..n$ (in parallel, i.e., each node i carries the steps below locally)
send $[V(i), L(i), choice = 3, dec = 0]$ to each of your neighbors
END DOALL

WHILE *(not all agents have joined a group)* DO
[$\star$ *Stages 2-5* are repeated until consensus on the coalition structure is reached $\star$]

Stage 2:
DOALL $i = 1..n$
FOR all $j \in N(i)$ DO [$\star$ check if $dec(j) = 1$ $\star$]
if $dec(j) == 1$ then delete j from $N(i), L(i),$ and $C(i,j'),\ \forall j' \in N(i) - \{j\}$
END DO [$\star$ end of FOR loop $\star$]
FOR all $j \in N(i)$ DO [$\star$ FOR all remaining (undeleted) indices j $\star$]
compute $C(i,j) \leftarrow L(i) \bigcap L(j)$
END DO [$\star$ end of FOR loop $\star$]
END DOALL

Stage 3:
DOALL $i = 1..n$
FOR all $j \in N(i)$ DO
compute *utility value* $val[C(i,j)]$ of each candidate coalition $C(i,j)$ [8]
END DO
pick $C(i,l)$ such that $val[C(i,l)] = max_{j \in N(i)}\ val[C(i,j)]$ (subject to $|C(i,j)| \leq K$)
$C(i) \leftarrow C(i,l)$;

[8] We require throughout, that all groups to be considered, that is, all "candidate coalitions", be of appropriately bounded size, $|C(i,j)| \leq K$.

if *(there is more than one such choice of max. utility value)* then
set $choice(i) \leftarrow 2$;
else *(if there are other choices* $C(i,j')$ *but only of strictly smaller utility value)*
set $choice \leftarrow 1$;
else *(if node* i *has no alternatives left for a non-trivial coalition that would include* i*)*
set $choice \leftarrow 0$;
END DOALL

Stage 4:
DOALL $i = 1..n$
send $[V(i), C(i), choice, dec = 0]$
END DOALL

Stage 5:
DOALL $i = 1..n$
compare $C(i)$ with $C(j)$ received from one's neighbors $j \in N(i)$;
if (*there exists a clique* $\{i, j_1, j_2, ...j_l\}$ *such that* $C(i) = C(j_1) = C(j_2) = ... = C(j_l)$)
then set $dec \leftarrow 1$ (an agreement has been reached);
broadcast group $G = \{i, j_1, j_2, ..., j_l\}$ and $dec(i) = 1$ to all neighbors $j \in N(i)$:
send $[V(i), C(i) = G, choice, dec = 1]$
else (*based on UID* i *and the priority as defi ned by the relation* $\prec$)
either DO NOTHING
or change your mind: $C(i) \leftarrow new\ choice\ C(i,j)$
(from the list of candidate groups that are still available)
END DOALL
END DO $[* end\ of\ WHILE\ loop *]$

This article was processed using the LaTeX macro package with LLNCS style