Feasible formation of coalitions among autonomous agents in non-super-additive environments *

Onn Shehory
The Robotics Institute
Carnegie-Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213, USA
onn@ri.cmu.edu
Tel: +1-412-268-3740
Fax: +1-412-268-5569

Sarit Kraus
Dept. of Math and Computer Science,
Bar Ilan University,
Ramat Gan, 52900 Israel
sarit@cs.biu.ac.il
and
Institute for Advanced Computer Studies,
Maryland University, College Park, MD USA

Abstract

Cooperating and sharing resources by creating coalitions of agents are an important way for autonomous agents to execute tasks and to maximize payoff. Such coalitions will form only if each member of a coalition gains more if it joins the coalition than it could gain otherwise. There are several ways of creating such coalitions and dividing the joint payoff among the members. In this paper we present algorithms for coalition formation and payoff distribution in non-super-additive environments. We focus on a low-complexity kernel-oriented coalition formation algorithm. The properties of this algorithm were examined via simulations. These have shown that the model increases the benefits of the agents within a reasonable time period, and more coalition formations provide more benefits to the agents.

Key Words Distributed AI, Coalition Formation, Multi-Agent Systems.

*This material is based upon work supported in part by the NSF under grant No. IRI-9423967, ARPA/Rome Labs contract F30602-93-C-0241 (ARPA Order No. A716) and the Army Research Lab under contract No. DAAL0197K0135. A preliminary version of this paper appears in the proceedings of AAAI-96.

Contents

1 Introduction

Cooperation among autonomous agents may be mutually beneficial even if the agents are self-interested and try to maximize their own expected payoffs [21, 29]. An important method for cooperation in multi-agent systems is coalition formation. Membership in a coalition may increase the agent's ability to satisfy its goals and maximize its own personal payoff. Moreover, there may be occasions where the formation of coalitions is the only or most beneficial method for satisfying the goals. There are situations where agents can cooperate by forming one coalition in which all of the agents are members (grand coalition), however in environments where the cost of cooperation is an ascending function in the number of the cooperating agents, the partition of the agents into subgroups will reduce their costs (since coordination and communication come about only within the subgroups). This paper addresses its solutions to this type of environments, to which game theory refers as non-super-additive environments (e.g., in [22]).

Work in game theory such as [28, 38, 42, 44] describes which coalitions will form in N-person games and how the players will distribute the benefits of the cooperation among themselves, but does not provide algorithms for coalition formation nor does it consider the specific constraints of a multi-agent environment, such as distribution, communication costs and limited computation time. Our paper presents a multi-agent approach to the coalition formation problem and, in particular, we devise distributed coalition-formation procedures of two types. The first is computation-oriented (i.e., includes extensive computation and minor negotiation) and the second is negotiation-oriented (computation is vastly reduced, and negotiation activity is increased).

The need for two different mechanisms stems from the trade-off between the quality of solutions and the speed in which they can be reached. One aspect of the quality is the sum of the payoffs the agents receive. In this respect, when forming coalitions, it is necessary to search an exponential number of possible solutions in order to guarantee a solution within a bound from the best solution, in terms of payoff maximization (as shown in [30]). Another aspect of quality is the stability of the solution. Stability refers to a state where, once a stable solution is found, the agents have no incentive to search for other solutions (or little incentive in weaker types of stability). To guarantee stability of the types commonly discussed in the context of coalitions in the game theory literature, it is necessary (in most of the cases) to search an exponential number of solutions. Our first mechanism, the computation-oriented one, leads to the formation of stable[1] coalitions and to maximization of the individual payoffs of the agents. It searches an exponential solution space and can guarantee a bound from

[1]There are several different stability concepts, each depends on assumptions and regulations which vary from one concept to the other. We concentrate on a kernel-oriented stability, as defined later in this paper (section 3). This choice is for practical reasons, nevertheless proves to be efficient as we later show.

optimum (in addition to other advantages). When attempting to reduce computation time the search must be limited to a polynomial number of possible solutions. In such a case a bound from optimum on the sum of payoffs is not guaranteed. Our second mechanism, the negotiation-oriented one, leads to stability of the coalitions, efficiency of the coalition formation process, and is also an anytime algorithm. While satisfying the requirement for a polynomial search (to reduce computation time), in its worst case the payoffs the agents receive are not within a bound from the optimal payoffs. Nevertheless, as we show via simulations, the average case payoffs are close to optimal.

The mechanisms provided in this paper facilitate coalition formation among autonomous agents, each of which has tasks it must fulfill, and access to resources that it can use to fulfill these tasks. Coalition formation is utilized to enhance the ability of the agents to satisfy tasks cooperatively. Such an environment is the taxi-driver domain, where autonomous computational agents are taxi-driver representatives. The taxi drivers may own different cabs, thus having different costs, transportation capabilities and resulting payoffs. Cooperation and coalition formation may increase their ability to achieve greater and more complex transportation capabilities. Examples where cooperation and coalition formation are employed to solve AI problems are the transportation domain [31], the electricity plants domain [7, 16], and the information agents domain [19]. In particular, coalition formation methods developed in this paper serve as the basis for coalition formation among information agents, as presented in [20].

The main contribution of this paper is the two algorithms it provides, to be used by self-interested agents in a non-super-additive environment (presented in sections 4 and 5 of this paper). No such algorithms where previously devised for such environments, that consider inseparably issues of optimal payoffs, stable solutions and distribution, given the limitations on communication and computation in multi-agent systems. To the algorithms we add prerequisites, testing and discussions. The outline of the paper is as follows. We first introduce basic concepts such as protocols, strategies and equilibrium (sections 2.1 and 2.2) and define a specific type of equilibrium, a *time-bounded equilibrium*. Protocols, strategies and equilibrium are later used throughout the paper. We follow in section 2.3 with assumptions regarding the coalition formation environment and definitions of coalition concepts which we borrow from game theory and modify to fit the specific properties of our problem. Next, we introduce the kernel stability concept (section 3), upon which our coalition formation mechanisms in the subsequent sections are based. In section 4 we introduce a computation-extensive coalition formation mechanism which supports Pareto-optimality[2]. We devise a distributed algorithm and provide details (section 4.2), methods of computation (in section 4.3) which are used for both this algorithm and the next one, and complexity

[2]Broadly speaking, Pareto-optimal solutions maximize the payoffs given a specific coalition configuration.

analysis (in section 4.4). In section 5 we introduce the second algorithm, for which we develop the concept of polynomial kernel (section 5.1). We provide the details of the algorithm (in section 5.2) as well as protocols and strategies to complement this algorithm (section 5.4) and a complexity analysis of it (section 5.5). The latter algorithm was simulated, and the results of this simulation are presented in section 6. Related work, both in DAI and in game theory, is discussed in section 7, and in section 8 we close the paper with discussion and conclusions.

2 Environment description

2.1 Protocols and Strategies

Research in distributed AI (DAI) is divided into two basic classes: *Cooperative Distributed Problem Solving (CDPS)* and *Multi-Agent Systems (MAS)* [4]. Research in CDPS considers how the efforts required for solving a particular problem can be distributed among a number of modules or "nodes." Research in MAS is concerned with coordinating intelligent behavior among a collection of autonomous heterogeneous intelligent (possibly pre-existing) agents. In MAS global control, globally consistent knowledge, and globally shared goals or success criteria may not exist. These classes are actually the two extreme poles in DAI research spectrum. Our research falls closer to the MAS pole since it deals with interactions among *self-interested*, *rational* and *autonomous* agents. However, any interaction among agents requires some protocols. As more protocols are enforced upon the agents, the amount of communication required to reach a beneficial agreement usually decreases. Yet the protocols may be contradictory to the rationality of an individual agent.

In environments of heterogeneous agents, protocols must be agreed upon and enforced by the designers. For this, deviation from the protocols must be revealable and penalizable, or protocols must be self-enforced. The designers of agents will agree upon protocols, provided that they are not advantageous to any particular type of agent, and leave enough opportunity for the individual agents to utilize their resources and strengths. In coalition-formation, the need for protocols increases further. As was noted by Shapley and Shubik [34], in situations where not every cooperative demand by every coalition can be satisfied, some constraints must be placed on coalitional activity lest it be trapped in an endless loop of rejected suggestions for coalition formation.

Protocols are necessary to regulate the interaction between the agents, however they allow the agents to choose specific strategies that will enable use of their strengths to improve their individual outcomes. For example, by defining the concepts of coalition and coalitional value (section 2.3), we limit the possible results of the coalition formation algorithms. The proposed strategies allow the individual agents to fulfill their tasks and to increase their own benefits. For example, the method that agents employ to handle proposals (see section 5.4.2) is a strategy (and not a protocol) since agents can use the provided strategies however are not forced to use them. Nevertheless, these strategies can increase the payoff of the individual agent and satisfy a time-bounded equilibrium requirement (as defined below).

2.2 Equilibrium

There is a variety of definitions of equilibrium. Most commonly used is the Nash equilibrium [23], according to which a strategy profile p is in equilibrium if each entity, either a single agent or a coalition, maximizes its own expected utility with respect to its own strategy in p, given that the other entities follow their strategies in p. Searching the whole strategy space of the coalition formation process and computing the expected utilities from the outcomes of following these strategies may be intractable. Therefore, when computations are limited to polynomial time (as is usually the case in multi-agent systems), the computation of Nash equilibria may be infeasible. Furthermore, avoiding the excess time required for search and information evaluation will leave the entities with resources needed for task execution. Recognizing this, game-theory researchers have two approaches to proceed with [12]. One is to insist that the cost of obtaining and processing information (e.g., computing the expected utility of a given action) be incorporated into the model. The main drawback of this approach is the vast increase in the complexity of the model, which is likely to be analytically intractable. The second approach, that of bounded rationality, employs different notions of equilibria (e.g., epsilon-equilibria [27]), as approximations and/or rules of thumb for decision making by the entities. Such models can preserve greater analytical manipulability, but always seem somewhat unsatisfactory, because one can find ways that allow entities to use the information they have in order to make better decisions.

We propose to combine these two approaches by defining the concept of *time-bounded equilibrium*, as follows: a strategy profile p is in time-bounded equilibrium if each entity believes, given the information it obtained using a pre-defined time, or equivalently, a depth of search, that it maximizes its expected utility with respect to its strategy in p given its *bounded computation time* and that the other entities follow their strategies in p.

In particular, computing the expected utility of a possible outcome of a given action will be done w.r.t the time the entity believes[3] is beneficial to allocate to it. Given an entity's beliefs of its possible actions and its expected utility when taking these actions, which are computed given the bounded time available, the entity acts rationally, i.e., it uses strategies which are in equilibrium.

[3]Note that we implicitly assume that agents have a preference for acquiring some benefits (not necessarily maximal) over acquiring no benefits at all. Engaging in coalitions will provide the participants with benefits, whereas additional efforts to compute better strategies might leave them with no agreement and no benefits.

The rationale of the time-bounded equilibrium in coalition formation environments is as follows: since computations are costly and the value of a coalition is finite, the benefits of an entity when forming a coalition may decrease as a result of the computation time spent on the negotiation and formation processes. These benefits may be nullified at some point in time if the process is not accomplished within a reasonable amount of computation. Although the devaluation details are not necessarily known to the agents, their designers should have either information or assumptions with regards to this devaluation. By setting the computation time or the depth of search in the strategy space, the designers reduce the risk of the avoidance of beneficial coalition formation. The latter occurs when the benefits from forming the coalitions are lesser than the cost of excessive strategy-search.

2.3 Assumptions and definitions

The following definitions and assumptions are necessary for our coalition formation approach. We assume:

- Information about each agent's resources, tasks and payoff functions is accessible to the other agents at the beginning of the cooperation phase. The access to this information may be costly[4]. Information about the dynamics of the coalition formation is not necessarily accessible.
- Agents can communicate to negotiate and make agreements. However, communications are costly.
- There is a monetary system (e.g, money or utility points) that can be used for side-payments. The agents use this monetary system to evaluate resources and productions.
- Resources and money are transferable among the agents (although agents may reach agreements and form coalitions even if money is non-transferable [21]).

Below we modify definitions from classical game-theory coalition formation (e.g., [17]) and adjust them to the MAS requirements. Consider $N = \{A_1, A_2, \ldots, A_n\}$, a group of n autonomous agents. The agents are provided with, or have access to, resources. $\forall A_i \in N$ exists a resource vector $\rho_i = \langle \rho_i^1, \rho_i^2, \ldots, \rho_i^l \rangle$, where ρ_i^j is the quantity of resource j of A_i's. Agents use resources to execute tasks. A_i's outcome from task-execution is expressed by a payoff function $U^i : \Theta \to \mathcal{R}$ that exchanges resources into monetary units (Θ – resource domain, $\mathcal{R}$ – reals). Each agent attempts to maximize its personal payoff.

[4]This assumption is reasonable for cases in which the agent-system consists of dozens of agents, and we address the algorithms that we present later to such cases. Algorithms for coordination among hundreds of agents can be found in [37].

Example 2.1 *A specific taxi-drivers domain consists of four taxi-drivers: Ann (A), Barbara (B), Christie (C) and Debbie (D), each owns a black-taxi (a resource) and located in Victoria station (VS), London (see map in section 4.3.1, figure 1). A black-taxi can carry up to 4 passengers. The taxi-drivers' costs are 30 pence per mile (only whole miles are considered), and an insurance fee of 1 pound per day per taxi or 1.5 pounds per day per two taxis, plus 1 penny processing fee for more than 2 taxis*[5]*. The drivers' gross income is composed of a 60-pence base rate per trip and 80 pence per mile*[6] *(denoted by m). At the end of every trip the drivers return to VS. The profit p from one trip is $p = 0.6 + (0.8 - 0.3)m = 0.6 + 0.5m$. The daily profit is the sum of the profits of single trips minus the insurance costs (which depends on cooperation among the drivers). Coalition formation may increase each driver's transportation capabilities and their benefits, however not all coalitions are equally beneficial, and once some coalitions have formed it may be non-beneficial to enlarge them. Thus, the taxi-driver domain is not super-additive because of the insurance fees and it satisfies the assumptions above.*

A coalition is a group of agents that have decided to cooperate and how the total benefit should be distributed among them. Formally:

Definition 2.1 Coalition *Given a group of agents N and a resource domain Θ, a coalition is a quadruple $C = \langle N_C, \rho_C, \overline{\Theta}, U_C\rangle$, where $N_C \subseteq N$; $\rho_C = \langle \rho^1, \rho^2, \ldots, \rho^l\rangle$ is the coalition's resource vector, where $\rho^j = \sum_{A_i \in N_C} \rho_i^j$ is the quantity of resource j that the coalition has. $\overline{\Theta}$ is the set of resource vectors after the redistribution of ρ_C among the members of N_C ($\overline{\Theta}$ satisfies $\rho^j = \sum_{A_i \in N_C} \overline{\Theta}_i^j$). $U_C = \langle u^1, u^2, \ldots, u^{|N_C|}\rangle$ is the coalitional payoff vector, where $u^i \in \mathcal{R}$ is the payoff of agent A_i after the redistribution of the payoffs*[7]*.*

In our work we assume no differential tendency for certain coalitions based on friendship, animosity etc. Such tendencies may significantly affect the resulting coalitions. Each coalition is attached a value (as in game theory [22]), and a function for calculating this value.

Definition 2.2 Coalition Value *Let $C = \langle N_C, \rho_C, \overline{\Theta}, U_C\rangle$. $V(C)$ is the value of C if the members of N_C can together reach a joint payoff V. That is, $V(C) = \sum_{A_i \in N_C} U^i(\overline{\Theta}_i)$, where U^i is the payoff function of agent A_i and $\overline{\Theta}_i$ is its resource-vector after redistribution in C.*

This coalition value enables the agents to evaluate coalitions. Note that the specific distribution of the resources among the members of the coalition strongly affects the results

[5]This fee, which is reasonable since larger groups require more processing, results in non-super-additivity.

[6]For pricing a trip, the driving back to VS is considered. For instance, if a taxi has to take passengers for a 4 miles trip, the number of miles for pricing the trip will be $2 \times 4 = 8$.

[7]The notation U_C is used for the payoff distribution within a coalition C. Later, we use another notation U without the subscript $_C$ for payoff distributions to all agents, and not within a specific coalition.

of the payoff functions, thus affecting the total coalitional payoff. In the case of a singleton coalition of an agent A_i, the coalition value $V(A_i)$ is the agent self-value (since the resources are uniquely defined). For a specific coalition C, $V(C)$ is unique because $\overline{\Theta}$ is unique. A proper resource redistribution $\overline{\Theta}$ will result in the maximization of $V(C)$. The complexity of computing $\overline{\Theta}$ and $V(C)$ depends on the type of payoff functions of the coalition members. Linear payoff functions allow polynomial computation, whereas non-linear functions may require approximation[8] methods to allow for a polynomial computation time. Thus, the so-called hidden complexity of value-calculation should not hinder coalition formation[9]. Within this framework we assume that the value of a given coalition does not depend on the other coalitions that are formed. The calculation of coalitional values is essential for coalition formation, however we view it as a separate research topic, and will present results of its investigation in future research.

In game theory $V(C)$ depends only on N_C whereas here, it depends on $\overline{\Theta}$ as well[10]. Negotiation in a distributed environment requires agreement upon a unique coalitional value for each N_C. Proper protocols can force the agents to calculate $\overline{\Theta}$ such that there is a unique $V(C)$ for each N_C. This may hold both for the case in which $V(C)$ is maximized (this maximization is beneficial for all agents) and for the case of value calculation under time constraints [32].

Assumption 1 Coalition joining (individual rationality)
An agent joins a coalition only if it can benefit at least as much within the coalition as it could benefit by itself. An agent benefits if it fulfills tasks, or receives a payoff that compensates it for the loss of resources or non-fulfillment of some of its tasks. This assumption is usually called "individual rationality" in game theory [28, 1].

Coalition-formation usually requires disbursement of payoffs among the agents. We define $U = \langle u^1, u^2, \ldots, u^n \rangle$, a payoff vector to all of the agents, u^i is the payoff to A_i. U is different from the above defined U_C, which is a payoff vector of a specific coalition. In each stage of the coalition formation process, the agents are in a coalitional configuration which is a set of coalitions $\mathcal{C}= \{C_i\}$, that satisfies: $\cup_i C_i = N$; $\forall C_i, C_j$; $C_i \neq C_j$; $C_i \cap C_j = \emptyset$. A pair of a payoff vector and a coalitional configuration are denoted by $PC(U,\mathcal{C})$, or just PC (Payment Configuration). Since we assume individual rationality of the agents, we consider

[8]Methods for approximated value calculations, using methods from finance, are part of an on-going, unpublished research.

[9]Note that when the number of agents is small, small deviations from the exact coalitional values may modify the results of the coalition formation mechanisms, since each value has a more significant effect on the stability of the whole configuration. In such cases, however, the exact values can be calculated to avoid these modifications, even when computations are complex.

[10]Although this dependency is important, for ease of deliberation we do not explicitly present the resources in the rest of this article. Nevertheless, they are used both in the examples and in the calculation of values in the simulation.

only individually rational payment configurations (which are denoted as IRPCs in game theory, e.g.,[25]).

We define a coalitional configuration space (CCS) and a payment configuration space (PCS). A (rational) CCS is the set of all coalitional configurations $\mathcal{C}$ such that $\forall C_i \in \mathcal{C}, V(C_i) \geq \sum_{A_j \in C_i} V(A_j)$. A PCS is a set of possible individually-rational PCs. That is, a PCS consists of pairs $(U, \mathcal{C})$ where U is an individually-rational payoff vector and $\mathcal{C} \in$ CCS. For each $\mathcal{C} \in$ CCS there is a set $\{U\}$ of payoff vectors. The magnitude of $\{U\}$ may sometimes be infinite, thus the PCS may be an infinite space as well. However, representing payoffs in payoff vectors by ranges (e.g., $x_i = [x_i^1, x_i^2]$), allows for a finite set of such vectors.

Lemma 1 CCS size
The magnitude of the CCS is of order $O(n^n)$.

The proof is straightforward using standard combinatorial methods.

A payoff vector is Pareto-optimal if no other payoff vector dominates it, i.e., no other payoff vector is better for some of the agents and not worse for the others. A specific Pareto-optimal payoff vector is not necessarily the best for all of the agents. There can be a group of Pareto-optimal payoff vectors where different agents prefer different payoff vectors. Therefore, Pareto-optimality is insufficient for the evaluation of possible coalitions. Hence, we present the concept of stability.

The issue of stability was studied in game theory in the context of n-person games (e.g., in [28]). While we emphasize the development of protocols for coalition formation, the game theorists concentrate mainly on which stable coalitions can form. However, the notions of stability they developed are useful for our purposes, when coalitions are formed during the coalition formation procedure. The members of such coalitions can apply these techniques to the distribution of the coalitional value. Game theorists have given several solutions for n-person games, with several related stability notions. Each of the stability notions is motivated by a different method of measuring the relative strengths of the participating agents. In this paper we concentrate on the Kernel solution concept.

3 The Kernel K

The kernel [8] is a stability concept for coalitional configurations. For each coalitional configuration in the CCS, it provides a stable payoff distribution. The kernel does not provide the coalitional configuration itself, nor does it provide a method for the agents to move from one coalitional configuration to another. In fact, the kernel does not even provide a method to compute what are the payments to the agents which are stable. It only provides a method to test the stability given specific coalitional configuration and payoff vector. Clearly, it is not a coalition formation algorithm, nor does it include the dynamism necessary for such an algorithm. However, the stability provided by the kernel is this important property which we seek when forming coalitions, and therefore it is incorporated into our solution.

The kernel is a PCS in which each coalitional configuration PC is stable in the sense that any pair of agents A_i, A_j which are members of the same coalition C in PC are in equilibrium with one another. A_i, A_j are in equilibrium if they cannot outweigh one another from C, their common coalition. Agent A_i can outweigh A_j if A_i is stronger than A_j, where strength refers to the potential of agent A_i to successfully claim a part of the payoff of agent A_j in PC. The details and the formal definition of the kernel are provided below.

During the coalition formation, agents can use the kernel solution concept to object to the payoff distribution that is attached to their coalitional configuration. This objection will be done by agents threatening to outweigh one another from their common coalition. The objections that agents can make given a $PC(U, C)$ are based on the excess concept. We recall the relevant definitions.

Definition 3.1 Excess *The excess [8] of a coalition C with respect to the coalitional configuration PC is defined by $e(C) = V(C) - \sum_{A_i \in C} u^i$, where u^i is the payoff of agent A_i in PC. C is not necessarily a coalition in PC, and it can be in any other coalitional configuration. $V(C)$ is the coalitional value of coalition C as in definition 2.2.*

The number of excesses is an important property of the kernel solution concept. Given a specific PC, the number of the excesses with respect to the specific coalitional configuration is 2^n (since there is one excess for each coalition, and there are 2^n coalitions). Changes in the payoff vector U may result in changes in the set of excesses, thus require recalculation of all of the excesses.

Agents use the excesses as a measure of their relative strengths. Since a higher excess correlates with more strength, rational agents must each search for its highest excess (with respect to a specific PC). This maximum is defined by the surplus (see details in, e.g., [28], p. 126).

Definition 3.2 Surplus and Outweigh *The maximum surplus S_{ij} of agent A_i over agent A_j with respect to a PC is defined by $S_{ij} = MAX_{C|A_i \in C, A_j \notin C} e(C)$, where $e(C)$ are the excesses of all of the coalitions that include A_i and exclude A_j, and the coalitions C are not in PC, the current coalitional configuration. Agent A_i outweighs agent A_j if $S_{ij} > S_{ji}$ and $u^j > V(A_j)$, where $V(A_j)$ is the coalitional value of agent A_j in a single agent coalition.*

In other words, given a coalitional configuration and a payoff vector, the agents compare their maximum surpluses, and the one with the larger maximum surplus is stronger. The stronger agent can claim a part of the weaker agent's payoff in the same coalitional configuration, but this claim is limited by individual rationality which requires $u^j > V(A_j)$. This means that in any suggested coalition, agent A_j must receive more payoff than it gets by itself in a single-member coalition. Two agents A_i, A_j that cannot outweigh one another are in equilibrium, which holds if one of the following conditions is satisfied:

1. $S_{ij} = S_{ji}$;
2. $S_{ij} > S_{ji}$ and $u^j = V(A_j)$;
3. $S_{ij} < S_{ji}$ and $u^i = V(A_i)$.

Using the concept of equilibrium, the kernel and its stability are:

Definition 3.3 Kernel and K-Stability
A PC is K-stable if $\forall A_i, A_j$ *agents in the same coalition* $C \in PC$*, the agents* A_i, A_j *are in equilibrium. A PC is in the kernel iff it is K-stable.*

The kernel stability concept provides a stable payoff distribution for *any* coalitional configuration in the CCS. It does not provide coalitional configurations – it merely determines how the payoffs will be distributed given a coalitional configuration (thus, it is not a coalition formation algorithm). Using this distribution, the agents can compare different coalitional configurations. However, checking the stability does not direct the agents to a specific coalitional configuration. The coalition formation model that we develop will perform this direction. The kernel, as well as the bargaining set [2], are solutions that always exist for all of the coalitional configurations in the CCS [8]. Other solution concepts may sometimes be empty (e.g., the core). The kernel has advantages over the bargaining set. One property of the kernel is that symmetric agents receive equal payoffs. That is, agents with the same bargaining strength (which is expressed by identical sets of excesses) will gain equal payoffs. Designers of agents should prefer such symmetry since, in a case where their agent's strength is equal to that of another agent, they would like it to receive at least the same payoff as the other agent does. Such symmetry is not guaranteed in other solution concepts (e.g., the bargaining set). Another more important property of the kernel is its magnitude. The kernel and the bargaining set are subsets of the PCS of all rational PCs, but the kernel, which is a subset of the bargaining set, is significantly smaller[11].

In addition, the mathematical formalism of the kernel allows one to divide its calculation into small polynomial non-related processes, thus simplifying it even in the general exponential case. Some exponentially-complex computing schemes for the kernel solution have been provided, e.g., by Stearns [40], which presented a transfer scheme that, given a coalitional configuration and an arbitrary payoff vector, converges to an element of the kernel. This transfer scheme is based on a possibly infinite number of iterations[12]. In each iteration,

[11]In many cases one can explicitly show that the size of the kernel is smaller than that of the bargaining set by at least one order of magnitude. However this relation is not always true, since there are some specific cases in which these two sets coincide.

[12]Later in this paper we present a modification of Stearns transfer scheme. This modification enables fast convergence in finite time, given a specified error.

agents with smaller maximum surpluses pass part of their payoff to agents with bigger maximum surpluses in the same coalition and thus, by reducing the differences between their surpluses, approach equilibria. The method is presented in detail in [40] and convergence is proven.

Due to its advantages, we selected the kernel as a basis for our coalition formation model. However, since the kernel is a subset of the bargaining set, an agent that does not use the kernel as the protocol for proposal preparation and evaluation, but uses, for instance, the bargaining set instead, will still fit into the kernel world of the other agents. That is, its bargaining-set-oriented proposals can successfully be evaluated by the other agents, using the kernel as a protocol.

There are more solution concepts for n-person games (for details we refer the reader to game theory literature such as [1]). We find other concepts less adequate for allowing distributed coalition formation among self-motivated agents in non-super-additive multi-agent environments. Therefore, we concentrate on the kernel.

4 The Pareto-optimal coalition formation model

In this section we present a Distributed, Exponential, Kernel-oriented Coalition Formation Model (DEK-CFM) that leads to a PC which is Pareto-optimal and k-stable. The DEK-CFM is strongly based on the kernel solution concept. The calculation of the kernel has an exponential complexity, and therefore the complexity of the model that we present in this section will be exponential. In cases where time, communications and computation are cheap or costless, or in cases where there is a small number of agents, an exponential coalition formation model (DEK-CFM) is adequate.

4.1 The Pareto-optimal approach

For computing the Pareto-optimal PCs we introduce the concept of local Pareto-optimality:

Definition 4.1 Local Pareto - Optimality
A payoff vector $U_p = \langle u_p^1, u_p^2, \ldots, u_p^n \rangle$ is locally-Pareto-optimal in a set of payoff vectors S, $S = \{U_1, U_2, \ldots, U_k\}$, if there is no other payoff vector $U_o = \langle u_o^1, u_o^2, \ldots, u_o^n \rangle$ in S, such that $\forall 1 \leq i \leq n$, $u_o^i \geq u_p^i$ and $\exists i$, $u_o^i > u_p^i$.

The local-Pareto-optimality may be verbally modified to express special cases. for example, personal-Pareto-optimality means that S is the set of vectors that was calculated by a single agent; pairwise-Pareto-optimality means that S was calculated by a pair of agents. An important property of the PCs in the kernel is that for each specific coalitional configuration $\mathcal{C}$, any $PC = (\mathcal{C}, U)$ which is in the kernel is locally-Pareto-optimal in the set of the PCs with the same $\mathcal{C}$, as proved in the following lemma:

Lemma 2

Given a specific coalitional configuration $\mathcal{C}$ and let $PC_{\mathcal{C}} = \{PC_i(\mathcal{C}, U_i)\}$, $PC_i \in K$ be the set of all PCs that consist of a payoff vector U_i and the specific coalitional configuration $\mathcal{C}$. $\forall j$, if $PC_j \in PC_{\mathcal{C}}$ then PC_j is locally-Pareto-optimal within $PC_{\mathcal{C}}$.

Proof: $\forall i$, $PC_i \in K$ implies that the sum of payoffs that the agents receive within each coalition in $\mathcal{C}$ is equal to the value of the coalition. Therefore, the sum of all of the elements of each $U_i \in \{U\}$, where $\{U\}$ is the set of all payoff vectors that correspond to the coalitional configuration $\mathcal{C}$ is equal to the sum of the coalitional values of all of the coalitions in $\mathcal{C}$. Hence, there cannot be a payoff vector $U_k \in \{U\}$ such that all of the payoffs to all of the agents are all bigger than all of the corresponding payoffs in another payoff vector $U_j \in \{U\}$. This is because if there was such a vector U_k, then the sum of its elements would be greater than the sum of the elements of U_j, which is in contradiction to the equality that the membership in the kernel implies. Therefore, the PCs are locally-Pareto-optimal. $\square$

While an element in the kernel is locally-Pareto-optimal with respect to other PCs with the same coalitional configuration, it is not necessarily locally-Pareto-optimal with respect to other coalitional configurations. There may be two elements in the kernel (with different coalitional configurations) where one is better for all of the agents. We demonstrate this property in the following example:

Example 4.1 *Consider a 4-agent non-super-additive environment, where $N = \{X, Y, Z, W\}$. Coalitional values are as follows: any single agent: 5, $V(XY) = 20$, $V(XZ) = 50$, $V(YZ) = 10$, $V(XW) = 10$, $V(YW) = 50$, $V(ZW) = 20$, $V(XYZ) = V(XYW) = V(XZW) = V(YZW) = 100$, $V(XYZW) = 60$. Let us consider the coalitional configuration $\{XY, ZW\}$. A possible corresponding K-stable payoff vector is $\langle 10, 10, 10, 10\rangle$. It is locally-Pareto-optimal with respect to all of the payoff vectors that correspond to the coalitional configuration $\{XY, ZW\}$. Refer to $\mathcal{C}$= $\{\{XY, ZW\}, \{XZ, YW\}, \{XW, YZ\}\}$, in which coalitions are of size 2. Possible corresponding K-stable payoff vectors are $\langle 10, 10, 10, 10\rangle$, $\langle 25, 25, 25, 25\rangle$, and $\langle 5, 5, 5, 5\rangle$, respectively. Among these PCs, the second is better for all agents. Therefore, $\langle 10, 10, 10, 10\rangle$ is not locally-Pareto-optimal in $\mathcal{C}$.*

4.2 The DEK-CFM protocol

In MAS, distributed cooperation models are sought. Therefore, we present a distributed, kernel-based coalition formation framework, attempting to minimize computations and communications overheads. The distribution of the DEK-CFM must be strictly regulated. Therefore, several elements of the model are part of the protocol and only a few methods of computation are strategies. The steps of the protocol are as follows:

Protocol 4.1 Distributed protocol for coalition formation

1. *Each agent should compute all of the coalitions in which it is a member, calculate the corresponding coalitional values and transmit them to all of the other agents*[13].

2. *Each agent* A_i *is assigned a unique random integer* $z_i \in [1, n]$ *via a distributed random method, e.g., [3].*

3. *Each agent* A_i *should compute coalitional configurations that consist of* z_i *coalitions*[14].

4. *For each of the computed coalitional configurations,* A_i *should find a K-stable PC.*

5. *From among its computed PCs,* A_i *should construct a list of personally-Pareto-optimal PCs to be supplied to other agents in the next step.*

6. *The agents should merge their personal PCs lists into one list. Among the PCs in this joint list, the Pareto-optimal PCs will be found. The merging process will be done according to the following:*

   - *The merging process will be performed through a sequence of iterations. In each iteration* j*, each agent* A_i *such that* $z_i \bmod 2^j = 1$ *will merge its locally-Pareto-optimal PCs list with the agent* A_k *where* $z_k = z_i + 2^{j-1}$ *(and* $z_i + 2^{j-1} \leq n$*).*
   - *After each merge, the locally-Pareto-optimal PCs with respect to the merged list will be found and held by the agent* A_i *with* $z_i \bmod 2^j = 1$*.*
   - *A merge iteration will terminate after all of the agents have been approached. This will happen after* $\lceil \lg_2 n \rceil$ *iterations.*

7. *The agents should choose, via a decision making method (see section 5.4.1), one of the PCs from the list that was constructed in the previous steps. According to this chosen PC, the agents will form coalitions.*

8. *The agent who designed the chosen PC and the agents who were involved in the choice of this PC must transmit to all of the other agents the details of the calculations that led to this specific PC.*

9. *A deceitful PC can be revealed and canceled upon the received calculations. In case of a cancellation, the whole process will be repeated.*

[13] Given the coalitional utility function and the resources of the coalition, the computed value of the coalition is unique as discussed in section 2.3, page 9. Therefore, agents that calculate values of the same coalitions will yield the same results. This property allows for checking values calculated by other agents thus preventing deceitful values.

[14] Agents can calculate other configurations as well without being monitored. However, the protocol allows the agents to transmit only the z_i-component configurations, and this can be monitored.

Above, we put some effort into reducing the complexity of the merging process from linear to ligarithmic. This may seem of little benefit since the overall complexity of the mechanism is exponential. However the merit of this reduction is not in reducing computation but in reducing communication, which is the main overhead of the distributed approach. In MAS which operate on networks, communication tends to be extremely slow as compared to computation. Hence, a reduced communication overhead is a desirable property.

Protocols are laws that should be incorporated into the agents by their designers. However, their enforcement consists of penalties for cases in which designers fail to perform, or avoid, this incorporation, thus allowing their agents to deviate from the protocols. Therefore, deviation of agents from the protocols must be revealable by others. The above protocol agrees with this requirement. For example, according to step 1 of protocol 4.1, all of the values except those of single agents will actually be calculated by more than one agent. Therefore, deceitful values will be revealed. Assuming a penalty for deceit, agents should avoid deceitful value calculations. In addition, the PC calculations of a winning PC are revealed to all and can be checked. Thus, deceitful PCs are canceled and therefore should be avoided.

K-stable PCs can be calculated using algorithm 1 in section 4.3. This algorithm, however, may be replaced by any other algorithm that will result in K-stable PCs with reasonable computational efforts. K-stability is complicated to calculate and much easier to check. Therefore, agents should not deviate from the calculation of K-stable PCs. This requirement can be achieved if, as part of the protocol, there will be a positive probability that the K-stability of a PC is checked by other agents.

The employment of a stochastic method to decide which agent will receive what random number z, leads to equal expected values for the calculational efforts (although calculations are not equally partitioned). Prior to the calculation of all of the configurations of size z_i, agent i cannot predict which configuration will provide itself with the largest payoff. Therefore, an agent is motivated to calculate all possible configurations of size z_i. In addition, the z_i-component configuration can easily be checked, thus deviation is revealable.

Completion of the DEK-CFM provides the agents with a resulting K-stable and Pareto-optimal PC. The distribution of the calculations speeds-up computation (since each agent performs only part of it) but does not change the computational complexity. Also, additional communication operations stem from this distribution. The requirement that agents transmit calculated configurations to others during the coalition formation process, as well as the termination of the process within $\lg_2 n$ iterations, result in an upper limit of the number of required transmissions of configurations which is $O(\lg_2 n)$ per agent.

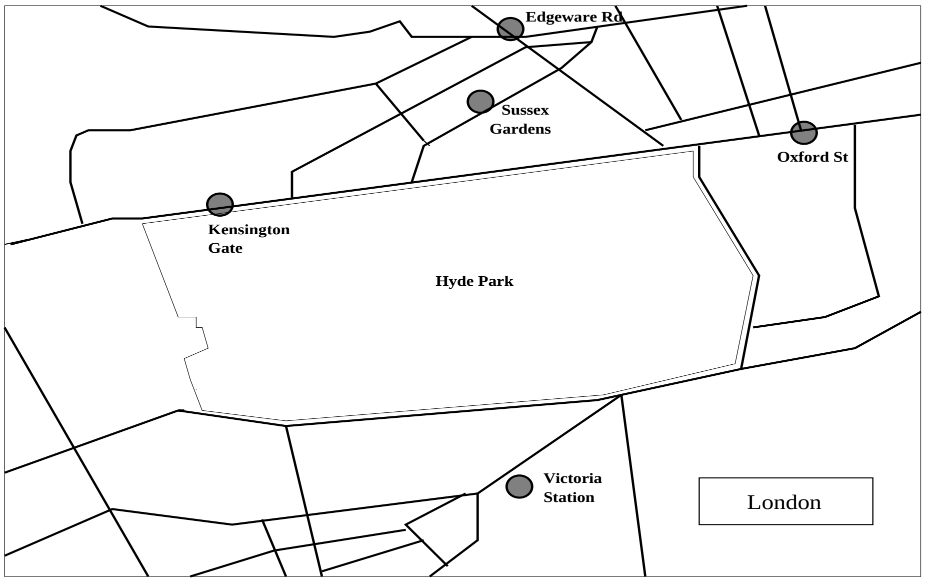

Figure 1: Passanger locations in central London

4.3 Methods of computation

To study the DEK-CFM complexity, we present the computational methods in more detail.

4.3.1 Coalitional values and coalitional configurations

The calculation of each coalitional value requires the evaluation and maximization of the coalitional payoff function, which is a payoff function of several resource variables. Some effort should be put into maximizing the functions by substituting the appropriate quantities of resources – those that will produce the maximum payoff. An example of value calculations can be found in the following taxi-driver case:

Example 4.2 *Recall of the 4 taxi drivers in example 2.1. Suppose that the drivers have received the following tasks: A has to take 7 people from VS to Sussex gardens (SG); B has to take 1 passenger from VS to Edgeware road (ER); C has to take 6 passengers from Kensington gate (KG) to VS; D has to take 1 passenger from Oxford street (OS) to VS. The locations are illustrated in figure 1.*

A typical payoff function of a coalition of taxi-drivers is of the form:

$$U = \sum_{CM}(0.6 + 0.5m_{trans}) \times T(CM) + \sum 0.3m_{parl} - Ins(CS)$$

where CM are the members of the coalition, m_{trans} is the number of miles per transportation (including the way back to base in VS), m_{parl} is the number of miles that can be avoided *due to parallel routes of the coalition members*[15], *and* $T(CM) = \lceil psg/4 \rceil$, *is the number of transportations that a single taxi-driver has to perform due to psg, the number of passengers in her task. Ins is the coalition insurance costs, which is a function of its size CS:* $Ins(CS) = 0.75 \times CS + 0.25 \times CS mod 2 + 0.01 \times GT2$ *(GT2 is 1 if* $CS > 2$*, 0 otherwise). Note that the payoff functions of taxi-driver coalitions are linear. As such, their maximization is simple – it requires only the substitution of the resources into the functions. For instance, the value V(AD) of coalition AD is computed as follows. Cooperation within AD enables avoiding six miles of driving. Therefore* $m_{parl} = 6$.

$$V(AD) = (2(0.6 + 0.5 \times 8) + (0.6 + 0.5 \times 6)) + (0.3 \times 6) - 1.5 = 13.1$$

The design of coalitional configurations is much simpler than the calculation of coalitional values. The only necessary computation is the creation of a set of permutations of agents while avoiding repetitions. Nevertheless, the number of coalitional configurations is much greater than the number of the coalitional values.

4.3.2 K-stable and Pareto-optimal PCs

The complexity of computing K-stable payoff vectors is of high exponential order. To present an algorithm that performs these computations, we first introduce new concepts. Via these concepts we intend to slightly extend the scope of the kernel. This small modification will vastly reduce the computational efforts that are necessary for reaching a PC in the kernel, as discussed in section 4.4. Our computational method relies on the transfer scheme of Stearns [40], and in order to present this scheme we present the definition of the demand function (as can be found in Stearns' work):

Definition 4.2 Demand function
Given a PC, the demand function d_{ij} *of agent* A_i *over agent* A_j *is defined as follows:*

$$d_{ij} = \left\{ \begin{array}{ll} min[(S_{ij} - S_{ji})/2, X_j] & \textit{if } S_{ij} > S_{ji} \\ 0 & \textit{otherwise} \end{array} \right\}$$

where X_j *is the payoff to agent* A_j *and agents* A_i, A_j *are in the same coalition in the PC.*

This demand function will later be used to modify the payoffs of the agents in order to reach K-stable PCs.

While the above definition is similar to the one by Stearns, the following definitions are new concepts defined regarding our own model. The measure of the difference between a payoff vector and an element of the kernel is defined below:

[15] Note that the routing problem itself is NP-complete, however there are polynomial approximation algorithms for its solution. Nevertheless, the routing problem is beyond the scope of this paper.

Definition 4.3 PC-Error
Given a coalitional configuration $\mathcal{C}$ and a payoff vector U, let S_{ij} be the surplus of agent A_i with respect to agent A_j in the PC. The error of the payoff vector U with respect to the kernel is the largest difference between mutual surpluses. Formally, $e = max_{i,j}(S_{ij} - S_{ji})$, where e is the PC-error, and agents A_i,A_j are in the same coalition in the PC.

In practice, we are not interested in the size of the error, since this size strongly relies on the sizes of payoffs in a particular payoff vector. We are interested in a measure that will enable comparison between errors with respect to different payoff vectors and with different payoffs. Such a measure is provided by the relative error:

Definition 4.4 PC Relative error
Given a PC and the PC-error with respect to the kernel, we define the PC relative error e_r to be the ratio between the PC-error e and the sum of all of the payoffs to all of the agents within the given PC.

With the PC relative error, we can examine the proximity of payoff vectors to the kernel. However, when we use an algorithm that leads to PCs in the kernel, we attempt to determine a small constant which will confine the process of PC calculation. For this purpose, we define the PC boundary error ε as a small positive constant which the designers of the agents should agree upon in advance. A K-stable PC which is computed subject to the ε precision will be denoted K-ε-stable. Given the definitions above, we can proceed to the presentation of the algorithm which we use for computing PCs in the kernel. In this algorithm we modify the transfer scheme of Stearns [40] and employ the modified scheme to reach a K-ε-stable payoff vector. The transfer scheme of Stearns guarantees convergence to a K-stable payoff vector but may require an infinite number of iterations in order to converge. We would like our algorithm to stop within a finite time. The modification we suggest does guarantee convergence to a K-ε-stable PC, within a *finite* number of iterations. The implementation of the modified transfer scheme allows the agents to compute a payoff vector for a given coalitional configuration, instead of performing real negotiation to reach such a PC. The algorithm is given below:

Algorithm 1 Truncated transfer scheme
The modified transfer scheme begins with a payoff vector U_0[16]. An iteration of the transfer scheme consists of the following steps:

1. *Start with a payoff vector U_i.*
2. *In case that U_i is non-realizable or inefficient, i.e., the sum of the payoffs of the agents is greater (or smaller, respectively) than the sum of the coalitional values, a correction*

[16]The initial payoff vector U_0 is determined either according to the strategy of the agent or by the protocol.

of the payoff vector is performed. We use the n-correction of Wu [43][17]*, which entails the subtraction (or addition) of an nth of the difference between the sum of the payoffs and the sum of the coalitional values, from (or to) each payoff.*

3. *Calculate all*[18] *of the mutual surpluses and the demand functions with respect to U_i.*
4. *Find among all of the demand functions the greatest one d_{ij}.*
5. *Pass part α, $0 < \alpha \leq d_{ij}$ of the payoff of at least one agent to another agent*[19]*.*
6. *As a result of these steps, the payoff vector U_i is modified and a new payoff vector U_{i+1} is formed.*
7. *If $e_r \leq \varepsilon$ then stop and return U_{i+1} as the result of the algorithm.*
8. *If the iterative process was not stopped in the previous step, the algorithm should proceed to the next iteration (in which the surpluses are re-calculated with respect to U_{i+1}).*

A finite sequence of iterations of the truncated transfer scheme causes a reduction in the magnitudes of the differences between the surpluses and thus leads to convergence into the kernel as implies the proof in [40]. That is, the above iterative approach leads from an initial payoff vector U_0 via k iterations (k does not depend on N) to a payoff vector U_k in which its elements do not differ from an element of the kernel by more than the small pre-defined ε (boundary error)[20]. Details about the relationship between the complexity of the algorithm and ε are in section 4.4 and particularly in lemma 3. We must emphasize that this transfer scheme is performed for each coalitional configuration that is designed on step 2 of protocol 4.1. Now that the computational processes have been described, we present the corresponding complexities.

4.4 Complexity of the DEK-CFM

The Pareto-optimal DEK-CFM is of exponential computational complexity as described below:

[17]While Wu uses this correction as a means for iterating in a transfer scheme for the core, we use it as a single correction in the beginning of our iterative algorithm. This n-correction is not necessary in Stearns scheme because it deals with efficient payoff vectors.

[18]Later in this paper we may confine the set of surpluses and demand functions to be calculated.

[19]According to the transfer scheme of Stearns, α in the range $0 < \alpha \leq d_{ij}$ leads to convergence, and $\alpha = d_{ij}$ will cause the fastest convergence (details of this transfer can be found in [40]).

[20]In a case where the pre-defined PC boundary error is the precision of the most precise agent in the environment, the resulting PC is in the best possible approximation of the kernel.

Computation of coalitional values and configurations

The DEK-CFM requires the computation of 2^n coalitional values and the design of $O(n^n)$ coalitional configurations (as was discussed in lemma 1) and in steps 1 and 3 of protocol 4.1. Therefore, the computational complexity of this stage is $O(n^n)$. These calculations are distributed among the agents, however the complexity remains $O(n^n)$.

Computation of K-ε-stable and Pareto-optimal PCs

The DEK-CFM will employ the transfer scheme algorithm for calculating K-ε-stable PCs. Here we discuss its complexity. Each iteration of the transfer scheme algorithm is constructed from the following steps:

- All of the 2^n excesses are calculated in each step. The calculation of each excess is of order of n (see definition 3.1).
- Among the excesses, the surpluses are searched for. Therefore, the excess and surplus calculations are together of order $O(n2^n)$.
- Among the surpluses, the greatest is found by $(O(2^n))$ operations.
- Additional calculations of a lower complexity are required to complete one iteration of the transfer scheme as well.

To summarize, the total complexity of one iteration of the transfer scheme is $O(n2^n)$. Many iterations should be performed to reach convergence of the transfer scheme. A reasonable assumption is that at least n (equal to the number of the agents) iterations should be performed, in order that each agent's payoff will be partially transferred at least once during the process. Obviously, the number of iterations until convergence will be affected mainly by the boundary error ε. The exact relationship is given by the following lemma:

Lemma 3

The resulting payoff vector of algorithm 1 will converge to an element of the kernel (with a relative error not greater then ε) within $n \log_2(e_{r_0}/\varepsilon)$ iterations, where e_{r_0} is the initial PC relative error.

Proof: At each iteration of the transfer scheme, half of the greatest difference between surpluses is passed from the payoff of one agent to the payoff of another agent. As a result, the largest difference between surpluses on the next iteration will be half of the original one. Thus, a sequence of k iterations will lead to a magnitude of the greatest difference between surpluses which is, by 2^k, smaller than the initial one. Therefore, the demand that $e_{r_0}/2^k = \varepsilon$ implies that $k = \log_2(e_{r_0}/\varepsilon)$. Since all n agents may be approached before the agent that was approached first is approached once again, we multiply the result by n. □

Since algorithm 1 will be performed for all of the elements of the CCS, it will be calculated $O(n^n)$ times. Therefore, we conclude that the complexity of the Pareto-optimal DEK-CFM is $O(n2^n n^n)$ computations. Due to the distribution, this order of complexity of computations is divided by n, but there is an additional communication complexity, which is of order[21] $O(n^2)$.

4.5 Discussion of the DEK-CFM

The DEK-CFM leads to the formation of coalitions such that the PCs are Pareto-optimal in the kernel. Such results should be preferred by designers whose agents have unconstrained time or very strong computational capabilities, since Pareto-optimal K-ε-stable PCs are of good quality for the agents. Although it may seem that we present a game theoretic approach, it is not so. As opposed to game theoretic research (see section 7), our distributed model provides the agents with a detailed step-by-step method for coalition formation. According to our model, the agents begin as *single agents* and gradually form coalitions. This is different from the common game theory approach, where the agents begin when they are already in a given coalition configuration, and do not pass from one coalition configuration to another, but just calculate the appropriate payoff vector. We provide a distributed approach that enables autonomous self-interested agents to do the best for their own benefits.

The DEK-CFM is of high computational complexity, thus may prevent beneficial coalition formation. Therefore we provide the agents with a CFM which radically cuts down the number of PCs from an exponential number to a polynomial number[22], and provides polynomial-K-ε-stable PCs. However Pareto-optimality, which must be considered with respect to *all* of the coalitional configurations (an exponential number) cannot be guaranteed by the CFM hereinafter.

5 The negotiation-oriented CFM

We present below a reduced cost CFM which is based on negotiation [6, 21] among the agents. This is a Distributed, Negotiation-based, Polynomial, Kernel-oriented CFM (DNPK-CFM). The DNPK-CFM consists of steps in which coalitions transmit, accept and reject proposals for creating new coalitions, i.e., proposals for sharing resources and tasks and distributing the common payoff. The DNPK-CFM starts with all agents in single-membered coalitions, and proceeds through a sequence of DNPK-CFM steps as described in section 5.2. In each

[21] Compared to a case of centralized calculations, where the communication complexity is $O(n)$.

[22] At least in its first stage.

step, at least one coalition will attempt to improve the payoffs of its members by making a coalition formation proposal to another coalition. The acceptance of such a proposal will improve the situation of the agents involved. The DNPK-CFM may continue until all of the proposals of all of the agents are rejected, thus reaching a steady state, which is defined below, or until the time-period that was allocated for negotiation ends.

Definition 5.1 Steady state
A steady state of the agents' environment is a state where the agents are arranged in a PC such that one of the following conditions applies:

1. *The agents have reached a K-ε-stable and Pareto-optimal*[23] *PC, or;*
2. *The agents have not reached a PC as in 1, but have no more possible beneficial proposals (allowed by the protocols) to be transmitted to others.*

When the agents use the DNPK-CFM, the coalitional configurations that are formed when the agents reach the steady state are stable according to a new stability concept, the polynomial-K-ε-stability (see definition 5.3 below). However, since the DNPK-CFM is an anytime algorithm [9], even if it is terminated after a limited number of steps *before* reaching a steady state, it still provides the agents with a polynomial-K-ε-stable PC.

5.1 The polynomial approach

Reduction of the complexity of calculating K-ε-stable PCs from exponential to polynomial will be performed via polynomial excesses, polynomial surpluses and polynomial-K-stability. These new concepts are introduced below. We define polynomial excesses as excesses that are calculated using a polynomial set of coalitions. The designers of agents must agree upon protocols that will direct their agents to a well defined polynomial set of coalitions. In order to reach such a subset, we suggest that the designers agree upon two integral constants K_1, K_2, $K_1 \leq K_2$, and allow only coalitions of sizes in the ranges $[K_1, K_2]$ to be considered for excess calculations. Although other values are allowed, small K_1, K_2 should be preferred, since the sizes of K_1, K_2 directly influence the order of the polynomial complexity (details in section 5.5). We leave the designers of the agents freedom in choosing the *predefined* constants K_1, K_2. Such freedom enables the designers, when developing a specific setting for their agents to work in, to fit the algorithm to the specific environment. Agents should not try to choose Ks contradictory to agreed upon K_1, K_2 because according to the protocols, objections based on different Ks are not acceptable. Since the agents will compute only polynomial excesses in order to outweigh one another, we introduce the polynomial surpluses:

[23] The recognition of Pareto-optimality is possible only when all possible configurations have been considered.

Definition 5.2 Polynomial Surplus
A polynomial maximum surplus SP is a maximum surplus that is computed from a polynomial set of excesses.

The agents may use the polynomial surpluses in order to outweigh one another or to examine PCs in order to find polynomial-K-ε-stability:

Definition 5.3 Kernel Polynomial-K-ε-Stability
A coalition $C \in \mathcal{C}$ is polynomially-K-ε-stable if $\forall A_i, A_j \in C, i \neq j$, either $u^i = 0$ or $u^j = 0$ or $|SP_{ij} - SP_{ji}| \leq \varepsilon$.

Given a specific coalitional configuration with an arbitrary payoff vector, it is possible to compute a polynomial-K-ε-stable PC by using a polynomial truncated modification of the convergent transfer scheme. In algorithm 1, step 3 must be modified; this modification entails calculation of mutual surpluses only with respect to the K_1 and K_2 bounding constants. We use this method in the DNPK-CFM protocol described in the following sections.

5.2 The DNPK-CFM protocol

The following protocol advances the agents from one coalitional configuration to another, thus increasing their payoffs. Given a specific coalitional configuration, the agents try to find a correspondent *stable* payoff vector. A convergence algorithm is used for this purpose. The protocol of the DNPK-CFM is provided below. Denote a proposal of coalition C_x to coalition C_y by P_{xy}.

Protocol 5.1 Negotiation protocol

1. **Preliminary stage:** *prior to negotiation, the agents must calculate the values of coalitions in the range of sizes K_1 to K_2, using the calculation methods of the DEK-CFM.*

2. **First stage:** *a coalition member may receive proposals only as a member of the coalition*[24].

3. *Each coalition will coordinate its actions either via a representative or by voting (or both). For details see section 5.4.1.*

4. *Each coalition C_p iteratively performs the following:*

[24] Thus, coalitions can only expand; if sub-coalitions are not allowed to receive proposals, no offers for destruction are possible.

- *Transmit a proposal to a target coalition; wait for a response. Details about waiting-period* t_{wait} *are in section 5.4.3. A proposal* P_{pr} *of* C_p *to* C_r *is the details of the joint coalition* C_{p+r} *and the coalitional configuration* PC_{new}*: (i) it consists of resource and task redistribution and payoff distribution and must be polynomially-K-*ε*-stable, calculated with respect to* K_1, K_2*; (ii) the new payoff vectors* U_{p+r} *of* C_{p+r}*, must increase the payoffs to* C_{p+r}*-members; (iii)* PC_{new} *must include* C_{p+r}*. Other coalitions remain unchanged in* PC_{new}*.*

- *Accept* P_{rp} *(a proposal of* C_r *to* C_p*) only if* $P_{pr} = P_{rp}$ *(*C_r *follows the same protocol). Send an acceptance message to* C_r*. When multiple pairs of coalitions have a mutually equal pair of proposals, the pair of proposals (from among these) which was proposed first is accepted.*

- *If* P_{pr} *was accepted and mutually confirmed by* C_p *and* C_r*, form* C_{p+r} *according to* P_{pr} *details*[25]*. If necessary,* C_{p+r} *should choose a coalitional representative.*

- *Notify coalitions* $C_a, a \neq r$ *of the acceptance of* P_{rp}*. Send rejection messages to all proposers of* $P_{ap}, a \neq r$[26]*.*

5. *The above sequence should be repeated until a steady state is reached (see definition 5.1), or when the time-period, agreed upon by the designers in advance, ends.*

6. *To enable perception of a steady state, each coalition must announce its status (i.e., that it has no more proposals to transmit). This announcement is valid only for the current iteration*[27]*. If the agents run out of computation time before a steady state has been reached, the algorithm terminates and the last PC holds.*

7. **Second stage (optional):** *the coalitions will follow the same sequence of steps as in the first stage of the DNPK-CFM. However, proposals that involve destruction, i.e., proposals which are addressed to single agents, although they are coalition-members, are allowed.*

8. *When a new PC changes the payoffs of the agents, it may lead some agents to leave their previous coalitions. These coalitions will destruct.*

[25]The payoff vector U_{new} of PC_{new} is valid from this point on and is used as the basis for future negotiation.

[26]Note that even if agents deviate from reporting a rejection, the limited waiting time for acceptance will cause a waiting agent to assume a rejection. Nevertheless, since early notification of rejection may be beneficial for the waiting agent, it may be willing to pay to ensure such a notification, thus motivate the sending agent not deviate from its role.

[27]Given the current PC, a coalition may have no proposals to make and it should report this. However, a new PC may cause this coalition to have possible proposals, thus it is no longer in a steady state.

9. *The second stage will end either when a steady state*[28] *is reached or when the computation time ends. If PCs for all of the possible coalitional configurations have been calculated during the second stage, no additional iterations are allowed*[29].

10. *To enable perception of a steady state, each coalition must announce its status when it has no more proposals to transmit. This announcement in the second stage does not depend on such an announcement from the first stage.*

The protocol above is enforceable since deviation from it is revealable, as demonstrated below: in the first stage, proposals addressed to coalition-members will be immediately detected if the agents accept these proposals and deviate from their current coalition. Similarly, a coalition member that does not follow coalitional decisions will be detected. Agents may not wait for the response to their offers, however if they act according to another proposal (join another coalition) their deviation is immediately detected. The latter also prohibits sending several proposals simultaneously. Proposals that allow only increments in the payoff of the agents are rational, but can also easily be checked by any agent as to their being in the kernel, and therefore deviation is exposable.

The limitation that destruction of coalitions be avoided in the first stage will radically shorten the coalition formation process by avoiding most of the intra-coalitional computation and communication in this stage. The number of iterations until a steady state is reached in the first stage of the DNPK-CFM is $O(n^2)$. The optional stage 2 may be used if, after stage 1 was completed, the agents have additional time and computational resources. They may proceed until they reach a Pareto-optimal polynomial-K-ε-stable PC or until the time ends. To find a Pareto-optimal PC, all of the PCs in the PCS must be approached, therefore the complexity will be $O(n^n)$ (if time constraints allow). Through stage 2, the average personal payoff of the agents may sometimes decrease. However, a Pareto-optimal PC can be reached, and the average of the expected final personal payoff increases. Nevertheless, we consider the first stage of the DNPK-CFM sufficient for reasonable coalition formation, as shown by results of simulations we have performed (see section 6).

A proposal will be designed with respect to the relevant information, i.e., the information about resources, tasks and payoff functions of other agents. Our protocol requires that, at the beginning of the coalition formation, the agents transmit this information. False transmission of such information is revealable: when an agent reports more resources than it actually has, it must later use them for task execution and its deceit is revealed; agents that report fewer resources than they actually have will be considered less valuable for cooperation thus decrease their benefits via this false report, and therefore avoid it. Dynamically changing

[28]Note that the steady state in the second stage is different from the steady state in the first stage since there are different restrictions on the proposals in these two stages.

[29]Otherwise, an infinite loop may emerge.

information with regards to coalitions that have formed should be transmitted only to the agents involved. Since several agents are involved, deceitful transmissions by one agent is revealable by the others. These properties makes the information transmission requirement enforceable.

As was stated above, an acceptance of a proposal implies an acceptance of the corresponding payoff vector by all of the agents. This may change the payoffs of agents who are not directly involved in the negotiation. These may react in the proceeding iteration with new proposals according to which they receive a greater payoff. Commonly, even in non-super-additive environments, larger coalitions gain greater outcomes, and therefore the a-priori expected payoff of the agents is larger. Hence, protocols that cause enlargement of coalitions (as we present) should be preferred a-priori by the designers of the agents. Such an increase in the average expected outcome can be observed from the simulation results (section 6).

5.3 A DNPK-CFM example

In example 4.2, taxi-drivers have received transportation tasks. Cooperation among the taxi-drivers on task execution can increase their common payoff. However, cooperation requires that some drivers drive more than they would have had to drive according to their original tasks. These will agree to drive more only if they are paid more than their additional costs. The drivers may use the DNPK-CFM in order to create coalitions.

We illustrate the implementation of the DNPK-CFM, by a single step. The DNPK-CFM scheme requires that each driver first decides with which other drivers it is interested in forming a joint coalition. A driver should rank the other drivers (as we suggest in strategy 5.1) according to the expected payoffs from forming pairwise coalitions with each. Such an estimate may result in the following ordered lists, where the first on each list is the most preferable one: A: D,B,C. B: C,D,A. C: D,B,A. D: A,B,C. Estimation is made with respect to tasks that can be divided among the members of a future coalition or passed between them, and the additional expected payoff from such redistribution. We concentrate on the list of A. A can observe that D's and B's targets, which are ER and OS, are very close to SG. They each have one passenger to pick up, so they can take three more, if necessary, and A has more passengers than she can take in one trip. Therefore, cooperation with B or D may be mutually beneficial, hence A prefers them over C. To choose between B and D, A has to calculate the coalitional values.

Next, each driver has to design a proposal and transmit it to the target driver. Here, A should design a proposal to be transmitted to D. A has to find a polynomially- K-ε-stable payoff vector for the proposed coalitional configuration (AD, B, C). Given the distances d(VS,SG)=4, d(VS,OS)=3, d(VS,ER)=5, d(VS,KG)=4, d(SG,OS)=1, d(SG,ER)=1, d(SG,KG)=1, d(KG,OS)=2, d(KG,ER)=2, d(OS,ER)= 2, the coalitional values are $V(A) = 8.2, V(B) = 4.6, V(C) = 8.2, V(D) = 2.6, V(AD) = 13.1$. $V(AD) = 13.1$ because by joining together A and D can drive 6 miles less (saving 1.8 pounds) and save 50 pence on insurance. According to the coalitional values, the payoff vector that A designs should be $(8.2 + x, 4.6, 8.2, 2.6 + y)$, where $x + y = 2.3$. To find such a polynomially-K-ε-stable payoff vector, a transfer scheme should be used. Setting $K_1 = K_2 = 2$, only coalitions of size 2 are considered during the transfer scheme (and their coalitional values should be calculated). Under such conditions the polynomial kernel is not unique, and the transfer scheme will lead within a few iterations to a polynomially-K-ε-stable payoff vector, e.g., $U = (9.35, 4.6, 8.2, 3.75)$. A will transmit this proposal to D as part of P_{AD}.

5.4 Strategies and protocol details of the DNPK-CFM

Protocol 5.1 requires that a coalition C_p decide which other coalitions to approach. We suggest that C_p will use the following strategies. Strategy 5.1 (below) allows a coalition to compute (given their bounds on computation), in some cases, a pure Nash equilibrium strategy, and in others a mixed strategy Nash equilibrium. For the latter, probabilities of proposal transmission are necessary. To allow for compact representation of the strategy, we define the following, referring to any two coalitions C_i, C_j:

1. $E^j_{P_i}$ is the expected outcome of coalition C_j from a proposal P_{ij} by C_i, given the bounded computation time.
2. q_{ij} is the probability of C_i sending a proposal to C_j.
3. T_{pc} is the time already spent for proposal computation.
4. Given a coalition configuration $\mathcal{C}$ and $C_i, C_j \in \mathcal{C}$, we define Q^k_{ij}, the probability that there would be k other pairs of coalitions $C_x, C_y \in \mathcal{C}$, $x \neq y \neq i \neq j$ such that for each of these k pairs, C_x and C_y mutually send the same offer to one another.

Note that Q^k_{ij} can be expressed by means of combinatorics (using sums of products of permutations of q_{ij}s), however, the details may be rather exhaustive and of minor interest. Denote the number of coalitions in the configuration by nc, k may never exceed $\lfloor nc/2 \rfloor$.

Strategy 5.1 Ranking and selecting proposals
Aim: rank and select proposals.
Input: all coalitional values within the K-bound; time threshold T.
Output: a proposal P_{px} to some coalition C_x.

Specification: coalition C_p should perform the following:

1. *If T does not allow for multiple proposal calculations:*

   (a) *Rank each coalition $C_r \neq C_p$ according to the payoffs that C_p expects that its members will receive from forming a joint coalition C_{r+p}. Select one of the following ranking methods according to the available computational resources:*

      i. *explicitly calculate the value of the joint coalitions and the polynomially-K-ε-stable corresponding payoff vector (such calculations are almost as complex as designing complete proposals for all other coalitions).*

      ii. *estimate the expected payoffs. Such an estimate can be done by checking the quantities of resources that the other coalitions have and examining the previous payoffs that the members of a coalition have received.*

   (b) *Compute a proposal for the most highly ranked coalition and assign to P_{px} (use strategy 5.2).*

   Else (T allows for multiple proposal calculations):

2. *$\mathcal{P}_{pr} = \emptyset$, where $\mathcal{P}_{pr}$ is the set of proposals of C_p to C_rs, $r \neq p$.*

3. *While $T_{pc} < T$ perform:*

   (a) *For each C_r, compute a proposal P_{pr} (use strategy 5.2) and $E^p_{P_{pr}}$ w.r.t T_{pc}. If $E^p_{P_{pr}} > 0$ put P_{pr} in $\mathcal{P}_{pr}$.*

   (b) *Select the best proposal (for C_p) $P_{pr'} \in \mathcal{P}_{pr}$. Compute proposals $P_{ar'} \in \mathcal{P}_{ar'}$ that $C_{r'}$ should receive from other coalitions[30] $C_a \in \mathcal{C}_{\dashv}, a \neq r'$. For each $P_{ar'}$, compute $E^{r'}_{P_{ar'}}$ w.r.t T_{pc}.*

   (c) *If $P_{pr'}$ is better than the proposals in $\mathcal{P}_{ar'}$, i.e., $\forall P_{ar'} \in \mathcal{P}_{ar'}$, $E^{r'}_{P_{ar'}} < E^{r'}_{P_{pr'}}$, designate $P_{pr'}$ as "superior", else designate $P_{pr'}$ as "inferior".*

   (d) *If there is a "superior" proposal in $\mathcal{P}_{pr}$, assign it to P_{px} and break; else continue:*

   (e) *If there are proposals in $\mathcal{P}_{pr}$,*

      i. *for each coalition C_i, decide which are the coalitions $C_j, j \neq i$, to which it may send proposals[31]. Compute proposals P_{ij} addressed to the selected coalitions and assign to $\mathcal{P}_i$.*

      ii. *for each C_i, for each $P_{ij} \in \mathcal{P}_i$, compute $E^j_{P_{ij}}$.*

[30] We denote the set of all other coalitions by $\mathcal{C}_{\dashv}$. Note that if coalition $C_a \in \mathcal{C}_{\dashv}$ follows this strategy, $C_{r'}$ should receive $\mathcal{P}_{ar'}$, unless network (or other) flaws have occurred.

[31] A coalition may decide not to send proposals to some other coalitions since these proposals are non-beneficial.

iii. calculate probabilities: while T allows and no solution yet found pick, beginning with the smallest[32]*, $\mathcal{P}'_i \subseteq \mathcal{P}_i$ and assign probability q_{ij} to each proposal $P_{ij} \in \mathcal{P}'_i$ by solving a set of equations*[33] *according to which, for every pair of proposals $P_{ij}, P_{ik} \in \mathcal{P}'_i$, $i \neq j \neq k$, $q_{ij}(E^i_{P_{ij}} \sum_{l=0}^{\lfloor nc/2 \rfloor -1} \frac{1}{l} Q^l_{ij}) = q_{ik}(E^i_{P_{ik}} \sum_{l=0}^{\lfloor nc/2 \rfloor -1} \frac{1}{l} Q^l_{ik})$, and $\sum_l q_{il} = 1$.*

iv. associate a probability q_{pr} to each proposal in $\mathcal{P}'_{pr}$.

v. randomly choose a proposal from $\mathcal{P}'_{pr}$ w.r.t their attached probabilities and assign to P_{px}.

4. If not empty, P_{px} is the selected proposal of C_p. Else, there are no beneficial proposals for C_p to offer – announce a steady state.

Note that the number of equations for probability computation (in item $3.e.iii$ of the strategy above) may be greater than the number of probabilities. Strictly mathematically speaking, this implies that there may be $\mathcal{P}_i$s where there in no solution. Nevertheless, as known from game theory, any finite n-person game has at least one equilibrium in mixed strategy. Since our game is such, there should be a solution (at least one) to the equations. The details of the proposal calculations are in strategy 5.2 in section 5.4.2. The computation of E_Ps (in $3.e.ii$) depends on T. The simplest evaluation of an E_P is by relying on payoffs that stem from the Kernel computation. Further computation, restricted by the time quota T, may be performed by additional search in the (kernel) solution space. To show that the suggested strategy satisfies the time-bounded equilibrium requirement we first must show that an agent will not deviate from it when computing E_P. This results from the introduction of the time threshold T. A coalition C_p would believe, relying on the information provided by the agents' designers, that computational efforts beyond T are non-beneficial. In addition, excess computation by C_p may result in missed deadlines, which will prohibit coalition formation and its benefits. Therefore, a deviation from the strategy that entails additional computation implies, according to this belief, a reduction in the expected utility, especially when other coalitions follow the same strategy. Hence, C_p will avoid such a deviation. Another possible deviation of C_p is the avoidance of some of the computations. This, however, is irrational, since additional computation within the time threshold may increase the benefits (with no risk of a reduction). Therefore, this type of deviation should be avoided as well.

Given its belief about its expected utility (calculated w.r.t the time bound), if other agents do not deviate from the strategy above, an agent should follow it (that is, there is an equilibrium). To show this, it is necessary to explain why should agents follow the

[32]A smaller $\mathcal{P}'_i$ imposes a smaller set of equations and therefore is simpler to solve.

[33]Note that the solution of the equations provides C_p with more probabilities than it actually needs for carrying out its strategy. Nevertheless in order to compute the necessary probabilities it must compute the whole set of equations.

proposal selection methods provided in the strategy. In the case of a superior proposal, this is rather simple. Since the best superior proposal P_s is better for the proposee C_r than its other proposals, and is the best proposal of C_p that will be accepted by C_r, according to the equilibrium strategy of C_r, it would accept it. The choice of a proposal other than P_s by C_p will decrease its expected benefit (thus is an irrational behavior and be avoided). When there are no superior proposals, an inferior one is selected. In case of inferior proposals, the probability computation in the strategy above results in equal expected utilities $\forall P_{pr} \in \mathcal{P}_{pr}$, so C_p is indifferent toward different proposals, given that the other agents follow the same (mixed) strategy. Therefore, coalitions that follow the proposed strategy will have a *mixed strategy Nash equilibrium* given the time constraints.

5.4.1 Coalitional decision making

As was stated in protocol 5.1, there must either be an agent who is responsible for receiving, designing and sending proposals and for any other coalitional actions, or the agents must use a voting method in order to make coalitional decisions. Between these two, the designers should select the one that better fits their agents' computational capabilities. Voting is better for computationally strong agents, where each agent has to compute its own preferences and then vote according to the results. This requires extensive computations. When designers attempt to minimize their agents' computational and communication efforts, they should prefer that their agents elect a representative who will perform the coalitional computations. A reduction in the variance of the distribution of calculations among the members of a coalition can be achieved by replacing the representative every iteration. Thus, the calculations will be distributed more homogeneously. In addition, advantages that may arise from the representative role will be distributed as well. We emphasize that the designers must agree upon one of the methods in advance, to prevent a situation where the agents act according to the two different approaches, thus reaching a conflict situation and avoiding cooperation.

When voting is selected for decision making, each decision (e.g., design a proposal) must be computed by all coalition members, and they should select the preferred one from among the results[34]. This will be done by a raffle or by voting. These decision-making methods are also used for choosing the representative when the designers adopt the representative approach.

Self-interested agents will not accept the representative job unless they are compensated for their additional efforts, or forced by the protocol to accept this role. If the designers adopt the representative method, their agents may choose the coalitional representative by a raffle or via a voting method (for examples see [10, 26]). Note that voting is a suggested algorithm,

[34]These results may differ due to the tendency of rational agents to compute (and present) a proposal according to which they are better-off.

and not forced, though should save time and effort on the part of the agents (which may be required, e.g., for negotiation about the representative) and reach a reasonable decision. An agent who deviates from the voting algorithm is considered absent from the vote.

5.4.2 Proposal structure and design

According to part 3 of protocol 5.1, proposals for the generation of new coalitions should be designed by the current coalitions. We denote a coalition that designs and transmits a proposal by C_p and a coalition that receives a proposal by C_r. All other coalitions are denoted by C_a. We suggest that C_p use the following strategy to design a proposal for C_r:

Strategy 5.2 Proposal design
Aim: design a proposal for a joint coalition C_{p+r} with respect to the proposal protocols.
Input: current PC and coalitional resources, tasks and payoff functions.
Output: a proposal - a new PC (PC_{new}).

Specification: coalition C_p should perform the following.

- *Calculate the coalitional value V_{p+r} of C_{p+r}.*
- *If $V_p + V_r \geq V_{p+r}$ quit the design of this specific proposal for C_r.*
- *Otherwise, $\forall C_a$ such that $|C_a|$ in $[K_1, K_2]$, retrieve V_a (if stored in memory in previous calls to this strategy), else calculate V_a and store for future use.*
- *Calculate U_{new}, the payoff vector of PC_{new} in which C_p and C_r form C_{p+r}, and all C_a remain unmodified, using algorithm 1 (section 4.3) with the current U (the payoff vector of the current PC) as its input with polynomial excesses, subject to K_1 and K_2.*
- *Compare U_{new} to the current U. If $\forall A_i \in C_{p+r}, u^i_{new} \geq u^i$ and $\exists A_i \in C_{p+r}, u^i_{new} > u^i$, then PC_{new} is the proposal to C_r. Otherwise, stop designing this specific proposal.*

Note that the calculation of the coalitional value may be complex, but if we restrict the payoff functions to low degree polynomial functions, the complexity of these will be polynomial. Linear functions will guarantee linear complexity.

Example 5.1 *Recall the taxi-driver example (sections 4.2, 5.3), in which A designed a payoff vector $(8.2+x, 4.6, 8.2, 2.6+y)$, where $x+y = 2.3$ to be transmitted to D as part of a proposal. A full proposal should include the coalitional configuration $\{AD, B, C\}$. Resource and task redistribution are also necessary. This redistribution should be done after reducing the driving of parallel routes. On her way from VS to OS to take a passenger, D can continue to SG and pick up 3 of A's passengers, then return to VS. Thus, D adds 2 units of distance to her* *tasks, as OS and SG are 1 unit apart, and D will drive there and back. However, this change enables A to complete her original task with one trip only. At first A had 7 passengers to take, and it required driving twice. Now, with D's support, only 4 passengers are left and this requires one trip. The avoidance of one trip to SG and back saves 8 units of distance. The net reduction of distance is therefore 6 units. This reduction affects A and D's total benefits, as expressed in the coalitional value.*

Strategy 5.2 requires that A first check if $V(AD) > V(A) + V(D)$. *Since this is true, proceed and calculate the coalitional values of coalitions of size 2 (because* $K_1 = K_2 = 2$*). Following the method in which* $V(AD)$ *was calculated, the values are:* $V(AB) = 15.7, V(AC) = 16.9, V(AD) = 13.1, V(BC) = 15.1, V(BD) = 9.5, V(CD) = 13.1$. *Using these coalitional values, A should calculate the appropriate polynomially K-*ε*-stable payoff vector using algorithm 1, beginning with the current payoff vector,* $(8.2, 4.6, 8.2, 2.6)$. *The resultant payoff vector of the transfer scheme with a PC boundary error* $\varepsilon = 0.05$ *is* $(9.35, 4.6, 8.2, 3.75)$. *This result should be compared to the current PC. Since the calculated PC is better than the current PC both for A and D, A should send it as a proposal to D.*

5.4.3 Waiting time

Given a predefined time period[35] t_{wait}, having sent a proposal P_{pr} to C_r, C_p must wait t_{wait} for response. During the waiting time C_p is committed to its proposal. If C_r's answer was not received within t_{wait}, C_p's obligation to respect P_{pr} expires. The waiting time protocol can simply be enforced in cases where deviation of C_p from it harms C_r, since C_r can report it (and C_p can be penalized). Nevertheless, if C_r is not affected by C_p's deviation, enforcement is not necessary. A coalition C_p who waits more than t_{wait} may lose good opportunities to cooperate with others and benefit from this cooperation. t_{wait} should be agreed upon by the designers of the agents.

5.5 Complexity of the DNPK-CFM

The analysis of the complexity refers to the first stage of the DNPK-CFM (unless designated differently). The complexity of the DNPK-CFM is strongly affected by K_1 and K_2 and the difference between them. According to these constants, the agents compute the polynomial set of coalitions, the coalitional values and the coalitional configurations. The complexity of these calculations is of the same order of the number of the coalitions considered (this relation holds for the exponential case as well). The number of coalitions considered for

[35]The decision upon t_{wait} must be done with respect to the assumptions of the designers with regards to network latencies. t_{wait} should usually be greater than the expected average latency, however preferably not greater than all possible latencies, lest agents might wait an unrealistic amount of time.

polynomial-K-ε-stability calculations in the DNPK-CFM is given by

$$n_{coalitions} = \sum_{i=K_1}^{K_2} \binom{n}{i} = \sum_{i=K_1}^{K_2} \frac{n!}{i!(n-i)!}$$

This expression is polynomial since $\binom{n}{i} \leq n(n-1)(n-2)...(n-i+1)$ which is a polynom of order $O(n^i)$, and the sum over a constant number $(K_2 - K_1)$ of polynoms is a polynom of the largest order of all polynoms in the sum.

The DNPK-CFM requires the computation of $n_{coalitions}$ coalitional values. It also requires the design of coalitional configurations. The number of coalitional configurations computed depends on the time constraints. When time-limited part of strategy 5.1 (first item), $O(n)$ configurations will be computed at each iteration. This number increases to $O(n^3)$ when the other part of the strategy is employed.

For each coalitional configuration, a polynomial-K-ε-stable PC must be calculated. using algorithm 1. Each iteration of algorithm 1 is constructed from the following steps:

- As stated previously, given a payoff vector U (for a specific coalitional configuration) there are $n_{coalitions}$ excesses. All of these excesses are calculated in each step, and the calculation of each excess is of order of n (see definition 3.1).
- All $n_{coalitions}$ excesses are searched for the surpluses. Therefore, the excess and surplus calculations have together an order of $O(n \times n_{coalitions})$.
- Among the surpluses, the greatest is found by $O(n_{coalitions})$ operations.
- According to lemma 3, the resulting payoff vector of algorithm 1 will converge to an element of the polynomial-kernel (with a relative error not greater then ε) within $n \log_2(e_{r_0}/\varepsilon)$ iterations, where e_{r_0} is the initial PC relative error.

Altogether, the complexity of one iteration of algorithm 1 is $O(n^2 \times n_{coalitions})$. Referring to the case where time is less bounded, algorithm 1 will be performed for $O(n^3)$ coalitional configurations per protocol iteration, for $O(n^2)$ iterations. The distribution of computations among the agents allows a reduction in the complexity (per agent) by a factor of n. Therefore, the complexity of the DNPK-CFM is $O(n^6 n_{coalitions})$ computations. Similarly, in the case where time is strictly bounded, the complexity will be $O(n^3 n_{coalitions})$ computations. In addition, there is a communication complexity of order $O(n^2 n_{coalitions})$.

If the DNPK-CFM proceeds through the optional second stage, the number of coalitional configurations increases and depends on the time constraints. The upper limit is of order $O(n^n)$, however the stability calculation of each configuration remains polynomial.

The first stage of the DNPK-CFM is relatively cheap in terms of computations, since only the computations necessary for reaching *polynomially-K-ε-stable* PCs are performed.

Contradictory to the exponential DEK-CFM, the low costs of the DNPK-CFM (first stage[36]) will encourage cooperation and coalition formation.

6 Simulation results

We have developed a simulation tool to examine the performance of the DNPK-CFM. Note that in the simulation we used the restricted form of strategy 5.1, that is, only the ranking phase was performed to reduce the computational complexity. This tool has been run for a variety of agent settings, and the results are summarized below. In particular, we simulated the taxi-drivers' example that was presented above. The first coalition that was formed[37] was AC and the payoff vector was $U_1 = (11.4, 4.6, 9.7, 2.6)$, which is polynomially-K-ε-stable but not K-stable and not Pareto-optimal. Therefore, the negotiation continued and the next coalition was BD, and the payoff vector was $U_2 = (11.4, 4.6, 9.7, 4.6)$. U_2 is K-stable and Pareto-optimal, and agrees with the exponential-calculation results. The Pareto-optimal simulation result was achieved both with limited K_1 and K_2 and without this limitation. These results were obtained for all runs. However, the DNPK-CFM is not deterministic, and its results may depend on the order of offering during the negotiation. For example, when we increased the allowed error ε (this increment reduces the computational complexity) in some of the runs, another PC was reached: (A, BC, D), which is polynomially-K-ε-stable, but not Pareto-optimal.

Next, we tested the performance of the DNPK-CFM with respect to different constants (K's,ε) and different environmental settings. We have performed several hundreds of runs of our simulation and the results we report are averages of these. For algorithmic and computational-complexity reasons, the simulation can process only polynomial payoff functions with the assumption of independent resources. We focused on linear payoff functions. In our experiments, the computational complexity led us to choose the constants K_1 and K_2 to be small, usually in the range of 5 to 10. We have also allowed the simulation to run with no K limitation, to reach an exponential solution when the number of agents is small[38] (up to 10 agents). To compute the coalitional values, we have drawn resource values w.r.t three distributions: random, uniform and normal. We also checked cases in which some or all of the values were artificially shifted to above or below average, or both. Since we used linear payoff functions, the distributions of the coalitional values were similar to those of the resources. The values of coalitions in the simulation were in the range 0 to 1000. We found out that except for the very extremely uneven distributions (i.e., very large variance),

[36]We view this stage as sufficient for reaching reasonably beneficial and stable coalitions.

[37]The order of the suggestions is based on the ranking heuristics, according to which A prefers C over the others, and vice versa. Hence, AC is a preferred suggestion.

[38]These means that 2^{10} coalitions were considered for excess computation. Although exponential, this is not intractable.

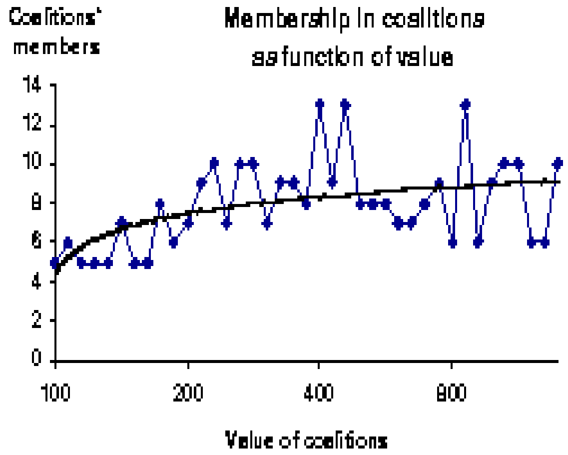

Figure 2: The number of agents participating in coalitions (Y-axis) increases as the average coalitional value (X-axis) increases.

no significant differences were detected in the evolution and performance of the simulated coalition formation.

Running the simulation has provided the following results, which are each an average over dozens of runs. Initially, we have shown that the simulated DNPK-CFM reaches a stable PC within a reasonable time (for the first stage of the DNPK-CFM). In addition, it has been found (for the settings that we have examined) that the DNPK-CFM continuously improves the agents' payoffs, whether it is normally terminated or halted artificially. The main results of the simulation for 5 through 16 agents, with $K_1 = 1$ and $K_2 \leq 10$ and with $\varepsilon = 1$ (which was less than 1% of the average coalitional value) are as follows:

1. The number of agents that participate in coalitions is an increasing monotonic function of the average of potential coalitional values. That is, when we increase the average coalitional values (X-axis, figure 2), keeping other parameters unchanged, more agents would join coalitions (Y-axis). An appropriate analytical curve-fit to the results is a logarithmic function[39], as in figure 2. The observation that the number of coalition members does not linearly increase (but does increase logarithmically) with respect to the increase in the average coalitional value may result from unresolvable conflicts which are present in non-super-additive environments. Such conflicts can prohibit the formation of possible coalitions.

[39]We examined linear, some polynomial, and exponential curve fittings as well. The error associated with these curves was greater than the one obtained for the logarithmic curve. This, however, does not prohibit the existence of a better fit.

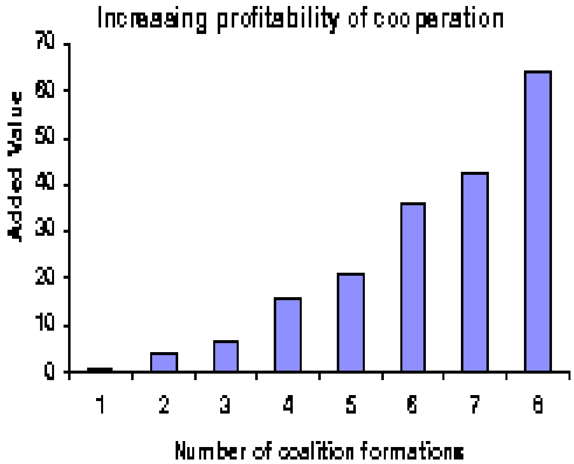

Figure 3: When the formation of coalitions increases agents' benefits, the excess benefits that result from cooperation grows as a function of the number of coalition formations.

2. In cases where cooperation is beneficial, i.e., when coalition formation can increase the agents' benefits[40], we observe (figure 3) that the additional utility (i.e., with respect to the utility without cooperation) grows as a function of the number of coalition formations. This means that not only it is beneficial to form coalitions, formation of more coalitions increases the average benefits of the agents, as expected due to the individual rationality of the agents.

3. The time necessary for coalition formation without bounding the K's is an exponential function of the number of agents that comprise the agent-system (this result was supported by simulation results presented in [37]). However, as can be observed from the graph (see figure 4), this exponent does not hold for the case of bounded K's. A quadratic curve fits the results of the simulation with a very low error ($R^2 = 0.9853$). In a system of 10 agents, where each agent is implemented on an Intel® 486 processor, the computation time per agent is approximately 3 minutes ($\varepsilon = 1$, $K_1 = 1$, $K_2 = 5$). This is a reasonable time for real-time implementation as the transportation example presented previously. In a system of 16 agents, the computation time per agent increases to 30 minutes. This time may be too long for real-time implementation. However, implementing the model on a faster machine and optimizing for the simultaneity of the agents-computations can reduce the computation time by an order of magnitude,

[40] Beneficial coalition formation was enabled by the choice of payoff functions and resource distributions, which produced coalition values where at least some of the values of larger coalitions are greater then values of smaller ones.

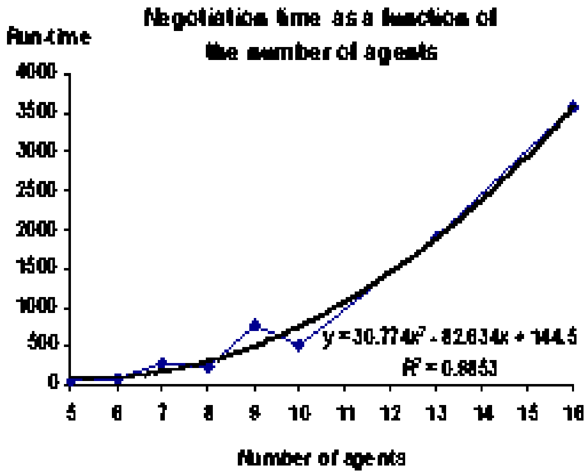

Figure 4: The time necessary for coalition formation with bounded K's appears to be a low polynomial. The best curve fit to the experimental results is quadratic.

thus even the 16-agent case will become real-time implementable.

Via the simulations, we have shown that the simulated DNPK-CFM reaches a stable PC within a reasonable time (for small K's, and only for the first stage of the DNPK-CFM). In addition, it has been found that for the settings that we have examined, the DNPK-CFM continuously improves the agents' payoffs, whether it is normally terminated or halted artificially. According to these results, the DNPK-CFM is a good coalition formation model for MAS in non-super-additive environments.

7 Related work in DAI and Game Theory

Our research applies modified game theory techniques to a DAI environment. We discuss the relevant work here.

7.1 Distributed AI

DAI researchers have been working on the complex subject of cooperation, e.g., [14, 36, 45], for constructing agents that are able to cooperate on task execution, thus increasing either their individual benefits or the system's benefits. In MAS agents are assumed to be self-interested and try to achieve their own goals and maximize their own personal payoff. In CDPS systems, agents are usually assumed to have a common goal and try to maximize a system's global payoff. Since we present a MAS model, we do not refer to CDPS.

In MAS, designers have control only over their own agents. An example of an MAS is the case of labor negotiation. Sycara [41] presented a model that combines case-based reasoning and optimization of multi-attribute utility. While Sycara deals with pairs of agents (possibly with a mediator) that have conflicting goals, we discuss multiple agents that are not necessarily in conflict with one another.

Gasser [13] focuses on the social aspects of agent knowledge and action in multi-agent systems. As in human societies, social mechanisms can dynamically emerge. Communities of programs can generate, modify and codify their own local language of interaction. Gasser's approach may be most effective when agents are interacting in environments where there is no agreed-upon, well-defined interaction mechanism, or in a continuously evolving domain. In our research we pre-define the social interaction among agents by designing coalition formation protocols.

Kraus et al [21] provide autonomous agents with negotiation mechanisms that enable them to cooperate via negotiating and reaching agreements. Their work deals with time constraints and with the efficiency of the resulting agreement. This agreement involves all of the agents, i.e., a grand coalition forms. However, conflicts among the agents may prevent an agreement and therefore can prevent cooperation, even in cases where cooperation is beneficial for some of the agents. In our research, we suggest algorithms that allow the formation of sub-groups in which the members cooperate within, thus increasing their personal payoff. As opposed to Kraus and Wilkenfeld's work, where time is included in the payoff functions of the agents, our research does not explicitly introduce time dependence, although time is considered a restricting factor.

Research by Zlotkin and Rosenschein [29] concentrates on definition and illustration of various classes of multi-agent domains. Their initial work concentrates on cooperation among pairs of agents. In their more recent work [45] coalition formation was explicitly dealt with. Here they discuss the special case of Sub-additive Task-oriented Domain (TOD), a domain which correlates with the super-additive environment. Coalition formation in such environments was discussed by Ketchpel in [18] as well. Our work mainly discusses non-super-additive multi-agent environments, and provides the designers of agents with explicit coalition-formation mechanisms for multiple agents.

Sandholm and Lesser [32] developed a coalition formation model for bounded rational agents and present a general classification of coalition games. In their model, the value of a coalition depends on the computation time, however, all configurations and possible coalitions are considered when computing a stable solution. We consider problems where the computational time of coalition values is polynomial. Therefore, we concentrate on polynomializing the number of coalitional configuration and on utility distribution and provide polynomial concepts of stability.

Ephrati and Rosenschein [11] have suggested the formation of a coalition of all of the agents (grand coalition). This formation is performed using a voting mechanism. In our work we allow formation of several coalitions, where each coalition consists of a sub-group of the agents, but not necessarily all of them. We also employ negotiation as a means for coalition-formation.

An example of coalition formation closer to MAS is the work of Sonenberg et al [39]. In their research, they deal with representation of joint actions, introduce a language for this representation, and present a centralized approach for team formation. A team formation procedure is described, where the formation process is dictated by team leaders. The team leader sends the joint goal, the joint plan, and the roles to be played to all potential team participants. This approach offers the team participants very little choice, although agents may still be individually motivated. Here we present distributed coalition-formation algorithms where no leadership is imposed on the agents. Thus, they are free to design and transmit coalition formation proposals.

7.2 Game Theory

As stated in the introduction, game theory has widely dealt with coalition formation, although mostly discussing stability and benefits distribution, and not algorithms of formation. Nevertheless we find several publications relevant to our work. The solutions for n-person games and stability will be only partially discussed here. Since we are interested in the dynamic approach to the coalition formation problem, we mainly refer to dynamic coalition formation methods and emphasize how they differ from our work and what the contribution of the DAI approach that we present is.

The dynamic theory of bargaining sets was founded by Stearns [40]. In his work, Stearns presented methods ("transfer schemes") for players in an n-person game that will enable them, starting with an arbitrary payoff allocation, *but with a coalition configuration that will not vary by the Stearns algorithm*, to reach a stable payoff configuration within the kernel [8] (presented in section 3) or the bargaining set [2]. In our work we do not concentrate on providing a method that will lead, given a specific coalition configuration, to a stable payoff allocation. Instead, we provide methods that will enable agents to begin from any initial coalition configuration and proceed using our models to *different* coalition configurations, that may be more beneficial for all of them. Nevertheless, we are interested in stable payoff allocations, and for this purpose we adopt Stearns' approach. We modify his algorithm to adjust it to the MAS with its restrictions, in order to generate proposals for payoff allocation within a given coalition configuration. This use of the modified transfer scheme enables the construction of the whole coalition formation model, wherein the transfer scheme is only a building block.

The main deficiency of the dynamic theory of bargaining sets is that its dynamics are very limited. They deal solely with situations where the coalition configuration is given and the payoffs are transferred. However, *the coalition configuration does not change.* In real dynamic environments, changes in the coalition configuration are an inseparable part of the dynamics, and such is the situation in our research. The main focus of our work is the evolution from one coalition configuration to another.

In the context of dynamic theory we find the work of Wu [43]. Wu presented a transfer scheme that converges to the core, *if* the core is non-empty. An interesting property of Wu's model is that it *implicitly* allows passing from one coalition configuration to another and does not necessarily concentrate on a specific coalition configuration. However, this property does not seem to be the essential of the model. This transfer scheme leads to a point in the core but does not provide an algorithm for passing from one coalition configuration to another, and cannot be used to pass from one point in the core to another. Our model does not necessarily lead to the core, and a coalition configuration that was reached by the model in a specific step can be the starting-point for reaching another coalition configuration in the next step of the model.

In [44], Zhou presents a new bargaining set such that both of the questions: "what coalitions will be formed?", and "what payoff vectors will be chosen?" are answered. As do other game theorists, Zhou also tries to predict which coalitions will form in a given situation. This bargaining set predicts that non-reasonable coalition configurations will not form, while other bargaining sets allow any coalition configuration to form. Zhou does not provide a dynamic algorithm for the formation of coalitions. Although Zhou's new bargaining set can be used to endogenously answer both questions of coalition formation and payoff distribution, it does not provide an answer to the questions: how will the coalitions form and what procedures will lead to this formation, which are answered in our work.

A new concept of coalitional stability was introduced by Chwe [5]. The farsighted coalitional stability introduced in this work does not determine what will happen in coalitional interaction but what can possibly happen. It defines a generalized type of a stable set, and similar to the von Neumann-Morgenstern [24] stable set, it is defined by conditions on sets' outcomes and not on individual outcomes. Such an approach does not fit the requirements of individually rational agents.

The Kernel which serves as the basis for our coalition formation algorithms is limited for TU (transferable utility) games. This limitation stems from the necessity to pass parts of the excesses to reach stability. Serrano [33] has introduced a reinterpretation of the Kernel which results in the expansion of the Kernel solution concept to the NTU games. This is an important addition to the Kernel theory, however it does not include any specification for implementing this concept among autonomous agents. As is common among game theorists, the main issue is what a stable configuration is, and not how the single agent acts in order to reach it and increase its own outcome.

Coalitions in the context of game theory were also discussed in [42, 38]. In his study, Vohra [42] discusses non-cooperative approaches to coalitions. He concentrates on the core and the bargaining set, specifically considering the consistency of cooperative and non-cooperative coalition theory. However, he does not discuss the kernel nor does he provide methods for coalition formation. A study of coalition formation is presented by [38], discussing several bargaining-set solution concepts and their properties. The research focuses on what coalitions will form and what payoffs will be assigned to the agents. However, it does not discuss the coalition formation process and the actions of the single agent within it.

A significant part of the recent research in n-person game theory discusses n-person games in which all of the agents, together, reach an agreement (e.g., [15]). Such approaches which allow for repeated games are fairly dynamic, however do not allow coalition formation (except for a grand coalition).

As can be observed, advances in game theory provide us with a good basis for our research, however they do not address the issues which are most important to our research, i.e., the explicit protocols and strategies to be followed by the agents to form coalitions.

8 Conclusion

Coalition formation is an important way to execute tasks and to maximize the payoff of autonomous agents in multi-agent environments. The purpose of a coalition formation algorithm is to reach a stable PC within a reasonable time period and with reasonable computational efforts. Since the general problem is of exponential complexity, we first developed an exponential algorithm (DEK-CFM) that reaches a Pareto-optimal solution. Then, with some significant modifications and restrictions, we reduced the complexity to a polynomial order (DNPK-CFM). The algorithms we present lead to coalition formation and payoff disbursements among the coalition members.

The DEK-CFM is a distributed algorithm that provides the agents with a stable Pareto-optimal coalition configuration and payoff vector. It is computation-oriented, best used for instances where there is a small number of agents or time and computations are cheap. The DEK-CFM is proposed as the appropriate solution for coalition formation in environments of distributed agents with sufficient computational resources.

The DNPK-CFM is negotiation-oriented. It is useful for instances where the number of agents is large (e.g., tens of agents), computations are costly and time is limited. This is because the model leads to coalition formation within a polynomial time and a polynomial amount of calculations. The ability to reach such a low complexity is due to a new approach presented – the *polynomial* K-ε-stability. The DNPK-CFM leads to distribution of both calculations and communications and to a vast reduction of the computational complexity. In addition, it is an anytime algorithm: if halted after any negotiation step, it provides the agents with a set of formed polynomial-K-ε-stable coalitions. A deficiency of the DNPK-CFM is that in polynomial time it cannot guarantee that a Pareto-optimal PC will be reached. However, for calculating a Pareto-optimal PC, all of the coalitional configurations in the CCS must be approached, and therefore there cannot be any polynomial method to find Pareto-optimality. An important advantage of our algorithm is that the average expected payoff of the agents is an increasing function of the time and effort spent by the agents performing the DNPK-CFM steps (except for the optional second stage). Therefore, if cooperation is beneficial for the agents, using the DNPK-CFM will always improve their payoffs. The last property, together with the anytime, distribution and other advantages of the DNPK-CFM have been proven via the simulations we performed.

The models we present are suitable for various multi-agent environments. They are not restricted to the super-additive environment, an environment for which there are already several coalition formation algorithms in DAI [18, 35, 45]. However, the generality of the models does not make them inapplicable. An application of the DNPK-CFM for handling the database autonomy requirement during decentralized information-gathering by rational information agents is presented in [20]. There, methods for terminological knowledge representation and inference, as well as for coalition formation among the information agents, are used. The DNPK-CFM is used, with minor modification, as the coalition formation protocol, to enable a beneficial information gathering in remote databases, while respecting the database autonomy requirements. Each database is represented by a computational agent, and the agents form coalitions.

As opposed to the majority of solution concepts presented in game theory, we present a detailed method for how the individual agent should act in order to form coalitions thus increase its personal payoff. We give examples to illustrate the models and to show their applicability. Due to their advantages, we believe that our algorithms should be adopted by designers of agents that would like their agents to cooperate and form coalitions in order to increase their benefits in both super-additive and non-super-additive environments.

References

[1] R. Aumann and S. Hart, editors. *Handbook of Game Theory with Economic Applications.* Elsevier Science Publishers, Amsterdam, 1992.

[2] R. J. Aumann and M. Maschler. The bargaining set of cooperative games. In M. Dresher, L. S. Shapley, and A. W. Tucker, editors, *Advances in Game Theory*, Princeton, N.J., 1964. Princeton University Press.

[3] M. Ben-Or and N. Linial. Collective coin flipping, robust voting games and minima of banzhaf values. In *Proc. 26th IEEE Symposium on the Foundations of Computer Science*, pages 408–416, Portland, 1985.

[4] A. H. Bond and L. Gasser. An analysis of problems and research in DAI. In A. H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 3–35. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[5] M. S. Y. Chwe. Farsighted coalitional stability. *Journal of Economic Theory*, 63:299–325, 1994.

[6] S.E. Conry, K.Kuwabara, V.R. Lesser, and R.A. Meyer. Multistage negotiation for distributed satisfaction. *IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Distributed Artificial Intelligence*, 21(6):1462–1477, December 1991.

[7] J. Contrares, F. Wu, M. Klusch, and O. Shehory. Coalition formation in a power transmission planning environment. In *Proceeding of PAAM-97*, London, 1997.

[8] M. Davis and M. Maschler. The kernel of a cooperative game. *Naval research Logistics Quarterly*, 12:223–259, 1965.

[9] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings, AAAI88*, pages 49–54, St. Paul, Minnesota, 1988.

[10] E. Ephrati and J. Rosenschein. The clarke tax as a consensus mechanism among automated agents. In *Proc. of AAAI-91*, pages 173–178, California, 1991.

[11] E. Ephrati and J. Rosenschein. Multi-agent planning as a dynamic search for a social consensus. In *IJCAI93*, pages 423–429, Chambery, France, 1993.

[12] J. W. Friedman. *Game Theory with Applications to Economics*. Oxford University Press, New York, 1986.

[13] L. Gasser. Social knowledge and social action. In *IJCAI93*, pages 751–757, Chambery, France, 1993.

[14] M.R. Genesereth, M.L. Ginsberg, and J. Rosenschein. Cooperation without communication. In *Proc. of the National Conference on Artificial Intelligence*, pages 51–57, Philadelphia, Pennsylvania, 1986.

[15] S. Hart and A. Mas-Colell. A model of n-person non-cooperative bargaining. Discussion Paper 10, Center for Rationality and Interactive Decision Theory, The Hebrew University of Jerusalem, 1992.

[16] N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence Journal*, 75(2):1–46, 1995.

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

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

[19] M. Klusch and O. Shehory. Coalition formation among rational information agents. In W. Van de Velde and J. W. Perram, editors, *Lecture Notes in Artificial Intelligence No. 1038, Agents Breaking Away*, pages 204–217. Berlin:Springer-Verlag, 1996.

[20] M. Klusch and O. Shehory. A polynomial kernel-oriented coalition formation algorithm for rational information agents. In *Proc. of ICMAS-96*, pages 157–164, Kyoto, Japan, 1996.

[21] S. Kraus, J. Wilkenfeld, and G. Zlotkin. Multiagent negotiation under time constraints. *Artificial Intelligence*, 75(2):297–345, 1995.

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

[23] J. F. Nash. Equilibrium points in n-person games. In *Proceedings of the national academy of sciences*, pages 48–49, USA, 1950.

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

[25] B. Peleg. The kernel of m-quota games. *Canadian Journal of Mathematics of Computation*, 17:239–244, 1965.

[26] B. Peleg. *Game Theoretic Analysis of Voting in Commities.* Cambridge University Press, 1984.

[27] R. Radner. Collusive behavior in noncooperative epsilon-equilibria of oligopolies with long but finite lives. *Journal of Economic Theory*, 22:136–154, 1980.

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

[29] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers.* MIT Press, Boston, 1994.

[30] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Worst-case-optimal anytime coalition structure generation. In *Proc. of AAAI-98*, Madisson, Wisconsin, 1998.

[31] T. W. 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.

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

[33] R. Serrano. Reinterpreting the kernel. Technical Report 95-22, Brown University, dept. of Economics, 1995.

[34] L. S. Shapley and M. Shubik. *Game Theory in economics.* Rand Corporation, Santa Monica, California, 1973.

[35] O. Shehory and S. Kraus. Coalition formation among autonomous agents: Strategies and complexity. In C. Castelfranchi and J. P. Muller, editors, *Lecture Notes in Artificial Intelligence No. 957, From Reaction to Cognition*, pages 57–72. Berlin:Springer-Verlag, 1993.

[36] O. Shehory and S. Kraus. Cooperation and goal-satisfaction without communication in large-scale agent-systems. In *Proc. of ECAI-96*, pages 544–548, Budapest, Hungary, 1996.

[37] O. Shehory and S. Kraus. A kernel-oriented model for coalition-formation in general environments: Implementation and results. In *Proc. of AAAI-96*, pages 134–140, Portland, Oregon, 1996.

[38] K. I. Shimomura. The bargaining set and coalition formation. Technical Report 95-11, Brown University, dept. of Economics, 1995.

[39] E. Sonenberg, G. Tidhar, E. Werner, D. Kinny, M. Ljungberg, and A. Rao. Planned team activity. Technical Report 26, Australian Artificial Intelligence Institute, Australia, 1992.

[40] R. E. Stearns. Convergent transfer schemes for n-person games. *Transactions of the American Mathematical Society*, 134:449–459, 1968.

[41] K. Sycara. Persuasive argumentation in negotiation. *Theory and Decisions*, 28:203–242, 1990.

[42] R. Vohra. Coalitional non-cooperative approaches to cooperation. Technical Report 95-6, Brown University, dept. of Economics, 1995.

[43] L. S. Wu. A dynamic theory for the class of games with nonempty cores. *Siam Journal of Applied Mathematics*, 32:328–338, 1977.

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

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