Task allocation via coalition formation among autonomous agents *

Onn Shehory
Dept. of Math and Computer Science
Bar Ilan University
Ramat Gan, 52900 Israel
shechory@bimacs.cs.biu.ac.il

Sarit Kraus
Dept. of Math and Computer Science
Bar Ilan University
Ramat Gan, 52900 Israel
sarit@bimacs.cs.biu.ac.il

Abstract

Autonomous agents working in multi-agent environments may need to cooperate in order to fulfill tasks. Given a set of agents and a set of tasks which they have to satisfy, we consider situations where each task should be attached to a group of agents which will perform the task. The allocation of tasks to groups of agents is necessary when tasks cannot be performed by a single agent. It may also be useful to assign groups of agents to tasks when the group's performance is more efficient than the performance of single agents. In this paper we give an efficient solution to the problem of task allocation among autonomous agents, and suggest that the agents will form coalitions in order to perform tasks or improve the efficiency. We present a distributed algorithm with a low ratio bound and with a low computational complexity. Our algorithm is an any-time algorithm, it is simple, efficient and easy to implement.

1 Introduction

Autonomous agents working in multi-agent environments may need to cooperate in order to fulfill tasks. Given a set of agents and a set of tasks which they have to satisfy, we consider situations where each task should be attached to a group of agents which will perform the task. The allocation of tasks to groups of agents is necessary when tasks cannot be performed by a single agent. It may also be useful to assign groups of agents to tasks when the group's performance is more efficient than the performance of single agents. Groups of agents may have different efficiency in task performance due to differing capabilities of the members of different groups. The task allocation will be done with respect to these differences.

The purpose of the allocation of tasks to groups of agents is to perform all of the tasks and to do so in an efficient way. We seek an algorithm that will enable such task allocation in a distributed manner, i.e., without any central authority. An important property of any given solution is a low computational complexity. Therefore, in this paper we present an algorithm that will enable the agents to form groups and assign a task to each group. We call these groups coalitions.

Distributed artificial intelligence (DAI) is concerned with problem solving in which several agents interact in order to achieve goals. During the past few years, several solutions to the coalition formation problem have been suggested by researchers in the field of DAI. These solutions concentrate on the special case of autonomous agents in a super-additive environment [1] [Shechory and Kraus, 1993; Ketchpel, 1994; Zlotkin and Rosenschein, 1994]. The solutions are given for coalition formation in Multi-Agent Systems (MAS), where each agent tries to increase its own personal utility via cooperation. Most of these solutions are based on concepts from game theory. One of the main problems of coalition formation in the case of MAS is how to distribute the common outcome of a coalition among its members. We intend to present a coalition formation algorithm which is appropriate for Distributed Problem Solving (DPS) cases where agents cooperate in order to increase the outcome of the system. In such cases, the disbursements to the agents are not as important as they are in MAS. In addition, we do not restrict the solution to the super-additive environment. Rather, we suggest a solution which is most appropriate for the non-super-additive case.

Distributed task allocation has been discussed in the context of DPS systems. A well-known example of such a task allocation mechanism is the Contract Net Protocol (CNP)[Smith, 1980]. This work discusses cases in which an agent that attempts to satisfy a task may divide it into several sub-tasks and sub-contract each subtask to another agent via a bidding mechanism. In the CNP, tasks are allocated to single agents and a procedure for task-partitioning is necessary. Contrary to these cases, we solve the problem of assigning tasks to groups of agents. Such a solution is necessary in cases where single agents cannot perform tasks by themselves and tasks cannot be partitioned, or the partition is computationally too complex. In the CNP the efficiency of the

*This material is based upon work supported in part by the NSF under Grant No. IRI-9123460 and the Israeli Science Ministry grant No. 6288. Kraus is also affiliated with the Institute for Advanced Computer Studies, Univ. of Maryland.

[1] In a super-additive environment any combination of two groups of agents into a new group is beneficial. For details see section 3.

solution was evaluated through simulations. We provide a formal analysis of the complexity of our solution.

Coalition formation is an important method of cooperation in multi-agent environments. Game theory describes which coalitions can form in N-person games under different settings and how the players will distribute the benefits of the cooperation among themselves, e.g., [Neumann and Morgenstern, 1947; Rapoport, 1970; Kraus and Wilkenfeld, 1991; Luce and Raiffa, 1957]. Game theory does not provide algorithms which agents can use in order to form coalitions. Given a previously formed coalitional configuration, game theory usually concentrates on checking its stability or its fairness[2]. Game theory rarely takes into consideration the special properties of a multi-agent environment. That is, the communication costs and limited computation time are not considered, and the solutions are not distributed. In our case, we are particularly interested in the coalition formation mechanism and how it will be distributed. We also seek a dynamic evaluation of the coalitions, where game theory usually provides a static evaluation. Therefore, the game-theoretic coalition formation theories are not appropriate for the multi-agent situation with which we are concerned.

The problem of coalition formation can be approached as a Set Partitioning Problem (SPP). Set partitioning entails the partition of a set into subsets, and the set partitioning problem is finding such a partition that has a minimal cost (for details see section 3.2). Since coalition formation of agents results in the partition of the agents into subgroups, SPP may be an appropriate way for determining which coalitions will form as a result of a coalition formation algorithm. The SPP and the Set Covering Problem (SCP)[3] , have been dealt with widely in the context of NP-hard problems [Garey and Johnson, 1979]. Exact solutions and approximations to SPP and SCP have been proposed in the fields of operations research, combinatorial algorithms, and graph theory [Garfinkel and Nemhouser, 1969; Balas and Padberg, 1972; 1975; Christofides and Korman, 1975; Chvatal, 1979]. However, the solutions that have been proposed do not provide an appropriate solution to the problem of coalition formation among agents, due to three main deficiencies: 1) the exact and optimal solutions are solutions for NP-hard problems. That is, the complexity of the solution is exponential in the number of agents. Such a solution cannot be applied in cases where there are many agents in the environment, since the agents will be unable to calculate it; 2) the approximated solutions are of polynomial complexity, but they deal with a tightly restricted number of pre-defined possible subgroups of a given set. This restriction on the subgroups contradicts the fact that there are 2^n subsets of a given set of size n, and therefore makes such solutions inappropriate for coalition formation among agents, unless the possible coalitions will be artificially limited; 3) all of the present solutions are centralized. That is, the solutions can be calculated and implemented only by a central agent which dominates the coalition formation process and supervises the other agents. Such a case is not typical in distributed agents' environments, and is inappropriate for our case.

In this paper we combine the combinatorial algorithmic approach and concepts from operations research with autonomous agents' methods in order to form a task allocation method that employs a coalition-formation procedure. The resulting coalitions of this procedure are beneficial for systems of agents. We will concentrate on environments which are not necessarily super-additive [Conte *et al.*, 1991; Harsanyi, 1963].

We begin by illustrating the problem we intend to solve (section 2). Then, we give a brief description of the environment with which we deal in section 3. We also briefly present the basic definitions and assumptions in section 3.1. The algorithm is described in section 4, and its complexity is analyzed in section 4.4. Section 5 ends our paper with a discussion and conclusions.

2 Illustration of the problem

The problem we solve in this paper is that of task allocation among groups of autonomous agents in a DPS system. Given a set of tasks, the system as a whole has to satisfy all of the tasks, or at least seek the satisfaction of as many tasks as possible, thus maximizing its benefits. In our case there is no central authority that distributes the tasks among the agents. The agents shall reach an efficient task allocation by themselves, seeking a maximal outcome.

An example is a transportation company[4]. The company supplies transportation services via a system of autonomous and automated trucks, lift trucks, cranes, boats and planes which we call agents. This system is usually constructed in a distributed manner, since every single agent may perform limited tasks by itself. The agents differ in their capabilities. That is, they differ in the type of actions that they can perform, in the size, volume and weight of goods that they can carry at one time, in the transportation speed, its costs and the method by which it is performed. There may be occasions where agents cannot perform a given transportation task by themselves. In such cases, cooperation is necessary. Therefore, the agents shall form groups, and each group of agents will cooperatively fulfill a transportation task. We call such cooperating groups 'coalitions'. Obviously, not every coalition can perform any given task, and among those that are able to perform a given task, the efficiency and the costs may be completely different.

For example, suppose that a transportation task of delivering 10,000 flowers from New Jersey to New York City has been ordered. Such a task may be performed

[2]Stability and fairness have several different definitions in the context of game theory.

[3]The SCP, which is very similar to the SPP, is described in section 3.2.

[4]Transportation systems have been extensively used as examples for DPS systems (e.g., [Sandholm, 1993]). We do not intend to solve a specific transportation problem. However, we provide this example because the reader is familiar with it. More interesting examples (such as distributed car-pooling; for non-distributed car-pooling, some commercial implementations are already present) are too long for this paper.

by several trucks or by a single helicopter. A lift-truck is necessary in both cases. However, using a helicopter in such a case will probably cost much more than using trucks and will take approximately the same length of time (due to flight constraints and regulations). Therefore, the most appropriate coalition in such a case is a coalition of trucks and lift-trucks, with a size that would be appropriate for the freight size. This is because an overhead of agents in a coalition may prevent the formation of other beneficial coalitions and therefore may reduce the total outcome of the system.

If the transportation company has many agents, a distributed task allocation mechanism may be advantageous (as discussed in section 4.4). In such cases task allocation and cooperation shall be decided upon locally. However, such a company seeks the maximization of its benefits. Therefore, the company shall attempt to satisfy every transportation order. This is so because such success will bring an immediate profit to the company, and will also result in a satisfied client, which is important for the future. Since a single agent cannot always satisfy the client's order, close cooperation is necessary. For such cases, the company shall provide the agents with a simple but also efficient algorithm that will enable the formation of coalitions of agents.

The problem of the transportation company is generalized in this paper. We provide an algorithm which enables the allocation of tasks among a system of agents via the formation of coalitions. We show that the algorithm is simple to implement, has a short run-time (hence can be used as a real-time method), and yields results which are close to the optimal results.

3 Environment description

In order to elucidate the problem and its solution, we briefly present some general notations and definitions for concepts. We assume that agents can communicate, negotiate and make agreements [Werner, 1988]. Communications require time and effort on the part of the agents. We also assume that some resources can be transferred between agents. This ability may help the agents form more beneficial coalitions.

3.1 Definitions

There is a set of n agents, $N = \{A_1, A_2, \ldots, A_n\}$. Each agent A_i has a vector of real non-negative capabilities $B_i = \langle b_1^i, \ldots, b_r^i \rangle$. Each capability is a property of an agent that quantifies its ability to perform a specific action. For example, in the case of the transportation company, the volume and weight that can be transported by an agent is its transportation-volume capability. In order to enable the assessment of coalitions and of task-execution, an evaluation function shall be attached to each type of capability. Such a function shall transform the capability units into monetary units. In the transportation case, this function may be the income from shipping a freight. This may be a linear function in the size of cargo. We also assume that there is a set of m independent tasks[5] $T = \{t_1, t_2, \ldots, t_m\}$. For the satisfaction of each task t_j, a vector of capabilities $B_j = \langle b_1^j, \ldots, b_r^j \rangle$ is necessary.

A coalition can be defined as a group of agents that have decided to cooperate in order to achieve a common task. We assume that a coalition can work on a single task at a time, and that each agent is not a member of more than one coalition. A coalition C has a vector of capabilities $B_c = \sum_{A_i \in C} B_i$ (which is a sum over vectors). Coalition C can perform a task t only if the vector of capabilities necessary for its fulfillment B_t satisfies $\forall\ 0 \le i \le r\ , b_i^t \le b_i^C$. For each coalition C a value V can be calculated which is the joint utility that the members of C can reach by cooperating via coalitional activity for satisfying a specific task[6]. The coalitional value V is directly affected by the capabilities of the members of the coalition. Recall the transportation company. Given a specific transportation task, if the sum of transportation-volumes of a coalition is less than the volume necessary for the task, then the value of the coalition for this specific task is zero. If the coalitional transportation volume is much larger than necessary for satisfying the task, then the value is positive, but relatively small in comparison to the case of having the exact volume. Actually, the transportation-volume is not the only capability of the agents and therefore does not affect the coalitional value exclusively.

For reasons of convenience, we may sometimes employ the notion of coalitional cost c instead of coalitional value. This cost may be calculated as the reciprocal of the coalitional value. Such a calculation attaches a low cost to a high-valued coalition and vice versa. The coalitional rationality (as described below), which leads agents to try to increase the coalitional value, likewise leads them to try to reduce the coalitional cost.

We assume that the agents are coalitionally rational. 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 [7]. The agents benefit if they fulfill tasks. Coalitional rationality is necessary to ensure that whenever agents form a coalition they always increase the system's common outcome, which is the sum of the coalitional outcomes. We also assume that each agent tries to maximize the common utility; among all the possibilities that an agent has, it will choose the one that will lead to the maximum common utility. However, coalitional rationality of the agents does not necessarily entail a super-additive environment. In order to emphasize the difference between the super-additive environment and the ones we deal with, we describe the super-additive environment below.

A super-additive environment is such that the set of the possible coalitions satisfies the following rule: for

[5]The partition of a single task into subtasks is beyond the scope of this paper. However, if these subtasks are independent, then the algorithm that we suggest can be used recursively to allocate sub-groups of agents to the subtasks.

[6]This notion of coalitional value is different from the notion of game theory coalitional value, since here the value depends on the coalitional configuration and on the task allocation.

[7]This assumption is usually called "group rationality" in the game theory literature [Harsanyi, 1977; Rapoport, 1970; Luce and Raiffa, 1957].

each pair of coalitions C_1,C_2 in the set, $C_1 \cap C_2 = \emptyset$, if C_1,C_2 join together, then the newly formed coalition will have a new value $V_C^{new} \geq V_C^1 + V_C^2$, where V_C^1,V_C^2,V_C^{new} are the values of the coalitions. Rational agents in a super-additive environment will prefer the grand coalition over all other coalitions. A grand coalition is a coalition that includes all of the agents [Shapley, 1953]. Hence, in a super-additive environment, when the grand coalition is formed, the only problem that still remains is how the utility should be distributed among its members. This is a serious problem in MAS. However, in DPS systems such a problem is of lesser importance or may not exist at all.

Now that we have made the necessary assumptions and definitions, we can formally present the problem. Given a set of m tasks $T = \{t_1, \ldots, t_m\}$, a set of n agents $N = \{A_1, \ldots, A_n\}$ with their capabilities, the problem we solve is of assigning a group of agents $G \subseteq N$ to each task $t_i \in T$ such that $\bigcup G_i = N$ and $\forall i \neq j, G_i \cap G_j = \emptyset$ and $\sum_i V_i$ (the total outcome) is maximal. We solve the problem for the general case of non-super additive environments. However, even in the case of a super-additive environment, the solution will consist of coalitions G_i that do not have null[8] members. This is because the membership of null members in a coalition prohibits their membership in other coalitions and may thus prevent maximization of $\sum_i V_i$.

3.2 Set Covering and Set Partitioning

Since task allocation among agents may be approached as a problem of assigning groups of agents to tasks, the partition of the agents into subgroups becomes the main issue. Therefore, the task allocation problem becomes more similar to the set partitioning and the set covering problems.

Below we formulate the two problems: SCP and SPP. Given a set $N = \{A_1, \ldots, A_n\}$ and a set of subsets of N, $S = \{C_1, \ldots, C_m\}$, 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. If the members of S' are also pairwise disjoint (i.e., $\forall\ C_i, C_j \in S',\ i \neq j\ \ C_i \cap C_j = \emptyset$), then S' is a set-partitioning of N. We assume that each $C_j \in S$ has a positive cost c_j. The cost of a cover S' is $\sum_{C_j \in S'} c_j$. The set covering problem entails finding the cover with the minimum cost, and the set partitioning problem is defined correspondingly [Garfinkel and Nemhouser, 1969; Balas and Padberg, 1972; 1975].

The set covering problem is NP-complete [Cormen *et al.*, 1990]. This implies a computational complexity which is, in practice, too high. However, a variety of heuristic algorithms for solving this problem have been suggested, e.g., [Christofides and Korman, 1975; Balas and Padberg, 1975; Chvatal, 1979]. Among them we can find the algorithm of Chvatal [Chvatal, 1979], which has a logarithmic ratio bound [9]. That is, for any reasonable n, the derived solution is not too far from the optimal one. Given an algorithm with such a low ratio bound, it is very tempting to adopt it and implement it for the case of multi-agent coalition formation. Unfortunately, this cannot be done – the algorithm provides a solution to the set covering problem, in which subgroups may overlap. Such a situation is not allowed in the case of coalitions of agents; even if we do allow agents to be members of more than one coalition (i.e., overlapping subgroups) there still remain other problems. These include the following: the set covering problem deals only with a small 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 (or decrease the costs of the group), and may be self-motivated and try to increase their own benefits (even if such selfish behavior decreases the common benefits); the algorithms for SCP and SPP are all centralized and, since we deal with autonomous agents, we seek a distributed algorithm. Despite the deficiencies indicated above, we shall try to borrow some of the properties of the Chvatal algorithm, thus constructing a coalition formation algorithm with a low ratio bound.

4 The algorithm

The algorithm we present below is a greedy distributed set-partitioning algorithm with a low ratio bound. It was designed for the special case of autonomous agents in an environment which is not necessarily super-additive, and that work as a DPS system (i.e., they try to act in order to increase the performance and benefits of the group as a whole). Another important property of the algorithm we provide is that it is an any-time algorithm. That is, if the execution is stopped before the algorithm would have normally terminated, it still provides the agents with a solution which is better than their initial state or any other intermediate state.

The algorithm will be constructed from three main stages (which we present in detail later):

1. In the first stage of the algorithm the coalitional values shall be calculated. This is done because any efficient coalition formation algorithm requires information about the values of the possible coalitions, in order to be able to choose the preferred ones.
2. The second stage of the algorithm entails an iterative greedy process through which the agents decide upon the preferred coalitions and form them.
3. Finally, the benefits of the cooperation may be disbursed among the agents. This stage is not always necessary since we deal with DPS systems. However, if the disbursement of the outcome of the task-execution is necessary, then we can provide some fair methods for doing so.

As previously stated, the solution of the set partitioning problem in the case of autonomous agents is of an exponential complexity, since the number of the possible coalitions is exponential (2^n). Therefore, any attempt

[8] Null members are agents that contribute nothing to the coalitional performance.

[9] 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.

to reduce the complexity of such a solution shall include a reduction of the number of permitted coalitions. This can be done via the constraints of the specific problem under investigation, e.g., it may happen that all of the tasks must be performed by the same number of agents. If the problem itself does not provide any constraints, then the reduction can be done via heuristics. We suggest adopting the following heuristics: since communication and computation-time have costs, and the agents seek cost-reduction, they should try to avoid unnecessary communication and computational activities. Therefore, small-sized coalitions shall be preferred as more economical to design than larger coalitions. This is the case since the calculations and communication operations are exponentially dependent on the numbers of members in a coalition. These heuristics will be implemented in our algorithm by presenting an integer k which will denote the highest coalitional size allowed. This restriction will turn the number of coalitions into a polynomial number in n. In the transportation case, a limitation on the size of coalitions may be even more reasonable than in the general case. Here, it would be very convenient to assume that the volume of a single freight task never exceeds a given size, which is related to k, although it may sometimes be more efficient to perform the task by using a larger coalition. Such a restriction affects the number of coalitions in the same way as in the case of communication and computation restrictions. More information about the properties of the tasks and the coalitional values may enable the calculation of the expectation values of the outcome of different coalitions, and improve the heuristics we employ, thus reducing the number of coalitions and the complexity of the algorithm.

The initial coalitional state consists of n, single agents. The agents then begin negotiating [Kreifelts and Martial, 1990] and, step by step, form coalitions. Agents that join coalitions quit the coalition-formation process, and only the remaining single agents will continue negotiations. The reduction in the number of agents that continue negotiating reduces the computational and communication costs. At the beginning of this coalition formation process, each agent will calculate the values of coalitions in which it is a member.

4.1 Distributed calculation of coalitional values

As stated above, the first stage of the algorithm entails the calculation of the coalitional values in a distributed manner. In order to decide which coalitional values to calculate, each agent will perform the following steps:

1. Calculate all of the possible coalitions up to size k in which you are a member and form a personal list of coalitions.
2. For each coalition in the personal list, contact each member and ask for its task-performing capabilities.
3. Inform the agent whom you have approached that you are committed to the calculation of the coalitional values of the coalitions in which you are both members.
4. Construct a personal list of agents that you have approached and avoid repeated approaches to the same agents.
5. In case you were approached by another agent and it had committed to the calculation of the values of the common coalitions, erase all of your common coalitions from your personal list of coalitions.
6. Repeat the contacting of other agents until you have none to approach.

At this stage, the agent has a list of coalitions for which it had committed to calculate the values. It also has all of the necessary information about the capabilities of the members of these coalitions. Now, in order to calculate the values, each agent shall perform the following steps:

1. Check which capabilities are necessary for the execution of each task $t_i \in T$. Compare them to the capabilities of the members of the coalition, thus finding the tasks that can be performed by the coalition.
2. Calculate the expected outcome of the tasks that can be performed by the coalition. For each task perform the following:
    - First, calculate the monetary values of all of its capabilities and sum them.
    - Then, calculate the monetary values of the capabilities of the coalition which are not used for the fulfillment of the task and sum them[10].
    - Subtract the second sum from the first. This will be the expected outcome of the task.
3. Among all of the expected outcomes, choose the maximal one. This will be the coalitional value.

Having calculated the coalitional values, the agents can proceed to the next stage of the process.

4.2 Choosing coalitions

In the second stage of the algorithm, the preferred coalitions are chosen and the coalitional configuration is gradually achieved. In order to simplify the representation of the algorithm, we denote the ratio between the cost of the coalition and the coalition's size by $w_i = c_i / \mid C_i \mid$ and call it the coalitional weight. At the end of the first stage of the algorithm, each agent will have calculated a list of coalitions and their values. Each agent will choose the best coalition from among its list, i.e., the coalition C_i that has the smallest w_i. Next, each agent will announce the coalitional weight that it has chosen, and the lowest among these will be chosen by all agents. The members of the coalition that was chosen will be deleted from the list of candidates for new coalitions. In addition, any possible coalition from the lists of any agent, that includes any of the deleted agents, will be deleted from its list.

[10] Note that this calculation is done in order to reduce the value of coalitions whose capabilities do not precisely fit the capabilities necessary for task fulfillment. However, this reduction method was not tested for its quality, and other methods of such a reduction may be considered once the algorithm is implemented.

The procedures of calculating coalitional values and choosing the preferred coalitions will be repeated until all agents are deleted (that is, until all are assigned to coalitions), or until there are no more tasks to be allocated, or none of the possible coalitions is beneficial. The coalitional values shall be calculated repeatedly since they are affected by the coalitional configuration. This is because each value is calculated subject to the tasks that should be performed. Any change in the coalitional configuration means that a task was assigned to a coalition, so this specific task no longer affects the coalitional values which may previously have been affected by it. Therefore, the coalitional values that have been calculated with reference to a task that has just been allocated must be re-calculated. All other values remain unchanged.

4.3 The ratio bound

The algorithm we presented above has several advantages. One of the most important properties of the algorithm is its low logarithmic ratio bound. Each time a coalition C_j is added to the coalition configuration, the cost c_j of this coalition is added to the total cost of the solution. Each member of C_j receives a weight w_j, and the total cost of the solution is:

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

where $\mathcal{C}^*$ denotes the final coalitional configuration. The right equality is due to the fact that the order of summation does not change the result, and both summations are over all agents. However, since for any coalition C_j

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

and

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

then from (1), (2), and (3), we conclude that

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

hence, we derive the ratio bound

$$\rho = \frac{c_{total}}{C^*_{total}} \leq \sum_{i=1}^{max(|C_k|)} \frac{1}{i} \qquad (5)$$

This ratio bound grows logarithmically with the size of the coalitions to which the algorithm refers. Since we previously suggested presenting a constant to determine the maximum permitted coalitional size, we simultaneously limited the ratio bound to being a constant that can be simply calculated using (5).

4.4 Complexity of the algorithm

The efficiency of this algorithm should be judged from two main perspectives: computations and communications. The first step of the coalitional-values calculation will be the calculation of all of the relevant coalitions. This requires $\sum_{i=1}^{k}(n-1)!/((n-i-1)!i!)$ computation operations, which is of an average order $O(n^{k-1})$ per agent. Since there are n agents, each agent will approach up to $n-1$ agents. The average number of such communications is 1. However, considering the worst case, the communication complexity per agent at this stage is $O(n)$.

The value-calculation process will proceed with the assignment of tasks to coalitions. Since such an assignment is done for all tasks, it shall be done m times. The number of such operations per agent is of order $O(n^{k-1} \cdot m)$. However, while some agents perform $O(n^{k-1} \cdot m)$ value-calculations, others may happen to perform less than $O(n^{k-1} \cdot m)$ calculations. This property of the process is advantageous because it occurs when there are differences of computational capabilities among the agents. In such cases the non-equal partition of the calculations moderates the differences in the calculation-time of the agents, thus reduces the average time of calculation completion. Assuming that the number of capabilities depends neither on the number of agents nor on the number of tasks, each assignment operation requires $O(1)$ operations. However, the constant here may be large.

Choosing the largest value is of the order of the number of coalitions, i.e., $O(n^{k-1})$ computations per agent. The two processes of calculating coalitional values and choosing coalitions may proceed up to $n-1$ times in the worst case, where all of the coalitions are of single agents. The average case is much smaller, but still the communication and computational complexity will be $O(n \cdot m)$.

To summarize, the average computational complexity is of order $O(n^{k-1} \cdot m)$ and the communication complexity is of order $O(n \cdot m)$, both for each agent. This complexity can be compared to the centralized case, where a single agent performs all of the operations. In such a case, this single agent will experience $O(n^k \cdot m)$ computations and $O(n)$ communications. Therefore, the computational complexity is higher than in the average distributed case (the speed-up is of order $O(n)$), but the communicational complexity is lower[11].

5 Discussion

In this paper we presented an algorithm for task allocation among computational agents via coalition formation in a non-super-additive environment. The algorithm is suitable for cases where agents are motivated to act in order to maximize the benefits of the system as a whole. It is most appropriate for the incidents in which the agents cannot perform the tasks by themselves. However, it is also important for improving the efficiency of task execution when tasks can be performed by single agents. This may be the case when the performance of single agents is worse than their performance within groups.

Although the general task allocation problem is computationally exponential, we bring a polynomial-

[11] If, for all agents, there exists a communication channel between every pair of agents, then the computational overhead of the distributed case will not affect the performance of the algorithm.

complexity algorithm that yields results which are close to the optimal results and, as we have proven, are bounded by a logarithmic ratio bound. Another advantage of the algorithm, which is crucial in the case of a distributed system, is the distribution of the algorithm. We distribute the calculations in a natural way. That is, the distribution is an outcome of the algorithm characteristics since each agent performs mostly those calculations that are required for its own actions during the process. In addition, our distribution method prevents most of the possibly overlapping calculations, thus saving unnecessary computational operations.

The algorithm is an any-time algorithm. If halted before normally terminated, it still provides the system with several coalitions that have already formed. Since the first coalitions to be formed are the better ones, the results, when halted, are still of good quality. The any-time property of such an algorithm is important for dynamic environments, wherein the time-period for negotiation and coalition-formation processes may be changed during the process.

Due to space limitations, we neither expand the examples nor provide the complete details of the algorithm. We also do not provide a method for distributing the outcome of the task-execution among the agents. These topics shall be presented in a full-length paper.

To conclude, we suggest that designers of dynamic systems of distributed computational agents, wherein tasks shall be assigned to agents in order that the agents will perform them, use our algorithm in order to maximize the expected outcome of the system as a whole and to minimize the necessary run-time.

References

[Balas and Padberg, 1972] E. Balas and M. Padberg. On the set covering problem. *Operations Research*, 20:1152–1161, 1972.

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

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

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

[Conte *et al.*, 1991] 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.

[Cormen *et al.*, 1990] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-completeness.* W. H. Freedman and Company, New York, 1979.

[Garfinkel and Nemhouser, 1969] R. S. Garfinkel and G. L. Nemhouser. The set-partitioning problem: set covering with equality constraints. *Operations Research*, 17:848–856, 1969.

[Harsanyi, 1963] J. C. Harsanyi. A simplified bargaining model for n-person cooperative game. *International Economic Review*, 4:194–220, 1963.

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

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

[Kraus and Wilkenfeld, 1991] S. Kraus and J. Wilkenfeld. Negotiations over time in a multi agent environment: Preliminary report. In *Proc. of IJCAI-91*, pages 56–61, Australia, 1991.

[Kreifelts and Martial, 1990] T. Kreifelts and F. Von Martial. A negotiation framework for autonomous agents. In *Proc. of the Second European Workshop on Modeling Autonoumous Agents in a Multi Agent World*, pages 169–182, France, 1990.

[Luce and Raiffa, 1957] R. D. Luce and H. Raiffa. *Games and Decisions.* John Wiley and Sons, Inc, 1957.

[Neumann and Morgenstern, 1947] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, N.J., 1947.

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

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

[Shapley, 1953] 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.

[Shechory and Kraus, 1993] O. Shechory and S. Kraus. Coalition formation among autonomous agents: Strategies and complexity. In *Proc. of MAAMAW-93*, Neuchâtel, 1993.

[Smith, 1980] 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.

[Werner, 1988] 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.

[Zlotkin and Rosenschein, 1994] 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.