Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents *

Onn Shehory[1] **Sarit Kraus**[1,2]
1. Department of Mathematics and Computer Science
Bar Ilan University Ramat Gan. 52900 Israel
{shechory. sarit}@cs.biu.ac.il
2. Institute for Advanced Computer Studies
University of Maryland. College Park 20742,
Maryland, USA

Abstract

Goal-satisfaction in multi-agent environments via coalition formation may be beneficial in cases where agents cannot perform goals by themselves or they do so inefficiently. Agent coalition formation typically requires that each agent must be a member of only one coalition. This may lead to wasted resources and capabilities. Therefore, we present algorithms that lead agents to the formation of overlapping coalitions, where each coalition is assigned a goal. The algorithms we present are appropriate for agents working as a Distributed Problem Solving system in non-super-additive environments. They are any-time distributed algorithms with a low computational complexity and low ratio-bound.

Introduction

Goal-satisfaction in multi-agent environments via coalition formation may be more beneficial in cases where agents cannot perform goals by themselves or they do so inefficiently. The typical approach to agent coalition formation requires that each agent be a member of only one coalition. However, overlapping coalitions, where multiple memberships are allowed, may improve the utilization of resources, thus improving the goal-performing efficiency and increasing the outcome of agent-systems. We present algorithms that enable coalition-formation for distributed goal-satisfaction, where goals have precedence order and agents have no central authority, which provide sub-optimal solutions with a low complexity. The major differences of our approach from the common coalition formation methods employed in DAI (Zlotkin & Rosenschein 1994; Ketchpel 1994; Sandholm & Lesser 1995; Shehory & Kraus 1995) and in game theory (Zhou 1994; Kahan & Rapoport 1984) are the concepts of overlapping coalitions and goals' precedence order.

In (Shehory & Kraus 1995) a coalition formation procedure for agents in a Distributed Problem Solving system was presented. There, dynamically alternating inter-related coalitional values were presented. We adopt this approach and add to it the overlapping coalitions approach and the precedence-ordered goals concept. This combination leads to a more realistic multi-agent model. We present coalition formation algorithms, where each coalition is assigned a goal. Our algorithms are based on a combination of a combinatorial algorithmic approach and concepts from operations research with autonomous agents and distributed computing systems methods. The resulting, possibly overlapping, coalitions may increase the benefits of agent-systems as compared to the case of disjoint coalitions. We concentrate on environments which are not necessarily super-additive (Conte, Miceli, & Castelfranchi 1991). Non-super-additivity may arise when adding a new agent to the coalition is costly[1], and as the size of the coalition increases, it becomes less beneficial to form it. Such environments are more challenging and realistic in DPS systems (see below), whereas cooperation within a DPS system in a super-additive environment will most probably yield the grand coalition[2].

Related work

Several solutions to the coalition formation problem have been suggested by DAI researchers, concentrating on the case of super-additive environments e.g. (Ketchpel 1994; Zlotkin & Rosenschein 1994). Most of these solutions address Multi-Agent Systems (MAS), where each agent tries to increase its own personal utility via cooperation (Shehory & Kraus 1996). We present coalition formation algorithms which are appropriate for the Distributed Problem Solving (DPS) cases, where the agents cooperate in order to increase the outcome of the system (Shehory & Kraus 1995). In addition, our solution is most appropriate for non-super-additive domains as previously stated.

In (Sandholm & Lesser 1995), a coalition formation model for bounded-rational agents and a general classification of coalition games is presented. There, a coali-

*This material is based upon work supported in part by the NSF under Grant No. IRI-9423967 and the Israeli Science Ministry grant No. 6288.

[1]Such costs may arise from the intra-coalition coordination and communication costs; these increase with the size of the coalition.

[2]The grand coalition includes all of the agents.

tion value depends on the computation time. We also allow for varying coalitional values. However, we consider cases in which values vary with respect to the resource-consumption by previously-formed coalitions.

DPS systems in which tasks are allocated to agents have been discussed previously, e.g., the Contract Net Protocol (CNP)(Smith 1980). In the CNP, goals are allocated to single agents and a procedure for goal-partitioning is necessary. The CNP allows for single agents to perform more than one sub-goal. This is similar to our approach as we, too, allow that agents will be involved in the performance of more than one goal. However, we solve the problem of assigning goals to coalitions of agents. In cases where single agents cannot satisfy goals by themselves and goals cannot be partitioned, or the partition is computationally too complex, close cooperation is required, preferably within coalitions.

DAI research has previously addressed the problem of tasks with precedence order and of overlapping problem solvers, as in (Durfee, Lesser, & Corkill 1988). There, coordination is based on an organizational view of node activity, where each node acts subject to its local control, solving sub-goals of the global goal. The organizational view implicitly resembles the idea of a coalition. However, Durfee et al do not discuss coalitions nor do they address the problem of forming groups of nodes to improve the overall performance of goals.

Game theory provides an analysis of the possible coalitions that shall form as a result of a coalition formation process, and what the resulting disbursements to the agents shall be, assuming no multiple memberships in coalitions, e.g., (Rapoport 1970). However, game theory does not provide the algorithms for coalition formation. Given a coalitional configuration, game theory usually concentrates on checking its stability or its fairness and on the calculation of the corresponding payments. Game theory rarely takes into consideration the communication costs and limited computation time, and the solutions are not distributed. We are particularly interested in the distributed coalition formation mechanism. We also seek a dynamic evaluation of the coalitions, where game theory usually provides a static evaluation, and we allow agents to be members of more than one coalition.

Coalition formation where coalitions may overlap can be approached as a Set Covering Problem (SCP). Exact solutions and approximations to SCP have been proposed in the fields of operations research, combinatorial algorithms, and graph theory (Balas & Padberg 1975; Christofides & Korman 1975; Chvatal 1979). However, the solutions that have been proposed are inappropriate for the problem of coalition formation among agents (see the set covering section).

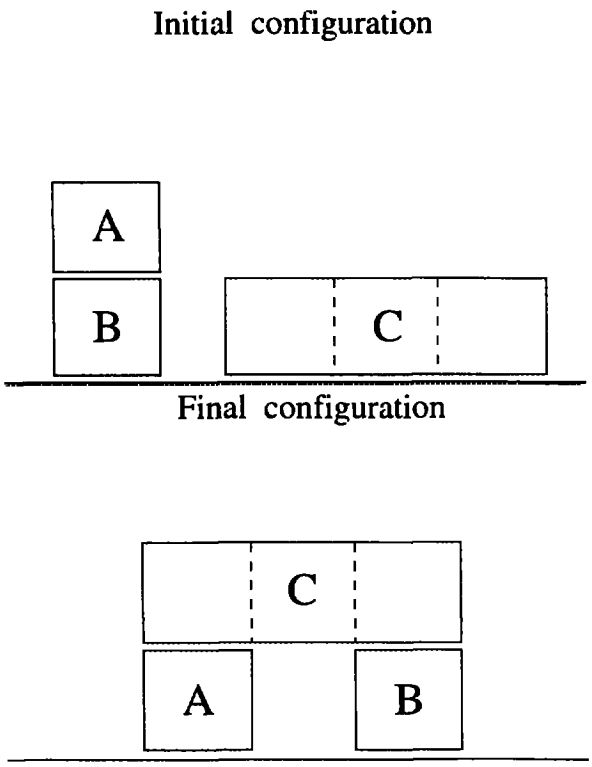
Distributed computing systems (DCS) research has dealt with problems of task allocation with precedence order. Optimal solutions were provided only for strongly constrained cases of two-processor systems (e.g. (Coffman & Graham 1972)). The general problem is NP-complete, but some approximation algorithms (Wang & Tsai 1988) provide good solutions for the multi-processor system as in (Chen & Yur 1990). There, the suggested solution is aimed at reducing the task turnaround time. The minimization of the task turnaround time is the main objective of task assignment within distributed computing systems. In our paper, the main issue is the development of a task allocation that will increase the benefits of the agent-system, and not necessarily reduce the execution time.

Research on the Loading Time Scheduling Problem (LTSP) (Bhatia, Khuller, & Naor 1995) presents approximation algorithms for precedence-ordered task execution in cases where tasks may be performed. The main factor in the LTSP is the loading time of the first task in a sequence of tasks on a specific machine, while we discuss neither the loading time nor the effect of task-sequences on a single agent. The differences of our case from the DCS research requires another approach, as we present below.

Problem representation

The problem we solve in this paper is that of the formation of overlapping coalitions for goal-satisfaction among autonomous agents in a DPS system, where goals may have a precedence order. Subject to this precedence order, the system as a whole must seek maximal goal-satisfaction, thus maximizing its benefits. There is no central authority that distributes the goals among the agents. An efficient allocation is achieved via the formation of overlapping coalitions.

We demonstrate the problem using a Blocks' World domain as in the figure below. The blocks are of various sizes (for simplicity, we consider the case of a unit-block and a row of attached unit-blocks). Each unit-block weighs one weight unit. The blocks should be moved from the initial configuration to the final configuration. This should be done by a group of agents, each capable of lifting a limited weight (e.g., 2 weight units) and moving it aside as much as necessary. Each block can be carried by a limited number of agents (due to physical limitations). We do not discuss the planning problem but assume a pre-given plan. This plan shall divide the goal into subgoals with a precedence order. Such subgoals, which are part of the global plan, are presented in the figure below. One cannot place block C before blocks A and B are properly located. In addition, cooperation among agents is necessary. Block C cannot be lifted by less than 2 agents. However, 4 agents can do so but less efficiently, due to coordination costs, and 20 of them will be far too many. Obviously, any member of the group that places block C may be a member of a group that places A or B. However, if the agents have a limited amount of fuel, they may run out after performing a set of goals, and therefore they cannot be part of all working groups.



Another example of our problem is that of a transportation firm. Transportation systems have been extensively used as examples for DPS systems, e.g., in (Wellman 1992; Sandholm 1993). The firm possesses autonomous and automated airborne, naval and terrain vehicles which we refer to as agents. The aim of the firm is to provide a variety of transportation services via its agents. Each individual agent is capable of performing limited transportation goals by itself. However, some goals cannot be performed by a single agent. The agents differ in their capabilities, i.e., in the method by which they perform deliveries, in the size, volume and weight of freights that they can deliver, in the transportation speed and in its costs. In cases where agents cannot perform a given transportation goal by themselves, cooperation is necessary. The agents shall form coalitions that will perform a transportation goal cooperatively. Some coalitions are not able to perform certain goals, and some goals cannot be performed unless other goals have been satisfied previously. The benefits that may arise from the performance of a specific goal depend on the coalition that performs this goal. This is a result of different performance efficiency and costs.

For example, suppose that a goal of taking 10 passengers from Mirabel to Dorval has been ordered. This may be performed by several private cars or by a single helicopter. However, using a helicopter in such a case will probably cost much more than using private cars and take approximately the same amount of time (due to flight constraints and regulations). Therefore, the most appropriate coalition in such a case is a coalition of private cars. This coalition shall be such that it consists of the optimal number of cars as per the number of passengers. An excess of agents in the coalition may be costly either for the coalition members or for the system as a whole, or both. This is due to the communication, coordination and internal organization costs, both for the formation of the coalition and for its maintenance. These costs are an ascending function of the coalition size. Hence, adding redundant members to a coalition and the resulting inefficient use of the resources are costly for the coalition and for the agent-system.

Suppose now that another order was accepted, according to which 30 suitcases shall be delivered from Mirabel to Dorval. One small truck cannot perform this goal, whereas two small trucks are sufficient. Therefore, a coalition that consists of two small trucks can be formed to perform this goal. However, if some of the private cars which are involved in the previous coalition carry 2 suitcases each, then the remaining suitcases can be delivered by a single small truck. Hence, such an overlapping coalition will be the most beneficial in this case.

The transportation firm may have many agents, and therefore a central authority for coalition formation may be too costly in time and computational efforts. Thus, using a distributed goal allocation mechanism rather than a centralized one may be advantageous. In such cases goal allocation and cooperation shall be decided upon locally. To maximize its benefits, the firm shall attempt to satisfy as many transportation orders as possible. Therefore, the firm shall provide its agents with a simple, but also efficient, algorithm that will enable the formation of coalitions of agents. This algorithm shall be used by the agents in cases where goals cannot be satisfied individually, or when cooperation is more beneficial.

The presented transportation problem is generalized in this paper. We provide algorithms for the allocation of goals to agents via the formation of coalitions. We show that the algorithms are simple to implement, have a short run-time (hence can be used as a real-time method), and yield results which are close to the optimal results.

The environment

We assume that agents can communicate, negotiate and make agreements (Werner 1988). To emphasize the non-super-additive property of the environment with which we deal, we assume that the addition of agents to a coalition is costly, and therefore expanding coalitions may be non-beneficial[3]. Without this assumption, in the case of overlapping coalitions, the grand coalition as a coalition that includes all of the

[3] The source of this additional cost is usually the communication and coordination activities, which grow with the size of coalitions.

overlapping coalitions within it. will form.

Definitions

We recall definitions from (Shehory & Kraus 1995), partially modifying them for the case of overlapping coalitions. $N = \{A_1, A_2, \ldots, A_n\}$, a set of n agents, where A_i has a vector of real non-negative capabilities $B_i = \langle b_1^i, \ldots, b_r^i \rangle$. A capability quantifies the ability to perform a specific action. An evaluation function that transforms the capability units into monetary units is attached to each capability type. $G = \{g_1, g_2, \ldots, g_m\}$ is a set of m goals where each goal g requires $B_g = \langle b_1^g, \ldots, b_r^g \rangle$ for its satisfaction.

There may be occasions where g_j cannot be performed unless g_i has already been satisfied. This is generalized by a partial precedence order between the goals, $g_{1_1} \preceq g_{1_2} \preceq \cdots \preceq g_{1_{n_1}}; \ldots; g_{k_1} \preceq g_{k_2} \preceq \cdots \preceq g_{k_{n_k}}$, where $g_i \preceq g_j$ means that g_i is the predecessor of g_j and g_j is the successor of g_i in the performance order. The precedence order and the resource consumption are the only dependencies that we assume. This restricts our solution to cases where no other explicit dependencies exist (this is appropriate for various domains, including the Blocks' world and several transportation domains).

A coalition can be defined as a group of agents that have decided to cooperate in order to achieve a common goal. We assume that a coalition can work on a single goal at a time, but each agent can be a member of more than one coalition, thus increasing its ability to use its resources for goal satisfaction. A coalition C has a vector of capabilities B_c which is the sum of the capabilities that the coalition members contribute to this specific coalition. Note that this sum is not the sum of all of the capabilities of the members, because agents may contribute their capabilities to more than one coalition. A coalition C can satisfy g only if g has no unsatisfied predecessors and B_g satisfies $\forall_{1 \le i \le r}\ b_i^g \le b_i^C$. Each coalition C has a value V which is the joint utility that the members of C can reach by cooperating via coalitional activity for satisfying a specific goal[4]. The coalitional value V is directly affected by the capabilities that the members of the coalition contribute to it, the precedence order of the goals and the number of members of the coalition. The method according to which the agents decide how to partition their capabilities between coalitions in which they are members, shall be provided in the coalition formation algorithm. To conform with common representation, we may use the coalitional cost $c = \frac{1}{V}$ instead of V. The group rationality (as described below), which leads agents to try to increase V, likewise leads them to try to reduce c.

We assume that the agents are group-rational. That is, they join a coalition only if they benefit as a coalition at least as much as the sum of their personal benefits outside of it (Harsanyi 1977; Rapoport 1970). The agents benefit if they satisfy goals. Group rationality is necessary to assure that whenever agents form a coalition, they increase the system's outcome, which is the sum of the coalitional outcomes. We also assume that each agent tries to maximize the common utility. Group rationality does not necessarily entail a super-additive environment, in which rational agents will prefer the grand coalition (Shapley 1953) over all other coalitions. Our solution is appropriate for non-super-additive environments.

More assumptions are as follows: We assume that the agent-population does not change during the coalition formation process. We do not assume complete information. That is, all of the agents must know about all of the goals and the other agents, however only coalition members must know all of the details required for satisfying a specific goal. In addition, the details of the intra-coalitional activity are not necessary for agents outside of the coalition. We do not assume clock synchronization among the agents. That is, in order to decide which agent will perform what action, the agents do not have to know precisely when other agents will act. Nevertheless, significant time differences between the agents' clocks may reduce the efficiency of the overall goal satisfaction.

Formally, given $G = \{g_1, \ldots, g_m\}$ with an (optional) precedence order and $N = \{A_1, \ldots, A_n\}$ with their capabilities; we are interested in an assignment of goals to coalitions $C_i \subseteq N$ such that $\sum_i V_i$ (the total outcome) is maximal and the precedence order is respected.

Set Covering

Since goal satisfaction by agents may be approached as a problem of assigning goals to coalitions of agents, the partition of the agents into coalitions becomes the main issue. This partition is similar to the set covering problem.

Set covering entails the partition of a set into possibly overlapping subgroups, and the set covering problem is finding such a partition that has a minimal cost. Formally, given $N = \{A_1, \ldots, A_n\}$ a set of elements and $S = \{C_1, \ldots, C_m\}$ a set of subsets of N, such that $C_j \subseteq N$ and $S \subseteq 2^N$: a set-cover is any $S' \subseteq S$ such that $\bigcup_{C_j \in S'} C_j = N$. The members of S' are the covering sets. The cost of a cover S' is $\sum_{C_j \in S'} c_j$ where $c_j > 0$ is the cost of C_j. The set covering problem entails finding the cover with the minimal cost (Balas & Padberg 1975).

The SCP is NP-complete (Cormen, Leiserson, & Rivest 1990). Several algorithms that result in a sub-optimal solutions have been suggested, e.g., (Christofides & Korman 1975; Balas & Padberg 1975; Chvatal 1979). The algorithm of Chvatal (Chvatal

[4]This notion of coalitional value is different from the notion of game theory coalitional value, since it depends on the coalitional configuration and on the goal allocation.

1979), for example, has a logarithmic ratio bound[5]. That is, for any reasonable n, the derived solution is not too far from the optimal one. The implementation of these algorithms for multi-agent coalition formation is problematic: the SCP discusses only a small pregiven set of subsets, and in the case of agents, the number of possible coalitions is 2^n (hence, we need heuristics for reducing this number); agents do not necessarily try to increase the common benefits of the group; the algorithms for SCP are centralized, whereas we seek a distributed algorithm; the SCP algorithms do not refer to cases with precedence order between the chosen subgroups. Despite the deficiencies indicated above, we shall try to borrow some of the properties of the Chvatal algorithm.

Coalition-formation for non-precedence goals

We present below a greedy distributed coalition formation algorithm, based on SCP approximated solutions which therefore has a low ratio bound. It was designed for the special case of autonomous agents in a non-cooperative environment that work as a DPS system (i.e., they try to act in order to increase the performance and the benefits of the system as a whole). The algorithm is an any-time algorithm, i.e., if stopped before normally terminated, it still provides the agents with a solution that is better than their initial state.

The solution of the set covering problem in the case of autonomous agents is exponentially complex due to the number of possible coalitions. A reduction in this number can be achieved either by restrictions of the specific problem under investigation, or via limiting heuristics. In case no specific limitations accrue from the properties of the specific problem, we recommend preferring small sized coalitions. We justify such heuristics by the associated cost-estimation: communication and computation-time are costly; small sized coalitions shall require fewer such operations and therefore shall be more economical to design, form and maintain and hence are preferable. These heuristics are implemented in our algorithm by an integer k which denotes the highest coalitional size allowed or possible. The resulting number of coalitions is $O(n^k)$, which is a polynomial number in n. However, this does not trivialize the problem since even with such restrictions the problem remains NP-complete.

The distribution of the calculations

Prior to calculation of the coalitional values and formation of the coalitions, the agents must distribute the calculations among themselves. Each agent A_i shall conduct the following preliminary steps:

1. Calculate all of the permutations that include up to k agents including A_i.
2. Form a list L_i of the permutations. This is the list of the potential coalitions of A_i.
3. Contact the agents which are members of the potential coalitions in L_i and have not yet contacted you[6].
4. For each agent A_j that you have contacted, perform the following:
   - Locate information about A_j's capabilities (i.e., retrieve B_j).
   - Commit to the calculation of the coalitional values of all of the coalitions in L_i in which both A_i and A_j are members.
5. For each agent that has contacted you, erase from L_i all of the common potential coalitions.
6. To avoid redundant contacts, save a list LC_i of agents that either contacted you or were contacted by you. Avoid contacting the agents on the list LC_i.
7. Repeat contacting other agents until $LC_i = N - \{A_i\}$ (i.e., no more agents to contact).

The calculation of coalitional values

After the preliminary stage, each agent A_i has a list L_i of potential coalitions for which it had committed to repeatedly calculate the values. In addition, A_i has all of the necessary information about the capabilities of the members of these coalitions, and it updates the information if necessary. Now, in order to calculate the values of the coalitions on its list L_i, each agent A_i shall perform the following steps:
Loop and for each coalition C on list L_i, perform:

1. Calculate the coalitional potential capabilities vector B_c^{pc}, by summing up the unused capabilities of the members of the coalition[7]. Formally, $B_c^{pc} = \sum_{A_i \in C} B_i$.
2. Form a list E_c of the expected outcomes of the goals in G when C performs them. $\forall g \in G$, perform:
   - Compare B_g to B_c^{pc}, and thus find the goals that can be satisfied by coalition C.
   - If $\forall i, b_g^i \in B_g \leq b_{pc}^i \in B_c^{pc}$, (i.e., g can be satisfied by C), calculate g's expected outcome e_g with respect to $|C|$, by calculating the monetary values of all of its required capabilities as expressed in B_g, summing them and subtracting the internal coordination costs. This is the expected outcome e_g when coalition C performs g. Put e_g in E_c.
3. The maximal expected outcomes on list E_c will be the coalitional value V_c. Calculate $c_c = \frac{1}{V_c}$.

[5] An approximation algorithm for a problem has a ratio bound $\rho(n)$ if $\rho(n)$ is smaller than the ratio between the optimal cost and the approximated cost.

[6] This can be recognized by the sending time, which can be attached to any message.

[7] Note that this sum is not B_c, the coalitional vector of capabilities.

Having calculated the coalitional values and costs, the agents can proceed to the next stage of the coalition formation procedure.

The construction of coalition configurations

In this stage, in each iteration of the algorithm, the agents decide step-by-step which coalition should be preferred and formed, and the coalitional configuration is gradually achieved. At the end of the first stage of the algorithm, each agent A_i will have a list L_i of coalitions and their values and costs which it had calculated. At the second stage, the agents shall form coalitions. Each agent A_i shall iteratively perform the following:

1. Locate in L_i the coalition C_j that has the smallest cost c_j.
2. Announce the coalitional cost c_j that it has located.
3. Choose the lowest among all of the announced coalitional costs. This c_{low} will be chosen by all agents. The corresponding coalition C_{low} and goal g_{low} shall be selected as well. The value of g_{low} is $\frac{1}{c_{low}}$.
4. If you are a member of the chosen coalition C_{low}, join the other members of C_{low} and form the selected coalition.
5. Erase from G the goal according to which the value of the newly-formed C_{low} has been calculated.
6. Update the capability-vectors of all of the members of C_{low} according to their contribution to the goal-satisfaction.

The above procedures of calculating coalitional values and costs, selecting the preferred coalitions and forming them will be repeated until there are no more goals to be performed or none of the possible coalitions is beneficial. We emphasize that the coalitional values must be re-calculated in every iteration because they are affected by the coalitional configuration. This is because whenever a coalition is formed, a goal is erased from G, the goals' set, and the capability-vectors are updated. This means that all of the coalitional values that were calculated with respect to this specific goal or with respect to the capabilities of the involved agents are no longer correct and must be re-calculated.

Quality assessment

An important property of the algorithm above is its logarithmic ratio bound. Each time a coalition C_j forms, its cost c_j is added to c_{tot}, the total cost of the solution. To express the ratio bound by the same notions as in (Shehory & Kraus 1995), we must define the agent weight w_i for the case of overlapping coalitions. Here, only agents who join a new coalition for the first time are considered for weight calculation. The number of new members in a coalition C_j is denoted by z_j. The agent's weight is the ratio between the coalition cost c_i and z_i, i.e., $w_i = \frac{c_i}{z_i}$. The total cost of the solution c_{tot} is the sum of the weights of all of the agents. In the final coalitional configuration, $\mathcal{C}$ is given by:

$$c_{tot} = \sum_{A_i \in N} w_i \leq \sum_{C_j \in \mathcal{C}^*} \sum_{A_i \in C_j} w_i \qquad (1)$$

where $\mathcal{C}^*$ is the optimal coalitional configuration. Since for any coalition C_j, the sum above is bounded as follows:

$$\sum_{A_i \in C_j} w_i \leq \sum_{A_i \in C_j} \frac{c_j}{|C_j|} = c_j \sum_{A_i \in C_j} \frac{1}{|C_j|} \qquad (2)$$

and

$$\sum_{A_i \in C_j} \frac{1}{|C_j|} \leq \sum_{i=1}^{|C_j|} \frac{1}{i} \qquad (3)$$

We can conclude from (1), (2), and (3) that

$$c_{tot} \leq \sum_{C_j \in \mathcal{C}^*} c_j \sum_{i=1}^{|C_j|} \frac{1}{i} \leq c^*_{tot} \sum_{i=1}^{max(|C_j|)} \frac{1}{i} \qquad (4)$$

Hence, we derive the ratio bound

$$\rho = \frac{c_{tot}}{c^*_{tot}} \leq \sum_{i=1}^{max(|C_j|)} \frac{1}{i} \qquad (5)$$

This ratio bound grows logarithmically with the size of the coalitions. Since we determined the maximum permitted coalitional size, we simultaneously limited the ratio bound to being a constant. Without this limitation, the bound grows to infinity, although very slowly. Note that this ratio-bound is not different from the one presented in (Shehory & Kraus 1995). However, the bound represents the worst case. In the average and best cases, where the overlapping ability improves the resource allocation with respect to the non-overlapping case, the results will correspondingly be better, as simulation results confirm (to be published in a future paper).

Complexity

In addition to the quality of the solution with respect to the optimal solution (given k), the computation and communication complexities shall be examined. The calculation all of the relevant permutations of agents requires $O(n^k)$ computation operations. During the first stage the agents contact one another; each of them contacts up to $n-1$ agents. The average number per agent is $O(1)$ but, considering the worst case, the communication complexity per agent at the coalitional-values calculation stage is $O(n)$.

In each iteration, after the value-calculation, $|G|$ coalitions are formed to satisfy all of the goals[8].

[8] Note that we assume that the planning for partitioning goals into sub-goals is given, and therefore the calculation of coalitional values is not as complicated as in (Sandholm & Lesser 1995).

The overall number of computation operations is therefore of order $O(n^k \cdot |G|)$. Assuming that the number of capabilities depends neither on the number of agents nor on the number of goals, each assignment operation requires $O(1)$ operations. However, the constant here may be large.

Choosing the largest value is of order of the number of coalitions, i.e., $O(n^k)$ computations. The two processes of calculating coalitional values and choosing coalitions may proceed up to $|G|$ times. The resulting communication complexity is $O(n \cdot |G|)$.

To summarize, the computational complexity is of order $O(n^k \cdot |G|)$ and the communication complexity is of order $O(n \cdot |G|)$, both on the part of the agent.

Goals with precedence order

In cases where goals have predecessors, each goal can be satisfied only if all of its predecessors are performed previously. Hence, our algorithm requires that the choice of a goal implies the choice of all of its predecessors. We denote by P_g the set of g and its predecessors. The choice of g will depend on the costs of, and the benefits from, the formation of coalitions that perform *all* of the goals in P_g.

Modifications to the algorithm

In the original problem the agents form coalitions iteratively, i.e., one coalition in each iteration, to satisfy the single goal g that currently appears most beneficial. In the precedence-order case, the agents shall form several coalitions in each iteration, to perform all of the goals in the specific set P_g, which appears most beneficial among all such possible sets. For the evaluation of these sets of goals, we introduce the concept of precedence value pV_g (p-value), the sum of values of the goals in P_g.

As in the previous case, where the value of a goal was re-calculated in each iteration until it was chosen, pV_g will be re-calculated in each iteration as well. Each such calculation requires the calculation of the values of all of the goals in P_g. For this we employ the calculation methods of the non-precedence case, where P_g is substituted into G. The internal precedence order within P_g is not considered regarding this value-calculation since all of the goals within P_g must be satisfied, if g is chosen. The distribution of the calculations among the agents will be performed as suggested in the value-calculation section. In each iteration, the best pV is chosen from among all of the current pV's. This implies that all of the goals in P_g will be satisfied, as well. Since we introduced the notion P_g, we must emphasize its difference from G, which denotes the set of all of the goals that were not yet performed. Initially, G includes all of the goals that must be satisfied by the agent-system.

The agents may either perform all of the calculations concerning each goal $g \in G$ individually, or they may distribute these calculations, but in each step of the algorithm (when necessary) agree upon the P_g to be calculated.

All of the agents, simultaneously, should iteratively perform:

- Given the current status of capabilities: $\forall g \in G$ do
  1. Compute greedily the values of all of the goals in P_g using the distributed methods of the coalition construction section, where the input set of goals for the greedy calculation is P_g.
  2. If, in step 1, all of the goals in P_g were attached a coalition to perform them, calculate pV_g.
  3. Otherwise, remove g from G.
- Choose the goal g^* with the maximal pV to be performed together with all of its predecessors. Form the required coalitions. Remove g^* and all of its predecessors from G and update the capability-vectors of the associated coalitions' members.

The above iterative procedure of calculating p-values and costs, selecting the preferred coalitions for goal satisfaction and forming them will be repeated until there are no more goals to be performed in G.

The modified ratio-bound and complexity

The original ratio-bound calculation is based on the concept of choosing the lowest cost each time and adding it to the total cost. The modifications of the coalitional values calculation do not affect this concept. Therefore, the logarithmic ratio-bound of the original algorithm is not modified, but the ratio-bound of the p-values contributes another logarithm into the ratio-bound, thus yielding:

$$\rho = \frac{c_{tot}}{c^*_{tot}} \leq \log^2 max(|C_j|) \qquad (6)$$

The change in the calculation of coalitional values due to the precedence order increases the computational complexity. $O(|G|^3)$ additional operations are necessary for the calculation of each pV. This shall be performed up to $|G|$ times. The overall additional complexity is $O(|G|^4)$, by which the complexity of the original algorithm shall be multiplied. Thus, the new complexity is $O(n^k \cdot |G|^5)$.

Conclusion

In this paper we presented an algorithm for coalition formation among computational agents where multiple memberships in coalitions are allowed. The algorithm is suitable for cases where the agents are motivated to act in order to maximize the benefits of the system as a whole. It is most appropriate for the situations in which the agents cannot perform goals by themselves. However, it may improve the efficiency of goal-satisfaction when the performance of single agents is lower than their performance within groups. The algorithm is also adjusted to cases in which the goals have

a precedence order, and agents can be involved in the performance of more than one goal.

General task allocation problems are known as (at least) NP-complete problems. We provide a polynomial-complexity algorithm with sub-optimal results. The distribution of calculations is an outcome of the algorithm characteristics, since each agent performs mainly those calculations that are required for its own actions during the process. In cases with no precedence order, this distribution method prevents most of the calculations that may have been repeated by individual agents. However, the last property is less significant in the case of precedence-ordered goals.

The algorithm is an any-time algorithm. In the non-precedence case the results, when halted, are of good quality. In the precedence-order case, better subgroups of goals and coalitions are formed prior to others. However, if the algorithm is halted before the performance of such a subgroup has been completed, the accumulative value of the goals within the subgroup is not necessarily the best. This means that although the any-time property of the algorithm exists for both cases, the case of precedence order may yield less beneficial intervention results. The any-time property of an algorithm is important for dynamic environments, wherein the time-period for negotiation and coalition-formation processes may be changed during the process.

References

Balas, E., and Padberg, M. 1975. On the set covering problem: An algorithm for set partitioning. *Operations Research* 23:74–90.

Bhatia, R.; Khuller, S.; and Naor, J. 1995. The loading time scheduling problem. In *Proc. of the 36th Annual IEEE Conf. of Computer Science (FOCS-95)*.

Chen, G. H., and Yur, J. S. 1990. A branch-and-bound-with-underestimates algorithm for the task assignment problem with precedence constraint. In *Proc. of the 10th international conference on Distributed Computing Systems*. 494–501. France: IEEE Computer Society.

Christofides, N., and Korman, S. 1975. A computational survey of methods for the set covering problem. *Mathematics of Operations Research* 21(5):591–599.

Chvatal, V. 1979. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research* 4(3):233–235.

Coffman, E. G., and Graham, L. R. 1972. Optimal scheduling for two-processor systems. *Acta Informatica* 1:200–213.

Conte, R.; Miceli, M.; and Castelfranchi, C. 1991. Limits and levels of cooperation: Disentangling various types of prosocial interaction. In Demazeau, Y., and Muller, J. P., eds., *Decentralized A.I. - 2*. 147–157. Elsevier Science Publishers.

Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1990. *Introduction to Algorithms*. MIT Press.

Durfee, E. H.; Lesser, V. R.; and Corkill, D. D. 1988. Coherent cooperation among communicating problem solvers. In Bond, A. H., and Gasser, L., eds., *Readings in Distributed Artificial Intelligence*. California: Morgan Kaufmann Publishers, Inc. 268–284.

Harsanyi, J. C. 1977. *Rational Behavior and Bargaining Equilibrium in Games and Social Situations*. Cambridge University Press.

Kahan, J. P., and Rapoport, A. 1984. *Theories of coalition formation*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Ketchpel, S. P. 1994. Forming coalitions in the face of uncertain rewards. In *Proc. of AAAI94*, 414–419.

Rapoport, A. 1970. *N-Person Game Theory*. University of Michigan.

Sandholm, T. W., and Lesser, V. R. 1995. Coalition formation among bounded rational agents. In *Proc. of IJCAI-95*. 662–669.

Sandholm, T. 1993. An implementation of the contract net protocol based on marginal cost calculations. In *Proc. of AAAI-93*, 256–262.

Shapley, L. S. 1953. A value for n-person game. In Kuhn, H. W., and Tucker, A. W., eds., *Contributions to the Theory of Games*. Princeton University Press.

Shehory, O., and Kraus, S. 1995. Task allocation via coalition formation among autonomous agents. In *Proc. of IJCAI-95*, 655–661.

Shehory, O., and Kraus, S. 1996. A kernel-oriented model for coalition-formation in general environments: Implementation and results. In *Proc. of AAAI-96*.

Smith, R. G. 1980. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transaction on Computers* 29(12):1104–1113.

Wang, L., and Tsai, W. 1988. Optimal assignment of task modules with precedence for distributed processing by graph matching and state-space search. *Bit* 28:54–68.

Wellman, M. P. 1992. A general-equilibrium approach to distributed transportation planning. In *Proc. of AAAI-92*, 282–289.

Werner, E. 1988. Toward a theory of communication and cooperation for multiagent planning. In *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*. 129–143.

Zhou, L. 1994. A new bargaining set of an n-person game and endogenous coalition formation. *Games and Economic Behavior* 6:512–526.

Zlotkin, G., and Rosenschein, J. S. 1994. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In *Proc. of AAAI94*, 432–437.