

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

Onn Shehory Sarit Kraus

Department of Mathematics and Computer Science

Bar Ilan University Ramat Gan, 52900 Israel

{shechory, sarit}@bimacs.cs.biu.ac.il

Tel: +972-3-5318863

Fax: +972-3-5353325

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 a waste of 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.

Content area: Cooperation and coordination

*Kraus is also affiliated with the Institute for Advanced Computer Studies, University of Maryland. 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 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, with 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 [20, 11, 13, 15] and in game theory [19, 10] are the concepts of overlapping coalitions and the precedence order of goals.

In [15] a coalition formation procedure for agents in a Distributed Problem Solving system has been presented. There, dynamically alternating inter-related coalitional values was 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 with comparison to the case of disjoint coalitions. We concentrate on environments which are not necessarily super-additive [7].

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. [11, 20]. Most of these solutions address Multi-Agent Systems (MAS), where each agent tries to increase its own personal utility via cooperation [?]. 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 [15]. In addition, our solution is most appropriate for non-super-additive domains. This is the case when adding a new agent to the coalition is costly¹, and as the size of the coalition increases it becomes less beneficial to form it.

In [13], a coalition formation model for bounded-rational agents and a general classification of coalition games. There, the value of a coalition 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.

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

DPS systems in which tasks are allocated to agents have been discussed previously, e.g., the Contract Net Protocol (CNP)[16]. 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 (within coalitions) is required.

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 that agents do not have multiple memberships in coalitions, e.g., [12]. 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.

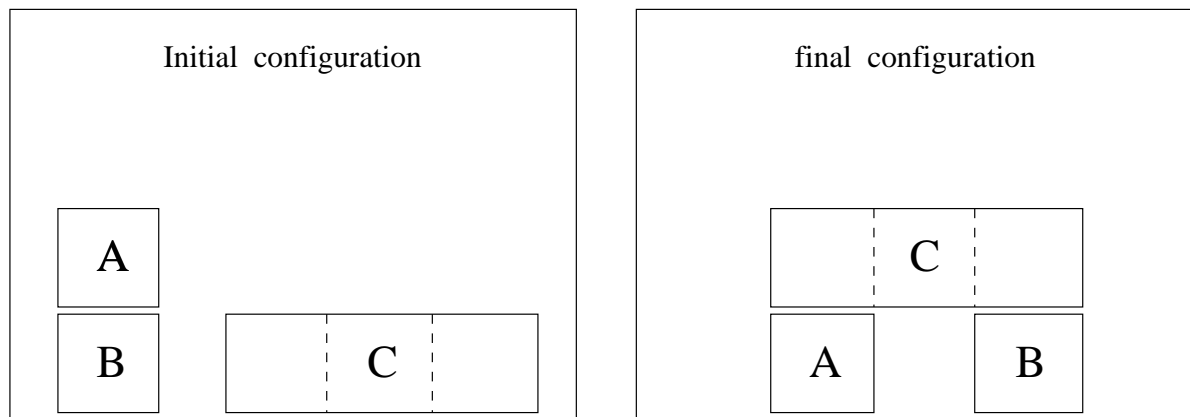
Coalition formation where coalitions may overlap can be approached as a Set Covering Problem (SCP). Exact solutions and approximations to SCP have been proposed in the fields of operations research, combinatorial algorithms, and graph theory [1, 4, 5]. However, the solutions that have been proposed are inappropriate for the problem of coalition formation among agents (see section 5).

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. [6]). The general problem is NP-complete, but some approximation algorithms [17] provide good solutions for the multi-processor system as in [3]. 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 work, 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) [2] 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.

3 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 above. 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 move 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 above. 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 way to many. Obviously, any member of the group that places 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 of function after performing a set of tasks, and therefore they cannot be part of all working groups.

4 The environment

We assume that agents can communicate, negotiate and make agreements [18]. To emphasize the non-super-additivity 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-beneficial². Without this assumption, in the case of overlapping coalitions, the grand coalition as a coalition that includes all of the overlapping coalitions within it will form.

4.1 Definitions

We recall definitions from [15], partially modifying them for the overlapping coalitions case. $N = \{A_1, A_2, \dots, A_n\}$, a set of n agents, where A_i has a vector of real non-negative capabilities $B_i = \langle b_1^i, \dots, b_r^i \rangle$. 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. $G = \{g_1, g_2, \dots, g_m\}$ is a set of m goals where each goal g requires $B_g = \langle b_1^g, \dots, b_r^g \rangle$ 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, $g_{1_1} \preceq g_{1_2} \preceq \dots \preceq g_{1_{n_1}}, \dots, g_{k_1} \preceq g_{k_2} \preceq \dots \preceq g_{k_{n_k}}$, 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 the Blocks' world and similar 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 increase 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 the 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 $\forall_{1 \leq i \leq r}, b_i^g \leq b_i^C$. 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 goal³. 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, in section 6. 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 [9, 12]. The agents benefit if they satisfy goals. Group rationality is necessary to assure that

²The source of this additional cost is usually the communication and coordination activities, which grow with the size of coalitions.

³This 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.

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 [14] over all other coalitions. Our solution is appropriate for non-super-additive environments.

Formally, given $G = \{g_1, \dots, g_m\}$ with an (optional) precedence order and $N = \{A_1, \dots, A_n\}$ 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.

5 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 $N = \{A_1, \dots, A_n\}$ a set of elements and $S = \{C_1, \dots, C_m\}$ 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 minimum cost [1].

The SCP is NP-complete [8]. Several algorithms that result in a sub-optimal solutions have been suggested, e.g., [4, 1, 5]. The algorithm of Chvatal [5], for example, has a logarithmic ratio bound⁴. 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 pre-given 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.

6 Coalition-formation for non-precedence goals

We present below a greedy distributed coalition formation algorithm, based on SCP approximated solutions and 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).

⁴An 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.

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 suggest to prefer 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 shall be preferred. These heuristics is 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.

6.1 The distribution of the calculations

Prior to calculating of the coalitional values and forming 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 all of the agents which are members of the potential coalitions in L_i and did not yet contact you⁵.
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).

⁵This can be recognized by the sending time, which can be attached to any message.

6.2 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 coalition⁶. 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 coalition C performs them. $\forall g \in G$, perform:
 - Compare B_g to B_c^{pc} , 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 $c_c = \frac{1}{V_c}$.

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

6.3 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 $\frac{1}{c_{low}}$.

⁶Note that this sum is not B_c , the coalitional vector of capabilities.

4. If you are a member of the chosen coalition C_{low} , perform:

- Join the other members of C_{low} and form the selected coalition.
- If you are a *new member*, i.e., not a member of any of the coalitions that were formed prior to C_{low} , perform:
 - Inform the other members that you are a *new member*.
 - Count the number of such messages that you have received, add 1 and store in z . This is the number of *new members* in coalition C_{low} .
 - Calculate your agent weight w_i as follows⁷: the agent’s weight is the ratio between the coalition cost and the number of *new members* in it and is given by $w_i = \frac{c_{low}}{z}$.

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.

6.4 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. In the final coalitional configuration \mathcal{C} :

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

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

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

⁷Note that this weight is equal for all of the *new members* in C_{low} . The rationale for the form of w_i is that it weighs the contribution of the *new member* to the system as a whole, with respect to the number of the other *new members* that have joined the current coalition.

and

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

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

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

Hence, we derive the ratio bound

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

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 moderately. Note that this ratio-bound is not different from the one presented in [15]. 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.

6.5 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. 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.

7 Goals with precedence order

In a case that 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 .

7.1 Modifications to the algorithm

In the original problem the agents form coalitions iteratively, 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 , the sum of the 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 sections 6.2 and 6.3, where P_g is substituted into G from 6.3. The internal precedence order within P_g is not considered as to 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 section 6.2. 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 performed yet. Initially, G includes all of the goals that must be satisfied by the agent-system.

The agents may either each perform all of the calculations concerning each goal $g \in G$, 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, in parallel, should iteratively perform:

- Given the current status of capabilities; for each $g \in G$ do
 1. Compute greedily the values of all of the goals in P_g using the distributed methods of section 6.3, 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 as in 6.3. 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 .

7.2 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 yields:

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

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 $O(n^k \cdot |G|^5)$.

8 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 distinct 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 halting 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 anytime 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

- [1] E. Balas and M. Padberg. On the set covering problem: An algorithm for set partitioning. *Operations Research*, 23:74–90, 1975.
- [2] R. Bhatia, S. Khuller, and J. Naor. The loading time scheduling problem. In *Proc. of the 36th Annual IEEE Conference of Computer Science (FOCS-95)*, Wisconsin, 1995.
- [3] G. H. Chen and J. S. Yur. 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*, pages 494–501, France, 1990. IEEE Computer Society.
- [4] N. Christofides and S. Korman. A computational survey of methods for the set covering problem. *Mathematics of Operations Research*, 21(5):591–599, 1975.
- [5] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
- [6] E. G. Coffman and L. R. Graham. Optimal scheduling for two-processor systems. *Acta Informatica*, 1:200–213, 1972.
- [7] R. Conte, M. Miceli, and C. Castelfranchi. Limits and levels of cooperation: Disentangling various types of prosocial interaction. In Y. Demazeau and J. P. Muller, editors, *Decentralized A.I. - 2*, pages 147–157. Elsevier Science Publishers, 1991.
- [8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- [9] J. C. Harsanyi. *Rational Behavior and Bargaining Equilibrium in Games and Social Situations*. Cambridge University Press, 1977.
- [10] J. P. Kahan and A. Rapoport. *Theories of coalition formation*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1984.
- [11] S. P. Ketchpel. Forming coalitions in the face of uncertain rewards. In *Proc. of AAAI94*, pages 414–419, Seattle, Washington, 1994.
- [12] A. Rapoport. *N-Person Game Theory*. University of Michigan, 1970.

- [13] T. W. Sandholm and V. R. Lesser. Coalition formation among bounded rational agents. In *Proc. of IJCAI-95*, pages 662–669, Montréal, 1995.
- [14] L. S. Shapley. A value for n-person game. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games*. Princeton University Press, 1953.
- [15] O. Shehory and S. Kraus. Task allocation via coalition formation among autonomous agents. In *Proc. of IJCAI-95*, pages 655–661, Montréal, 1995.
- [16] R. G. Smith. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transaction on Computers*, 29(12):1104–1113, 1980.
- [17] L. Wang and W. Tsai. Optimal assignment of task modules with precedence for distributed processing by graph matching and state-space search. *Bit*, 28:54–68, 1988.
- [18] Eric Werner. Toward a theory of communication and cooperation for multiagent planning. In *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 129–143, Pacific Grove, California, March 1988.
- [19] L. Zhou. A new bargaining set of an n-person game and endogenous coalition formation. *Games and Economic Behavior*, 6:512–526, 1994.
- [20] G. Zlotkin and J. S. Rosenschein. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In *Proc. of AAAI94*, pages 432–437, Seattle, Washington, 1994.