# Algorithms for Transitive Dependence-Based Coalition Formation

Bo An, Zhiqi Shen, Chunyan Miao, and Daijie Cheng

***Abstract*—Coalition formation methods allow autonomous agents to join together in order to act as a coherent group in which they increase their individual gains by collaborating with each other. Although there are some research efforts toward coalition formation in multiagent systems (MAS), such as game theory-based approaches, these methods cannot be easily applied in real-world scenarios. Based on a novel social reasoning theory, namely, transitive dependence theory, this work proposes two dynamic coalition formation algorithms for coalition formation: 1) without and-action dependence and 2) with and-action dependence, respectively. While most related work addresses the problem of searching for the optimal coalition structure (CS), the proposed algorithms aim to find out the optimal coalitions for specific goals. Theoretical analysis and experimental results suggest that 1) the algorithm for coalition formation without and-action dependence is of polynomial complexity and is efficient, and 2) when the incidence rate of and-action dependence is not high, the anytime algorithm for coalition formation with and-action dependence is also efficient although it has relatively high complexity (NP-complete).**

***Index Terms*—Coalition formation, multiagent systems, transitive dependence.**

## I. Introduction

THE decentralized control architecture has been widely used in manufacturing systems. These systems, also referred as multiagent manufacturing systems, are composed of autonomous components that communicate with other components and perform their functions without a central control [23]. Each system can be regarded as an agent that governs the operation of the system. Cooperation among autonomous agents may be mutually beneficial even if the agents are selfish [31]. A number of coalition mechanisms, widely studied in game theory and economics (e.g., [28], [29], [33]), have been successfully proposed and applied in many areas. However, most methods suffer from some limitations, for example, the coalition formation process is invisible. To avoid the limitations of game theory-based approaches, some other techniques, such as social reasoning mechanisms (e.g., [1], [5], [9], [12], [16]), are proposed to meet the needs of dynamic coalition formation in real-world agent environments.

Socially intelligent agents are autonomous problem solvers that achieve their objectives by interacting with other similarly autonomous entities. Social reasoning mechanisms have been successfully used to design and build such intelligent agents [5], [9], [35], [36]. Social reasoning refers to agents' reasoning about others. Although there are various kinds of relations in MAS, dependence relation is believed to be the most crucial one. Sichman *et al.* [36] describe fundamental concepts of a social reasoning mechanism based on dependence networks. Dependence relations exist in many application domains like supply chain, social networks, and virtual organizations. For example, in a simplified supply chain, a *Consumer Agent* depends on a *Computer Producer Agent* to produce computers and the Computer Producer Agent depends on a *Transporter Agent* to deliver computers. Understanding and reasoning with such dependence relations is essential for planning and problem solving. In [9] and [35], an abstract structure called dependence graph is used for the study of emerging social structures and the analysis of social phenomena regarding group formation and cohesiveness.

Although there is some related work with respect to dependence-based social reasoning (e.g., [1], [12], [16], [24], [34]), transitive dependence, however, has not yet been tackled. For instance, in the above supply chain example, there is a transitive dependence relation between the Consumer Agent and the Transporter Agent as if the Transporter Agent does not work, the Consumer Agent's purchase will fail. Previous work in [1] extends the related work on social reasoning by proposing a novel transitive dependence theory and presents a general framework for transitive dependence-based dynamic coalition formation.

The focus of this work is proposing algorithms for transitive dependence-based dynamic coalition formation that can generate *optimal* coalitions. Although our previous work in [1] (to the authors' best knowledge) is the first attempt to investigate transitive dependence-based coalition formation, the coalition formation mechanism in [1] is an approximate approach that will generate *suboptimal* coalitions. This research advocates the importance of transitive dependence in social reasoning and proposes two algorithms for searching potential partners. Since coalition formation is an extremely difficult and complicated task, an algorithm for simple transitive dependence based coalition formation scenarios is proposed first (see Section V), and then, the algorithm for complex transitive dependence-based coalition formation scenarios is presented (see Section VI). Additionally, Section II summarizes related work. An illustration of the problem to be solved is presented in Section III. The performance of the proposed algorithms is evaluated through

Manuscript received June 7, 2006; revised December 5, 2006 and April 16, 2007. Paper no. TII-06-06-0048.R2.

B. An is with the Computer Science Department, University of Massachusetts, Amherst, MA 01003 USA (e-mail: ban@cs.umass.edu).

Z. Shen is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail: zqshen@ntu.edu.sg).

C. Miao is with the School of Computer Engineering, Nanyang Technological University, Singapore (e-mail: ascymiao@ntu.edu.sg).

D. Cheng is with the College of Computer Science, Chongqing University, Chongqing, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2007.902255

Section VII. In the final section, the conclusion is presented and ideas for future work are outlined.

## II. Related Work

The literature of 1) game theory-based coalition formation and 2) social reasoning-based coalition formation forms a very huge collection. Space limitation precludes introducing all of them here. For a survey on game theory-based coalition formation, see [20] and [39]. The remaining part of this section only introduces and discusses some important related work on 1) applying game theory (e.g., Shapley value, kernel, and core), 2) virtual organization/enterprise creation, and 3) social reasoning mechanisms in coalition formation.

### A. Game Theory-Based Coalition Formation

The Shapley value [30] for an agent is a weighted average of all the utilities that the agent contributes to all possible coalitions. Any payoff division scheme according to the Shapley value provides an agent with the added value that it brings to the given coalition structure, averaged over all possible joining orders. Zlotkin and Rosenschein [41] present a simple coalition mechanism for sub-additive task-oriented domains based on the Shapley value. Reference [31] presents a Shapley value algorithm for coalition formation in super-additive environments. The limitation of the method in [31] is that it only can be used in super-additive environments. The Shapley value-based coalition formation mechanisms are also used in multilateral trades [40] and transmission expansion planning [10] in electric market. Conitzer and Sandholm [8] have studied a concise representation of characteristic functions that allows the agents to be concerned with a number of independent issues that each coalition of agents can address.

The kernel [13] is a payment configuration space in which the coalitional configurations are stable in the sense that there is equilibrium between pairs of individual agents in the same coalition. Suppose that there are two agents $ag_1$ and $ag_2$ in a coalition $C$. In a given payment configuration, they are in equilibrium if they cannot outweigh one another from $C$. In [32], a reduced complexity kernel-oriented coalition formation algorithm is presented. A problem of the algorithm is that in polynomial time, it cannot guarantee that a Pareto-optimal payment configuration will be reached. In [3], trusted kernel-based coalition formation is defined as a novel extension to the traditional kernel-based coalition formation process.

The core of a game with respect to a given coalition structure can be defined as the set of coalition configurations that do not necessarily have unique payoff distributions [20]. Only coalition structures that maximize the sum of all coalition values of coalitions in the considered structure are core-stable. However, searching for an optimal coalition structure is computationally difficult. Conitzer and Sandholm [7] define a concise general representation for games in characteristic form that relies on super-additivity. In [6], the authors remove some often unrealistic assumptions and propose a model that utilizes Bayesian reinforcement learning in a way that enables coalition participants to reduce their uncertainty.

Coalition structures are partitions of a set of agents into disjoint coalitions. Since finding the optimal coalition structure (CS) is NP-complete, Sandholm *et al.* [28] have presented an algorithm that establishes a tight bound within a minimal amount of search and show that any other algorithm would have to search strictly more. Reference [11] reports on a novel anytime algorithm for coalition structure generation that produces solutions that are within a finite bound from the optimal. However, the complexity of their algorithm remains exponential, and they also do not consider the case of overlapping coalitions. In [27], the authors present a novel algorithm for distributing this calculation among agents in cooperative environments.

Sandholm and Lesser [29] present 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. Shehory and Kraus [33] present several solutions to the problem of task allocation among autonomous agents.

Little work has been done in coalition formation among both self-interested and cooperative agents. Reference [22] presents a protocol that enables agents to negotiate and form coalitions and provides them with simple heuristics for choosing coalition partners. Soh and Li [37] propose an integrated multilevel learning approach to multiagent coalition formation. Reference [21] presents protocols and strategies for coalition formation with incomplete information under time constraints.

Related work in game theory mainly describes which coalitions will be formed in N-person games and how the players will distribute the benefits. These game theoretical approaches are typically centralized and computationally infeasible. Moreover, they seldom take into consideration the constraints of a multiagent environment, such as communication costs and limited computation time. Thus, it is hard to apply these approaches to real-world environments.

### B. Virtual Organization/Enterprise Creation

Virtual organization (VO) is a term being used to describe how different organizations come together to explore business opportunities and collaborating on a temporary basis [17]. For a survey on virtual organization/enterprise creation, see [25].

Reference [18] presents a model of the design and implementation processes for virtual organizations and the proposed model links the effects of managerial activity to the value created, and in doing so, it is a model of managerial affectivity. Putnik *et al.* [26] discuss the issues of *virtual enterprise integration* (*VEI*) as a contribution to a better understanding of the *VEI* phenomenon. Katzy and Dissel [19] present a framework for designing an agile virtual enterprise that can support short-term business opportunities, which is referred to as the "Value System Designer."

"Grid" computing has emerged as an important new field that focuses on large-scale resource sharing, innovative applications, and high-performance orientation. Foster *et al.* [14] define the "grid problem" as controlled and coordinated resource sharing and resource use in dynamic, scalable virtual organizations. Travica [38] discusses virtual organizations' relationship with electronic commerce. E-commerce and VO are related as on the one hand, e-commerce is conducted through virtualized

processes, on the other hand, e-commerce generates the momentum and purpose for virtualizing.

### C. Social Reasoning Mechanisms-Based Coalition Formation

Social reasoning mechanisms are considered as essential building block suitable for situations where agents may dynamically enter or leave the society, without any global control [5]. In order to acquire and use dependence knowledge, each agent has to 1) explicitly represent some properties of the other agents; 2) exploit this representation. thereby optimizing its behavior according to the evolution of the society; and 3) monitor and revise its representation to avoid inconsistencies.

An important aim in the field of multiagent systems is to study emergent social structures, such as groups and collectives. Some approaches aim in exploring social relations like power and dependence, which are based on decision-theoretic techniques [36]. Boella *et al.* distinguish four structural viewpoints on multiagent systems in an abstraction hierarchy: mind view, power view, dependence view, and coalition view [4]. Dependence relations are also used in coalition formation. David *et al.* present a utility-driven rationality and a complementary-driven rationality-based model for multiple partner coalitions. Morgado and Graca present a social reasoning mechanism that extends previous works [24]. In [16], a comparison analysis of a utility-driven method and a complementary-driven method is introduced. DEPINT [34] illustrates some essential aspects of an agent's social reasoning mechanism concerning 1) adaptation, 2) formation of coalitions, and 3) revision of inconsistent belief. Our previous work in [1] introduces transitive dependence relations and proposes a coalition formation framework. However, the proposed framework can only generate suboptimal coalitions.

This work extends the related work on social reasoning and discusses transitive dependence theory. We also propose algorithms for finding the best coalition with the least cost, which aims to find out the best set of coalition partners. The algorithms are suitable for real-world coalition formation problems as they work toward specific goals.

## III. Illustration of the Problem

The coalition problem to be solved in this paper is that of cooperative problem solving (CPS) among groups of autonomous entities. Given a problem (called a goal in this paper) of an agent, the agent has to invite other entities to form a coalition to solve the problem and, at the same time, strive to maximize its benefit while satisfying the goal. Also, the self-interested agents invited seek to get more benefits during coalition formation and problem solving.

For the ease of illustration, a simplified manufacturing example is used to demonstrate the coalition formation problem to be solved. In addition, price is considered as the only criterion.[1] More complex action dependence relations, such as or-action dependence and and-action dependence, will be discussed in Section IV. There are three factories: *furniture* factory, *cabinet* factory, and *wood* factory. Each factory can do some actions, e.g., the furniture factory can produce furniture, the cabinet factory can produce cabinets, and the wood factory can produce wood. For each action a factory can do, the factory has a reserve offer (the lowest offer it can accept) to do the action for other factories. Similarly, for each action an agent cannot perform, it also has a reserve offer (the highest offer it may propose) if it asks for other factories' help with the action. There are some action dependence relations. For example, if any other factory wants the cabinet factory to produce cabinets for it, it must produce wood for the cabinet factory.[2]

Consider that the furniture factory has a goal, which includes two actions: produce furniture and cabinets. The furniture factory can produce furniture but cannot produce cabinets, which can only be done by the cabinet factory. Thus, the furniture factory will seek the help of the cabinet factory for producing cabinets. Since the cabinet factory has an action dependence relation on producing wood about producing cabinets, it is the furniture factory's responsibility to find a factory to produce wood for the cabinet factory. Thus, the furniture factory has to ask for the wood factory's help with producing wood for the cabinet factory. The three factories have the possibility to form a coalition toward the goal (here, assume it is the only possible coalition for the furniture factory's goal).

Now we analyze whether the three self-interested factories will agree to form a coalition. First, if the highest offer the furniture factory can pay for producing cabinets is less than the reserve offer of the cabinet factory, the coalition will fail since the two factories cannot make any agreement; otherwise, the two factories can make an agreement. Assuming that the furniture factory and cabinet factory can make an agreement on producing cabinets, the furniture factory still has to consider the cabinet factory's action dependence on producing wood. If the highest offer the cabinet factory can pay for wood production is not less than the reserve offer of the wood factory, the wood factory will agree to join the coalition only if the cabinet factory pays at the wood factory's reserve price to the wood factory; otherwise, in order to form a coalition (an agent will not agree to perform an action for another agent if its reserve price of the requested action cannot be satisfied), the furniture factory has to pay the price difference between the reserve price of the cabinet factory and the wood factory when the cabinet factory pays at its reserve price to the wood factory.

The furniture factory will seek the best solution to achieve its goal. However, to find the optimal solution may be extremely difficult as there are many possible coalitions to solve the problem. For instance, in the example above, if there are two factories that can produce cabinets but with different reserve offers, the furniture factory has to make a choice on which factory to invite.

Given the existence of such action dependence relations, agents should not only consider some other entities who can help them but also take such action dependence relations into account. From the above simple example, it can be found that such problems are reasonable and are real-world problems. For instance, in some situations, producing cabinets for the

[1] In real-world coalition formation, there may be many more criteria, some of them of a subjective nature, others depending on historic data, etc. However, such multicriteria case can be formalized by combining all the criteria into an integrative criterion like utility.

[2] More complex action dependence relations, such as or-action dependence and and-action dependence, will be discussed in Section IV.

furniture factory will spend the cabinet factory's some precious resources; thus, the cabinet factory may ask the furniture factory to do other actions, not only to pay some money. Although the cabinet factory can use the payment of the furniture factory to ask some other factories to do the actions to make up the resources spent in producing cabinets for the furniture factory, there is no guarantee that it will really happen in some situations such as uncooperative environments.

The problem of the above example is generalized in this paper. We provide algorithms to solve this problem in a system of agents via the formation of coalitions and show that the algorithms are simple to be implemented, have a relatively short runtime (hence can be used as a real-time method), and yield the optimal results.

## IV. Definitions

This paper deals with coalition formation in an MAS where each agent has its goals, plans, and actions. However, a single agent may have limited abilities for a specific goal. Thus, autonomous agents may join together to reach some or all of their goals. In such a case, the agents form a coalition. In order to elucidate the problem and its solution, the following definitions and assumptions are presented (see [1] for a more detailed discussion of transitive dependence).

### A. Transitive Dependence

Let $Ag$ be a group of $n$ autonomous agents, $Ag = \{ag_1, ag_2, \ldots, ag_n\}$. Each agent $ag_i$ can perform a set of actions $A_{ag_i} = \{a_1, a_2, \ldots, a_{N_i}\}$, which represents the agent's ability. If agent $ag_i$ can perform an action $a_j$, i.e., $a_j \in A_{ag_i}$, $\text{lowest}^{ag_j}(a_j)$ is the lowest offer agent $ag_i$ would ask for if any other agent asks for the agent's help with action $a_j$; if action $a_j \notin A_{ag_i}$, i.e., agent $ag_i$ cannot perform action $a_j$, $\text{highest}^{ag_j}(a_j)$ is the highest offer agent $ag_i$ would pay when it asks for another agent to perform action $a_j$.

For each goal, there may be some plans and each plan consists of a set of actions. Let $g$ be a goal of agent $ag_i$, $P(g) = \{p_g^1, p_g^2, \ldots, p_g^k\}$ is the set of plans for goal $g$, in which $p_g^i = \{a_1, a_2, \ldots, a_m\}$.[3]

*Definition 1:* (direct dependence) Suppose agent $ag_i$ tries to achieve a goal $g$. $p_g^i = \{a_1, a_2, \ldots, a_m\}$ is a plan for goal $g$. Agent $ag_i$ has no ability to perform an action $a_k \in p_g^i$, i.e., $a_k \notin A_{ag_i}$, but it believes that agent $ag_j$ has ability to perform action $a_k$, i.e., $a_k \in A_{ag_j}$, then agent $ag_i$ directly depends on agent $ag_j$ about action $a_k$, i.e., $\text{Dep}\left(ag_i, ag_j, p_g^i, a_k\right)$.

*Definition 2:* (action dependence) Suppose agent $ag_i$ can perform an action $a_j$, but it wants any other agent who depends on it about the action $a_j$ to do an action $a_k \notin A_{ag_i}$ for it, then the agent $ag_i$ has an action dependence on the action $a_k$ about the action $a_j$, i.e., $\text{Adep}(ag_i, a_j, a_k)$.

Let $A$ be a set of actions. Agent $ag_i$ can perform an action $a_j$. If it wants any other agent who depends on it about action $a_j$ to do all the actions in $A$ for it, the action dependence is called *and-action dependence*. If it wants any other agent who depends on it about action $a_j$ to do any one action in $A$, the action dependence is called *or-action dependence*. Action dependence other than and-action dependence and or-action dependence is called *single-action dependence*.

For agents $ag_i$, $ag_j$, and $ag_k$, there are two dependence relations: $\text{Dep}\left(ag_i, ag_j, p_{g_1}^d, a_l\right)$ and $\text{Dep}\left(ag_j, ag_k, p_{g_2}^b, a_m\right)$. Although agent $ag_j$ can perform action $a_l$, it has an action dependence relation on action $a_m$ about action $a_l$, i.e., $\text{Adep}(ag_j, a_l, a_m)$. It can be found that agent $ag_i$ transitively depends on agent $ag_k$ about action $a_m$.

*Dependence chains* are used to describe the transition process of transitive dependence relations. Each dependence chain has a head and a tail. For agents $ag_i$, $ag_j$, and $ag_k$, if the following dependence relations hold: $\text{Dep}\left(ag_i, ag_j, p_{g_1}^d, a_l\right)$, $\text{Dep}\left(ag_j, ag_k, p_{g_2}^b, a_m\right)$, and $\text{Adep}(ag_j, a_l, a_m)$, the dependence chain from agent $ag_i$ to agent $ag_k$ is $Dpc = ag_i \overset{p_{g_1}^d, a_l}{\frown} ag_j \overset{p_{g_2}^b, a_m}{\frown} ag_k$.

For a dependence chain $Dpc$, $\text{Head}(Dpc)$ represents the agent at the head of the dependence chain, and $\text{Tail}(Dpc)$ represents the agent at the tail of the dependence chain. For agent $ag_i$ in a dependence chain $Dpc$, $\text{ToDep\_act}(ag_i, Dpc)$ and $\text{Deped\_act}(ag_i, Dpc)$ represent the actions that agent $ag_i$ depends on and is depended upon, respectively.[4] For the dependence chain $Dpc$ in the last paragraph, $\text{Head}(Dpc) = ag_i$, $\text{ToDep\_act}(ag_i, Dpc) = a_l$, $\text{Tail}(Dpc) = ag_k$, and $\text{Deped\_}act(ag_k, Dpc) = a_m$.

For a dependence chain $Dpc$, its transitive dependence can be described as $\text{Tdep}\left(\text{Head}(Dpc), \text{Tail}(Dpc), Dpc\right)$. For the sake of simplicity, the direct dependence is regarded as a special kind of transitive dependence. For instance, agent $ag_i$'s dependence on agent $ag_j$ about action $a_i$ that belongs to plan $p_g^i$ can be represented as $\text{Tdep}\left(ag_i, ag_j, ag_i \overset{p_g^i, a_i}{\frown} ag_j\right)$.

*Definition 3:* (transitive dependence) Transitive dependence can be recursively defined as follows: For agents $ag_i$, $ag_j$, and $ag_k$, $\text{Tdep}(ag_i, ag_j, Dpc_1) \wedge \text{Tdep}(ag_j, ag_k, Dpc_2) \wedge \text{ADep}(ag_j, \text{Deped\_act}(ag_j, Dpc_1), \text{ToDep\_act}(ag_j, Dpc_2)) \Rightarrow \text{Tdep}(ag_i, ag_k, Dpc_1 + Dpc_2)$, where $Dpc_1 + Dpc_2$ represents the connection of $Dpc_1$ and $Dpc_2$.

Transitive dependence is derived from direct dependence relations based on action dependence. Action dependence exists in multiagent systems, as well as in real society. Recall the following scenario: sometimes when you ask for one's help, no matter how much you can pay, he will not agree to help you until you promise doing something for him. As a consequence, it is intuitive to regard transitive dependence as an essential relation that is useful for cooperative problem solving and coalition formation.

### B. Coalitions

A coalition that consists of a set of agents and their requested actions can be formed based on dependence relations specified above. Thus, we define as follows.

*Definition 4:* (a coalition) For a specific goal $g$ of agent $ag_i$, a coalition $C$ to achieve the goal according to a plan $p_g^i$ consists of a set of agents represented as $Ag^C \subseteq Ag$. Agent $ag_j \in Ag^C$ in the coalition $C$ is asked to do some actions $A_{ag_j}^C \subset A_{ag_j}$;

[3] At the present stage, the order of the actions in a plan is assumed to have no effect on the achievement of the plan.

[4] An agent may appear more than once in a dependence chain.

such actions are called the *requested* actions of agent $ag_j$ for the coalition $C$.

In this paper, a coalition is formed for a specific goal of a depending agent. Given a coalition for a goal, whether the goal can be achieved by the coalition determines whether or not the coalition is feasible.

*Definition 5:* (feasible coalition) A coalition is feasible for a goal of a depending agent if, and only if, the set of agents in the coalition and their requested actions are sufficient for achieving the goal.

A feasible coalition also means that the transitive dependence relations related to achieving the goal of a depending agent are satisfied. In the manufacturing example, the furniture factory and the cabinet factory cannot form a feasible coalition since the cabinet factory's action dependence cannot be satisfied if there is no other agent (e.g., the wood factory) to produce wood for the cabinet factory.

*Definition 6:* (minimal feasible coalition) A feasible coalition is a minimal feasible coalition if, and only if, absence of any requested action of an agent in the coalition will lead the coalition to be unfeasible.

For a specific goal, a depending agent will strive to reason out a feasible coalition with the lowest cost.

### C. Assumptions

For rational agents, coalitions will take place if, and only if, each member of a coalition gains more if it joins the coalition than it could gain before. In order to make the creation of mutually beneficial coalitions possible, the following two assumptions are needed.

*Assumption 1:* (complete information) Agents' information about other members in MAS is complete.

It is obvious that agents must have some information about other members before reasoning about possible coalition partners. This kind of information can be acquired and updated dynamically.[5] Agents' information can be acquired by: 1) passive receiving. When an agent joins a multiagent system, it must present itself to the others. When an agent leaves the system, it has to tell the other members about this; 2) active inquiring. When an agent wants to know some information about another agent, it can inquire directly about the member or ask for other members' help; 3) internal reasoning. Agents can get information about the other members by reasoning.

*Assumption 2:* (payoff distribution) If a depending agent $ag_i$ asks for the help from agent $ag_j$ with action $a_m$ directly, it just need to pay $\text{lowest}^{ag_j}(a_m)$ to agent $ag_j$. If the depending agent $ag_i$ transitively depends on agent $ag_j$ about action $a_l$, and agent $ag_j$ has an action dependence relation on action $a_m$ about action $a_l$, i.e., $\text{Adep}(ag_j, a_l, a_m)$, and if an agent $ag_k$ can do action $a_m$, the following holds: if $\text{highest}^{ag_j}(a_m) < \text{lowest}^{ag_k}(a_m)$, in order to invite agents $ag_j$ and $ag_k$ to join the coalition, agent $ag_i$ has to pay the price difference $\text{lowest}^{ag_k}(a_m) - \text{highest}^{ag_j}(a_m)$ to agent $ag_k$ when agent $ag_j$ pays $\text{highest}^{ag_j}(a_m)$ to agent $ag_k$; otherwise, i.e., $\text{highest}^{ag_j}(a_m) \geq \text{lowest}^{ag_k}(a_m)$, agent $ag_i$ pays nothing to agent $ag_k$ when agent $ag_j$ pays $\text{lowest}^{ag_k}(a_m)$ to agent $ag_k$.

This work does not consider the negotiation process of bargaining for an action.[6] Given the assumption of payoff distribution, a depending agent will try to find out a feasible coalition with the least cost. For agents invited, their desires are satisfied by receiving acceptable payoffs (according to assumption 2, agents invited always receive the minimal price they ask for). According to the payoff distribution mechanism, it follows that the optimal coalition formation will be a minimal feasible coalition.

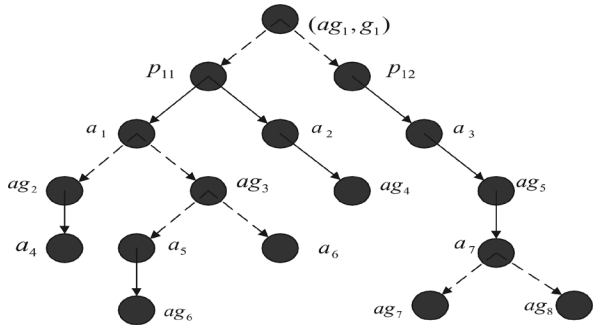


Fig. 1. Simple example.

## V. Algorithm for Coalition Formation Without and-Action Dependence

In order to enable an agent to coordinate its activities with other agents and to participate in coalitions, one of the important elements to take into account in its conception should be a proper social reasoning mechanism that allows the agent to reason about the other agents. The algorithm presented in this section intends to find out the optimal coalition in MAS environments in which there is no "and-action" dependence.

To illustrate the algorithm, a simple example is given in Fig. 1.[7] Agent $ag_1$ has a goal $g_1$ and has two alternative plans, $p_{11} = \{a_0, a_1, a_2\}$ and $p_{12} = \{a_0, a_3\}$. Suppose that agent $ag_1$ can only perform action $a_0$, and actions $a_1$ and $a_2$ can be performed, respectively, by the set of agents $\{ag_2, ag_3\}$ and $\{ag_4\}$. Agent $ag_5$ can perform action $a_3$. An action dependence relation

[5]While reasoning about the other agents to establish a coalition, an agent can adopt two different perspectives: a global perspective, which corresponds to reasoning about all the dependence relationships among the agents that can participate in the coalition, and a local perspective, in which the agent only reasons about direct dependence. In dynamic, distributed, and heterogeneous multiagent systems, the complete information assumption seems unpractical. However, when agents have incomplete information, they can take the local perspective instead of the global perspective, which is utilized in this paper while reasoning coalitions. The depending agent makes a list of possible partners, ordered by a preference measure. It then asks each agent from the list, from the top to the bottom, if it is willing to take part into the coalition. Then every agent in the list reasons about transitive dependence in the same way. The procedure stops when some agent accepts to take part into the coalition, and then the depending agent sends it a coalition formation message.

[6]Negotiation can also be utilized in our framework. Agents can negotiate for each action dependence. If the two agents related to an action dependence relation can reach an agreement, the agreement would be the solution of the final utility distribution. If they cannot reach an agreement, a depending agent still has to pay the price difference between their proposals in the final round of negotiation.

[7]Multiple choices in the graph are represented with doted lines.

**Input**: The depending agent $ag_i$'s goal $g_i$, the plan set $P(g_i)$ of plans for $g_i$, abilities and action dependence relations.
**Output**: The set of agents and their requested actions to form a coalition to achieve goal $g_i$.

The procedure for getting the best solution for goal $g_i$ includes the following four steps:

1) Get the least cost $cost_{ag_j}(a_j)$ when the depending agent $ag_i$ asks agent $ag_j$ to perform action $a_j$ in plan $p^k_{g_i}$, where $p^k_{g_i} \in P(g_i)$ and $Dep(ag_i, ag_j, p^k_{g_i}, a_j)$.
2) Get the least cost $cost_{min}(a_j)$ when the depending agent $ag_i$ asks other agents to perform action $a_j$ in plan $p^k_{g_i}$, where $\forall ag_j$, $Dep(ag_i, ag_j, p^k_{g_i}, a_j) \Rightarrow cost_{min}(a_j) \leq cost_{ag_j}(a_j)$.
3) Get the least cost $cost_{min}(p^k_{g_i})$ when the depending agent asks other agents to achieve plan $p^k_{g_i}$, where $cost_{min}(p^k_{g_i}) = \sum_{j=1}^{|p^k_{g_i}|} cost_{min}(a_j)$.
4) Choose the plan with the lowest cost $cost_{min}(p^k_{g_i})$. For all $p^j_{g_i} \in P(g_i)$, the following holds: $cost_{min}(p^k_{g_i}) \leq cost_{min}(p^j_{g_i})$.

/*The algorithm for the first step is given in Fig.3.*/

Fig. 2. Algorithm for getting the optimal coalition for a goal.

of agent $ag_2$ is $\mathrm{ADep}(ag_2, a_1, a_4)$. An or-action dependence relation of agent $ag_3$ is $\mathrm{ADep}(ag_3, a_1, a_5 \vee a_6)$. An action dependence relation of agent $ag_5$ is $\mathrm{ADep}(ag_5, a_3, a_7)$. Action $a_5$ can be performed by agent $ag_6$. Action $a_7$ can be performed by two agents $ag_7$ and $ag_8$. Actions $a_4$ and $a_6$ cannot be performed by any agent.

Suppose that there are $n$ agents, and each agent can perform at most $m$ actions. Agent $ag_i$ has a goal $g_i$. There is a set $P(g_i)$ of plans for goal $g_i$. To find the optimal plan, the depending agent $ag_i$ should get the least cost of each plan. To get the least cost of a plan, the depending agent should compute the least cost of each action in the plan that the depending agent depends on other agents. To get the least cost of an action in a plan, the depending agent should get the least cost when it asks another agent to perform the action. For instance, taking the dependence situation in Fig. 1 as an example, to decide which plan to use, the agent $ag_1$ should get the least cost of plans $p_{11}$ and $p_{12}$, respectively. To get the least cost of plan $p_{11}$, agent $ag_1$ should get the least cost of actions $a_1$ and $a_2$, respectively. To get the least cost of action $a_1$, the cost of asking agents $ag_2$ and $ag_3$ to perform action $a_1$ should be calculated, respectively, first. Therefore, the procedure for getting the optimal solution for a goal includes four steps as in Fig. 2.

The first step in Fig. 2 is the most important step. To get the least cost of asking an agent to perform an action that the depending agent depends on other agents, the depending agent should analyze all the action dependence relations related to achieving the action. A directed *dependence graph* is introduced to describe the process of computing the least cost of asking an agent to perform an action. As there are $n$ agents in MAS and each agent can perform at most $m$ actions, there are at most $n \times m$ valid vertices and one invalid vertex in the dependence graph. Each valid vertex, represented as $(ag_i, a_j)$, represents that agent $ag_i$ can perform action $a_j$. If agent $ag_i$ has an action dependence relation on action $a_j$ about action $a_i$, and agent $ag_j$ can perform action $a_j$, then there is an edge from vertex $(ag_i, a_i)$ to vertex $(ag_j, a_j)$. However, if there is no agent that can perform action $a_j$, then there is an edge from vertex $(ag_i, a_i)$ to the invalid vertex. An or-action dependence relation is treated as multiple single action dependence relations, i.e., it can be divided into several single action dependence relations.

Now discuss how to get the least cost when the depending agent $ag_i$ asks another agent $ag_j$ to perform an action $a_j$, which is represented as *root* vertex $(ag_j, a_j)$ of the dependence graph. Suppose that there are $z > 1$ vertices in the graph: $v_0, v_1, \ldots, v_{z-1}$, in which $v_0$ represents root vertex $(ag_j, a_j)$ and $v_{z-1}$ represents the invalid vertex. If root vertex has no action dependence, i.e., there is no edge from root vertex, the cost of root vertex will be the lowest offer of root vertex. When root vertex has an action dependence relation (assume this in the remaining part of this paper), to find a set of vertices to satisfy root vertex (root vertex is satisfied if and only if the action dependence of root vertex can be satisfied by other vertices), following the definition of transitive dependence, the depending agent should reason out a *feasible* path from root vertex that can satisfy root vertex $v_0$. Each path is a dependence chain. Let the feasible path be $P$, the head of the path be $\mathrm{Head}(P) = v_0$, and the tail of the path be $\mathrm{Tail}(P)$. As root vertex has an action dependence relation, to satisfy root vertex, an edge from root vertex should be connected to root vertex. Recursively, if $\mathrm{Tail}(P)$ has an action dependence, to satisfy root vertex, an edge from $\mathrm{Tail}(P)$ should be connected to $\mathrm{Tail}(P)$. Therefore, a feasible path $P$ can be constructed in accordance with the following four rules.

1) If $\mathrm{Tail}(P)$ is the head of an edge, the edge can be added into the end of path $P$. When $\mathrm{Tail}(P)$ is the head of an edge, $\mathrm{Tail}(P)$ has an action dependence relation. To satisfy root vertex, a vertex should be added at the end of the present path $P$. If $\mathrm{Tail}(P)$ is the head of more than one edge (i.e., or-action dependence), any edge can be added into the end of path $P$.
2) If $\mathrm{Tail}(P)$ appears more than once in path $P$, the path would be unfeasible since there will be a circle in the path. Root vertex cannot be satisfied if there is a circle in the path.
3) If $\mathrm{Tail}(P)$ is not the head of any edge, path $P$ ends.
4) If the tail of path $P$ is the invalid vertex, $P$ is unfeasible since the invalid vertex means that the action dependence relation cannot be satisfied.

According to the rules for constructing feasible paths, to get the least cost of asking agent $ag_j$ to perform an action $a_j$, the depending agent just needs to find the lowest cost of the paths from root vertex to any valid vertex that 1) can be reached by root vertex; and 2) is not the head of any edge. A vertex that can be reached by root vertex means that the vertex is related to the achievement of root vertex. A vertex that is not the head of any edge means that the vertex has no action dependence relation. If there is a feasible path, root vertex can be satisfied since each action needed to be performed due to action dependence can be satisfied. Obviously, root vertex can be satisfied if, and only if, there is at least one feasible path. Therefore, to get the least cost of root vertex, we just need to find the feasible path with the lowest cost. According to assumption 2, the cost of a feasible

path is the sum of what the depending agent should pay for the action of root vertex and all the action dependence relations in the path.

Taking the edge from $(ag_j, a_j)$ to $(ag_k, a_k)$ in a feasible path as an example, according to the payoff distribution mechanism, if $\text{highest}^{ag_j}(a_k) \geq \text{lowest}^{ag_k}(a_k)$, the depending agent has to spend nothing; otherwise, the depending agent has to spend $\text{lowest}^{ag_k}(a_k) - \text{highest}^{ag_j}(a_k)$. Thus, we can define how much the depending agent should spend for an action dependence relation as the weight of the corresponding edge. The weight of an edge is defined as: 1) if the tail of the edge is the invalid vertex, the weight of the edge is INF; 2) otherwise, taking the edge from vertex $(ag_j, a_j)$ to vertex $(ag_k, a_k)$ as an example, the weight of the edge is $\max(0, \text{lowest}^{ag_k}(a_k) - \text{highest}^{ag_j}(a_k))$.

The algorithm for getting the least cost when the depending agent asks an agent to perform an action in a plan is in Fig. 3. The dependence graph is represented as an adjacent matrix. This algorithm first calculates the shortest distances from root vertex to all other vertices (step 1). Then, the sum of the lowest offer of root vertex and the minimum of the shortest distances of all the feasible paths is selected as the least cost of root vertex (step 2). Finally, $\text{mincost}$ is the least cost of root vertex $v_0$, and $P(v)$ represents the set of agents and their requested actions to form coalition.

**Complexity of the algorithm**: The efficiency of this algorithm should be judged from computations. First, we evaluate the complexity of the algorithm for getting the least cost when the depending agent asks an agent to perform an action in a plan (see Fig. 3). Step 1 of the algorithm in Fig. 3 is similar to Dijkstra's Shortest Path Algorithm [2, pp. 167–172]; thus, the complexity of step 1 is $O(z^2)$. The complexity of step 2 is also $O(z^2)$. Therefore, the algorithm for getting the least cost when a depending agent asks an agent to perform the action in a plan is $O(z^2)$, i.e., $O(n^2 \times m^2)$ since $z = n \times m + 1$. Assume that there are at most $k$ plans for each goal and there are at most $l$ actions in each plan. According to the procedure for getting the best solution for a specific goal, the complexity of the algorithm for coalition formation without and-action dependence is $O(n^2 \times m^2 \times k \times l)$. The algorithm is of polynomial complexity.

## VI. Algorithm for Coalition Formation With and-Action Dependence

The algorithm in this section aims to find the "best" coalitions for specific goals in an MAS in which there are and-action dependence relations.

When there is and-action dependence, the procedure for getting the best solution for a goal is still similar to the procedure for coalition formation without and-action dependence (see Fig. 2). Also, the first step in Fig. 2 is the most important. When there is no and-action dependence, to get the least cost of root vertex, a depending agent just needs to find the feasible path with the lowest cost. With and-action dependence, however, a feasible path cannot guarantee that root vertex can be satisfied. Therefore, to get the least cost of asking an agent to perform an action, the depending agent should search all the paths from root vertex and find a coalition of agents with the least cost. Thus, with and-action dependence, the problem of finding the lowest cost of root vertex becomes much more difficult.

**Input**: The dependence graph and root vertex
**Output**: The least cost of root vertex and the corresponding set of agents and their requested actions.

An adjacent matrix $G[z][z]$ is used to describe the dependence graph, for any vertices $i$ and $j$, if there is an edge from vertex $i$ to vertex $j$, then the weight of the edge is equal to $G(i,j)$, otherwise $G(i,j)$ =INF. Let $P[z][z]$ be the matrix to store the shortest roads from vertex $v_0$ to other vertices. $P(v,w)$ is TRUE only if vertex $w$ is a vertex in the shortest path from vertex $v_0$ to vertex $v$. $D[z]$ is used to store the shortest distances from vertex $v_0$ to other vertices. $final[z]$ is used as flags. $final(v)$ is TRUE only if the shortest distance from vertex $v_0$ to vertex $v$ has been calculated. $lowest(root)$ is the lowest offer of root vertex.

The algorithm includes the following two steps:
1) Get the shortest distance from vertex $v_0$ to all other vertices.

```
for(v = 0; v < z; v = v + 1){
  final(v) =FALSE; D(v) = G(v0, v);
  for(w = 0; w < z; w = w + 1) P(v, w) =FALSE;
  if(D(v) <INF) {P(v, v0) =TRUE;P(v, v) =TRUE;}}
D(v0) = 0; final(v0) =TRUE;
for(i = 1; i < z; i = i + 1){
  min =INF;
  for(w = 1; w < z; w = w + 1)
    if(!final(w))
      if(D(w) < min){v = w; min = D(w);}
  final(v) =TRUE;
  for(w = 1; w < z; w = w + 1)
    if(!final(w)&&(min + G(v, w) < D(w))){
      D(w) = min + G(v, w);
      P(w) = P(v); P(w, v) =True; } } }
```

2) Get the least cost $mincost$ of root vertex and the set of agents to form coalition.

```
mincost =INF; D(v0) =INF;
for(i = 0; i < z; v = v + 1){
  feasible =TRUE;
  for(w = 1; w < z - 1; w = w + 1)
    if(G(i, w) ≠ INF) feasible =FALSE;
  if(feasible&&D(i) < mincost){
    mincost = D(i) + lowest(root); v = i;} }
```

Fig. 3. Algorithm for getting the least cost of root vertex.

*Theorem 1:* With and-action dependence, getting the least cost of root vertex (LCR) is NP-complete.

*Proof:* Membership of NP is easy: given an instance, simply guess the action in root vertex can be satisfied by a set of agents and their requested actions (a solution). The size of the solution is bounded by $(n \times m)$ and, since $n$ and $m$ are finite, guessing can be done in polynomial time. Thus, verifying the feasibility and cost of a solution in a dependence graph can be done in polynomial time.

For completeness, we must show that LCR is in some sense no easier than all other NP-complete problems. To do this, it suffices to show that any instance $I$ of some known NP-complete problems can be transformed into an instance of $\tau(I)$ of LCR such that the transformation can be done in polynomial time, and the transformed problem $\tau(I)$ has a solution only if the original problem $I$ has a solution. For LCR, we define a reduction from the well-known 0–1 Knapsack problem.

*Formal definition of 0–1 Knapsack problem*: There is a knapsack of capacity $c > 0$ and $N$ items. Each item has value $v_i > 0$

TABLE I
A 0–1 KNAPSACK PROBLEM INSTANCE

| $N=3$ | $c=10$ | | |
|---|---|---|---|
| $i$ | 1 | 2 | 3 |
| $v_i$ | 2 | 5 | 4 |
| $w_i$ | 5 | 7 | 6 |

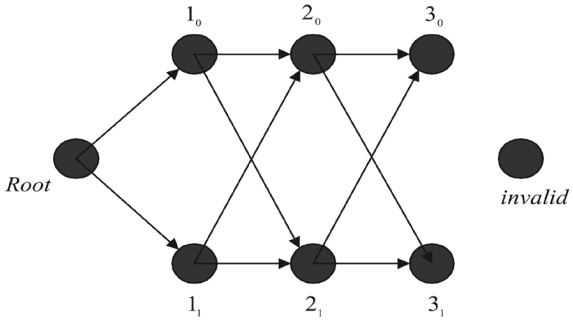


Fig. 4. Dependence graph for example Knapsack-01.

and weight $w_i > 0$. Find the selection of items ($\delta_i = 1$ if selected, 0 if not) that fit, $\sum_{i=1}^{N} \delta_i w_i \leq c$, and the total value, $\sum_{i=1}^{N} \delta_i v_i$, is maximized.

To see how the reduction works, consider the example Knapsack-01 illustrated in Table I. The most valuable set of items in this example is $\{2, 3\}$. We transform the example into an instance $\tau(\text{Knapsack} - 01)$ of LCR. To do this, we first create all the vertices for the dependence graph. We create two vertices for each item $i$: $i_0$ and $i_1$. We also create a root vertex and an invalid vertex. The edges between the vertices are defined as: 1) there is one edge from root vertex to the two vertices of item 1; 2) for each vertex of item $i$ ($i = 1, 2, 3$), there is an edge from the vertex to vertices $(i+1)_0$ and $(i+1)_1$, respectively. There is no edge connected with the invalid vertex. The weight of each edge is defined as: the weight of the edge connected to vertex $i_0$ is 0, and the weight of the edge connected to vertex $i_1$ is -$w_i$.

The dependence graph transformed from the example Knapsack-01 is showed in Fig. 4. Then the depending agent begins to search all the feasible paths from root vertex, for example, root $\rightarrow 1_0 \rightarrow 2_1 \rightarrow 3_0$. Vertex $i_0$ means that item $i$ is not in the selection of items, and vertex $i_1$ means that item $i$ is in the selection of items. From all the feasible paths from root vertex, the path that fits in the fixed volume of the knapsack and has the most valuable set of items will be chosen as the solution. For the example Knapsack-01, the path results in the most valuable set of items that fit in a knapsack of fixed volume is root $\rightarrow 1_0 \rightarrow 2_1 \rightarrow 3_1$. The transformation from the 0–1 Knapsack problem to LCR is entirely automatic (see Fig. 5) and is polynomial. ■

Here we introduce an algorithm for solving the problem of getting the lowest cost of root vertex for coalition formation with and-action dependence (see Fig. 6). The algorithm is still based on the dependence graph (see Section VI for details), in which an and-action dependence relation is still described as multiple single action dependence relations. When the function $LCR(i, \text{history})$ is called, the depending agent will compute the least cost of that which have passed all the vertices in history, where history is the set of vertices that have been visited from root vertex to vertex $i$. Let $V(i)$ be the set of vertices that have edges from vertex $i$. If vertex $i$ appears in history, the path is not feasible since there is a circle, then the cost of vertex $i$ is $LCR(i, \text{history}) = INF$; if vertex $i$ is the invalid vertex, the path is not feasible since there is an action dependence relation that cannot be satisfied, i.e., $LCR(i, \text{history}) = INF$; if vertex $i$ is an or-action dependence vertex or a single action dependence vertex, the depending agent will compute the least cost of all the vertices $V(i)$ that have passed all the vertices in history $\cup\, i$, and the minimum of the sum of the least cost of each vertex $v \in V(i)$ and the weight of the edge from vertex $i$ to vertex $v$ will be chosen as the cost of vertex $i$; if vertex $i$ is an and-action dependence vertex, the depending agent will

---

Let $N$ be the number of items of different values and volumes. For each item $i$, create two vertices $i_0$ and $i_1$. Create a *root* vertex and an invalid vertex. Draw one edge from *root* vertex to the two vertices of item 1.
For each vertex $i_0$ or $i_1$, $i < n$
Draw one edge from the vertex to vertices $(i+1)_0$ and $(i+1)_1$ respectively.
The weight of the edge connected to vertex $i_0$ is 0, and the weight of the edge connected to vertex $i_1$ is $-w_i$.

---

Fig. 5. Reducing the 0–1 Knapsack problem to LCR.

---

**Input**: The dependence graph and root vertex
**Output**: The least cost of root vertex and the corresponding set of agents and their requested actions.

Let $history$ be the set of vertices has been searched from root vertex. $LCR(i, history)$ is a function, where $i$ is the vertex whose value is to be computed. The definitions of the dependence graph, the adjacent matrix $G[z][z]$ and $lowest(root)$ are the same as in Fig. 3.

```
LCR(i, history){
  If vertex i appears in history or is the invalid vertex
    Return INF;
  If vertex i is an or-action dependence vertex or a single
action dependence vertex {
    Put vertex i in history, let min =INF;
    For each other vertex v, where G(i,v)! = INF, i.e., vertex
v has an edge from vertex i
      If min < (G(i,v) + LCR(v, history))
        min = (G(i,v) + LCR(v, history))
    Return min;
  }
  If vertex i is an and-action dependence vertex {
    Put vertex i in history, let min = 0;
    For each other vertex v, where G(i,v)! = INF{
      min = min + G(i,v) + LCR(v, history)
      if min = INF break; }
    Return min; }
  Otherwise Return 0;
}
```

/*The least cost of root vertex $v_0$ is $lowest(root) + LCR(v_0, history)$, where $history = \emptyset$. The procedure for getting the corresponding set of vertices according to the least cost of root vertex is omitted here.*/

---

Fig. 6. Getting the lowest cost of root vertex.

1. if there is no solution that can satisfy root vertex, return;
2. using the depth-first search method, get a solution to satisfy root vertex and keep the solution if it's the first solution;
3. else if it's better (i.e., lower cost) than the kept solution, replace the solution with the current one;
4. go to step 1.

Fig. 7. Anytime algorithm.

compute the least cost of all the vertices $V(i)$ that have passed all the vertices in $\text{history} \cup i$, and the sum of the least cost of each vertex $v \in V(i)$ and the weight of the edge from vertex $i$ to vertex $v$ will be chosen as the cost of vertex $i$. If $V(i) = \emptyset$, vertex $i$ is a valid vertex and $i \notin \text{history}$, the cost of the vertex will be zero. To get the least cost of root vertex $v_0$, we just need to call the function $LCR(v_0, \text{history})$, where $\text{history} = \emptyset$, and the sum of the lowest offer of root vertex and $LCR(v_0, \text{history})$ is the least cost of root vertex $v_0$.

To reduce the computational loads of each vertex, a depending agent can use some memory to record the value it has computed. When it is asked to compute the value of a vertex in a path, the depending agent first searches its memory to find whether the asked computation exists. If the result of the computation is in the memory, it just needs to report the result directly; otherwise, it has to compute the least cost of the vertex according to the algorithm in Fig. 6.

**Complexity of the algorithm**: Since there are $z$ vertices in the graph, the worst case of the number of the paths the depending agent has to search from root vertex is $(z-1)!$. Therefore, the computational complexity of the algorithm for getting the least cost of root vertex is $O((z-1)!)$, i.e., $O((n*m)!)$. Assume that there are at most $k$ plans for each goal and there are at most $l$ actions in each plan. According to the procedure for getting the best solution for a specific goal, the complexity of the algorithm for coalition formation with and-action dependence is $O((n*m)! \times k \times l)$. Compared with the algorithm for coalition formation with or-action dependence, the algorithm in this section is of high complexity.

Given the NP-complete property of the problem of coalition formation with and-dependence, it will take much longer time to run the algorithm in complex environments. To produce a coalition in limited computing time, an anytime algorithm is presented in Fig. 7. Using the depth-first search method, the algorithm first gets a solution, and then, it continues to search for the lowest value of root vertex. Anytime algorithms offer a tradeoff between solution quality and computation time that has proved useful in solving time-critical problems such as planning and scheduling, belief network evaluation, and information gathering [15]. From the anytime algorithm in Fig. 7, it can be found that the best solution so far will be chosen as the final solution when the deadline approaches. Moreover, if time resource is limited, the solution for a plan can be the final solution, and the optimal solution for other plans can be calculated if the time permits.

TABLE II
INPUT DATA SOURCES

| | |
|---|---|
| $n$ | *The number of agents in the market* |
| $m$ | *The number of actions an agent can perform* |
| $k$ | *The number of plans for a goal* |
| $p_a$ | *The probability of an action of an agent having an action dependence relation* |
| $p_{or}$ | *The probability of an action dependence relation being an or-action dependence relation* |
| $p_{and}$ | *The probability of an action dependence relation being an and-action dependence relation* |

## VII. EXPERIMENTATION

### A. Testbed

To realize the idea of transitive dependence-based coalition formation, a simulation testbed that consists of a virtual marketplace, a society of agents, and a controller was implemented. The controller generates agents, randomly determines their parameters (e.g., abilities, reserve offers, goals, plans), and simulates the entrance of agents to the virtual marketplace. Using the testbed, a series of experiments were carried out in order to demonstrate the features of the transitive dependence-based coalition formation method and to evaluate the effectiveness of the proposed algorithms. To evaluate the performance of the algorithms for coalition formation in a wide variety of test environments, agents are subject to different market densities and some other parameters (see Table II).

### B. Experimental Settings

All the six input parameters in Table II are generated randomly following a uniform distribution. The number of agents in the market determines the market's density. With less agents, the market density decreases. The number of actions an agent can perform represents the agent's ability. For a specific goal, the value of $k$ determines how many plans can be used to fulfill the goal. If agent $ag_i$ can do an action $a_j$, $p_a \in [0,1]$ represents the probability that the agent $ag_i$ has an action dependence relation about the action $a_j$. For an action dependence relation, $p_{\text{or}} \in [0,1]$ represents the probability that the action dependence relation is an or-action dependence relation. Similarly, $p_{\text{and}} \in [0,1]$ represents the probability that an action dependence relation is an and-action dependence relation. Obviously, it follows that $0 \leq (p_{\text{or}} + p_{\text{and}}) \leq 1$. $1 - p_{\text{or}} - p_{\text{and}}$ is the probability that an action dependence relation is a single-action dependence relation. Moreover, given the fact that an agent $ag_i$ can do an action $a_j$, $p_a \times p_{\text{or}}$ (respectively, $p_a \times p_{\text{and}}$) is the probability that the agent $ag_i$ has an or-action dependence relation (respectively, an and-action dependence relation) on other actions about the action $a_j$. With the increase of $p_{\text{or}}$ (respectively, $p_{\text{and}}$), the number of or-action dependence relations (respectively, and-action dependence relations) in the market increases.

### C. Performance Measure

The performance measures include average processing time and standardized utility (see Table III). Reducing average processing time of coalition formation is a very important perfor-

TABLE III
PERFORMANCE MEASURE

| | |
|---|---|
| *Standardized Utility* | $U_{stand} = U_{fact}/U_{optimal}$ |
| *Average Processing Time* | $P_{time} = \sum_{i=1}^{N_{total}}(T_{end}^{i})/N_{total}$ |
| $N_{total}$ | *Total number of coalitions* |
| $U_{fact}$ | *Average utility of all the coalitions* |
| $U_{optimal}$ | *Average optimal utility of all the coalitions* |
| $T_{end}^{i}$ | *The processing time spent in the ith coalition formation process* |

mance measure, especially for scenarios with bounded computational resources [29]. Although our algorithms can guarantee that the results would be optimal, the anytime algorithm for coalition formation with and-action dependence has high complexity. If there is limited time resource, evaluation of the distance from the practical coalition results to the optimal results is an essential criteria. The standardized utility investigates how well the actually gained coalition values are as compared with the optimal coalition values.

### D. Results

An extensive amount of stochastic simulations were carried out for many different coalition formation scenarios. Due to space limitation, only some representative results are presented in the remaining part of this section. In these experiments, to evaluate the performance of the proposed coalition formation algorithms in relatively complex environments, assume that the number of actions an agent can perform is 30, the number of plans for a goal is 6, and the probability $p_a$ of an action of an agent having an action dependence relation is 0.9.

### E. Observation 1

The experimental results in Fig. 8 show that, with the increase of agents, the average processing time increases. In Fig. 8, when the number of agents increases from 10 to 100, the average processing time increases from 0.04 to 2.50 s when $p_{\text{or}} = 0.4$. Also, it can be found that $P_{\text{time}}$ increases more and more rapidly. This result corresponds to the intuition that the coalition formation reasoning becomes much harder with the increase with agents.

The simulation results in Fig. 8 also suggest that, with different $p_{\text{or}}$, $P_{\text{time}}$ almost has no difference for the same market density. In Fig. 8, when $p_{\text{or}} = 0.2$ and $p_{\text{or}} = 0.8$, the average processing times are 0.40 and 0.39 s, respectively, when the number of agents is 40.

### F. Observation 2

From the experimental results in Fig. 9, it can be found that, with the increase of agents, the average processing time increases in coalition formation with and-action dependence. In Fig. 9, when the number of agents increases from 10 to 100, the average processing time increases from 0.02 to 0.69 s when $p_{\text{and}} = 0.4$. Unlike the results in Section VII-E, $P_{\text{time}}$ has an almost linear increase in Fig. 9.

From Fig. 9, it can be found that $P_{\text{time}}$ increases with the increase of $p_{\text{and}}$ for the same market density. In Fig. 9, when $p_{\text{and}} = 0.4$ and $p_{\text{and}} = 0.8$, the average processing times are 0.25 and 0.36 s, respectively, when the number of agents is 50.

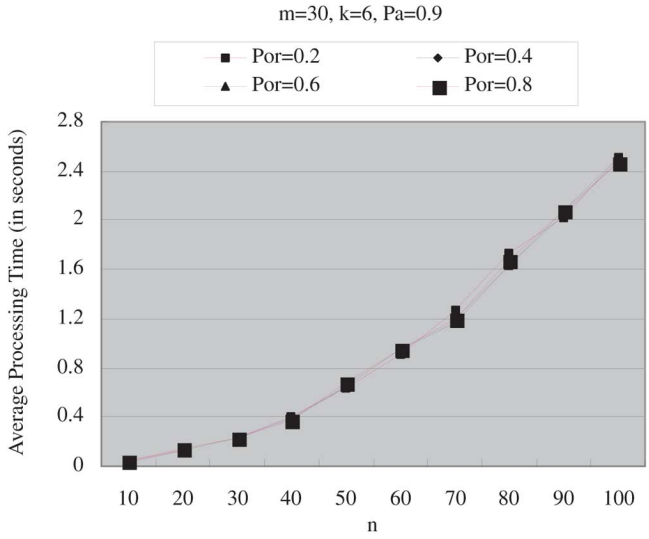


Fig. 8. Agents and average processing time.

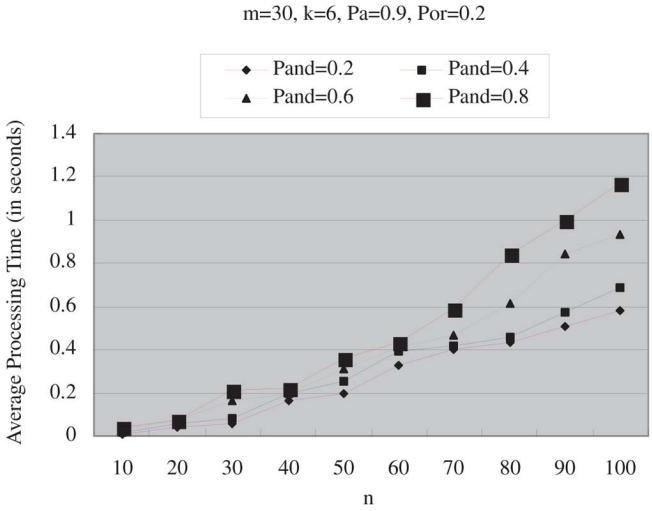


Fig. 9. Agents and average processing time.

The results coincide with the results in Fig. 10, where the average processing time gradually increases with the increase of $p_{\text{and}}$. In Fig. 10, $P_{\text{time}}$ increases from 0.14 to 0.32 s when $p_{\text{and}}$ increases from 0.10 to 1.

### G. Observation 3

The algorithm for coalition formation with and-action dependence is an anytime algorithm (see Section VI). Therefore, the depending agent may find a solution before finding out the optimal solution. Since bounded time is a commonly used assumption, it is necessary to evaluate how well a coalition formation result is when there is not enough time to get the optimal result. In this experiment, let $T_{\max}$ be the time enough for running the algorithm and $t$ be the limited practical time to run the algorithm. From the experimental results in Fig. 11, it can be found that the standardized utility increases with the increase of $t/T_{\max}$, i.e., the coalition results are more closed to the optimal coalition results. Especially when $t/T_{\max} > 0.7$, the practical coalition results are very closed to the optimal results (e.g.,

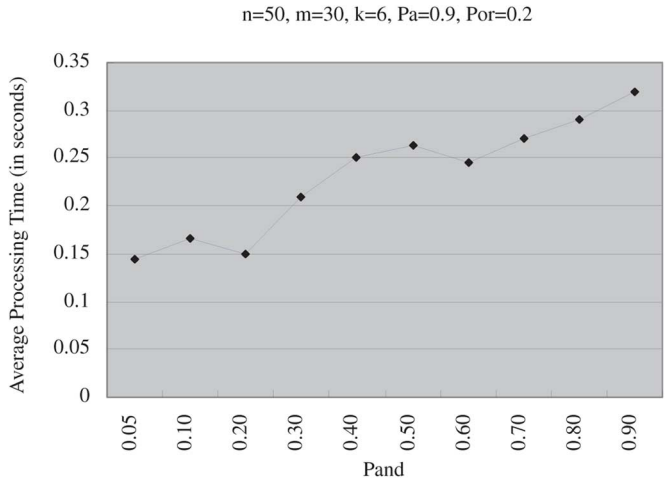


Fig. 10. $p_{\text{and}}$ and average processing time.

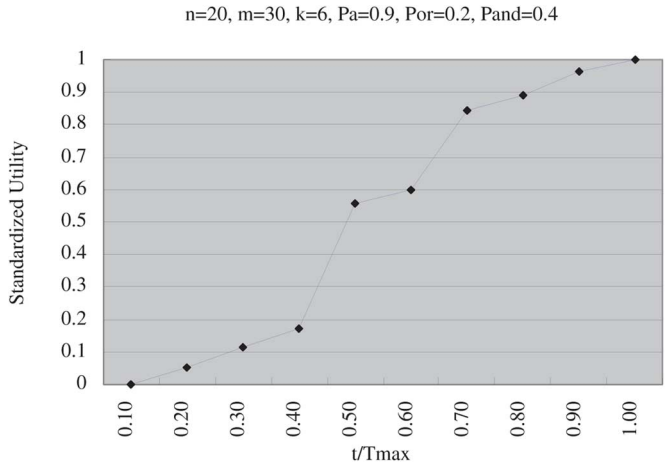


Fig. 11. $p_{\text{and}}$ and standardized utility.

in Fig. 11, the standardized utilities are always higher than 0.8 when $t/T_{\max} > 0.7$).

## VIII. Conclusion

The main contribution of this paper is complementing our previous work by proposing coalition formation algorithms that can generate optimal coalitions. This paper does not address the problem of finding the optimal division of agents into coalitions such that the total of the payoffs to all the coalitions is maximized but aims to propose efficient coalition formation algorithms for specific goals pursuit. When there is no and-action dependence, the proposed algorithm is of polynomial complexity. Although the problem of coalition formation with and-action dependence is NP-complete, experimental results show that the algorithm is still efficient. Compared with other coalition formation methods, the transitive dependence-based approaches are more intuitive.

The proposed framework is inspiring for dynamic coalition formation although. The proposed coalition formation algorithms can be applied in many application domains, such as multiagent manufacturing systems, in autonomic and service oriented computing, dynamic web/grid service composition, virtual organizations (VOs) formation in service-oriented grid where all the entities can be treated as autonomous agents, supply chain management, workflow, and enterprise integration. In addition, our approach can be extended to 1) handle resource conflict situations when more agents require the same resource at the same time, and 2) solve much more complex application problems in which some other relations are also involved or there are more evaluation criteria.

Finally, a future agenda of this work is further analyzing the problem of transitive dependence-based coalition formation with and-action dependence and proposing possible algorithms (e.g., distributed algorithms) with polynomial complexity toward the NP-complete problem.

## Acknowledgment

The authors would like to thank the associate editor and four anonymous reviewers for their valuable comments and suggestions for improving this paper.

## References

[1] B. An, C. Miao, and D. Cheng, "A coalition formation framework based on transitive dependence," *IEICE Trans. Inf. Syst.*, vol. E88-D, no. 12, pp. 2672–2680, 2005.

[2] S. Baase, *Computer Algorithms Introduction to Design and Analysis*, 2nd ed. Reading, MA: Addison-Wesley, 1988.

[3] B. Blankenburg, R. K. Dash, S. D. Ramchurn, M. Klusch, and N. R. Jennings, "Trusted kernel-based coalition formation," in *Proc. 4th Int. Joint Conf. Autonomous Agents and Multiagent Systems*, 2005.

[4] G. Boella, L. Sauro, and L. van der Torre, "Social viewpoints on multiagent systems," in *Proc. 3rd Int. Joint Conf. Autonomous Agents and Multiagent Systems*, 2004, pp. 1358–1359.

[5] C. Castelfranchi, A. Cesta, and M. Miceli, "Dependence relations in multi-agent systems," *Decentralized AI—3*, 1992.

[6] G. Chalkiadakis and C. Boutilier, "Bayesian reinforcement learning for coalition formation under uncertainty," in *Proc. 3rd Int. Joint Conf. Autonomous Agents and Multiagent Systems*, 2004, pp. 1090–1097.

[7] V. Conitzer and T. Sandholm, "Complexity of determining nonemptiness of the core," in *Proc. 18th Int. Joint Conf. Artificial Intelligence*, Aug. 2003, pp. 613–618.

[8] V. Conitzer and T. Sandholm, "Computing Shapley values, manipulating value division schemes, and checking core membership in multi-issue domains," in *Proc. 19th Nat. Conf. Artificial Intelligence*, 2004, pp. 219–225.

[9] R. Conte and J. S. Sichman, "Dependence graphs: Dependence within and between groups," *Comput. Math. Organiz. Theory*, vol. 8, no. 2, pp. 87–112, 2002.

[10] J. Contreras and F. F. Wu, "Coalition formation in transmission expansion planning," *IEEE Trans. Power Syst.*, vol. 14, no. 3, pp. 1144–1152, Aug. 1999.

[11] V. D. Dang and N. R. Jennings, "Generating coalition structures with finite bound from the optimal guarantees," in *Proc. 3rd Int. Conf. Autonomous Agents and Multiagent Systems*, 2004, pp. 564–571.

[12] N. David, J. S. Sichman, and H. Coelho, "Extending social reasoning to cope with multiple partner coalitions," in *Proc. 9th Eur. Workshop Modelling Autonomous Agents in a Multi-Agent World*, 1999, pp. 175–187.

[13] M. Davis and M. Maschler, "The kernel of a cooperative game," *Naval Res. Logist. Quart.*, vol. 12, pp. 223–259, 1965.

[14] I. Foster, C. Kesselman, and S. Tuecke, "Anatomy of the grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, no. 3, pp. 200–222, 1992.

[15] E. A. Hansen and S. Zilberstein, "Monitoring and control of anytime algorithms: A dynamic programming approach," *Artif. Intell.*, vol. 126, no. 1–2, pp. 139–157, 2001.

[16] M. Ito and J. S. Sichman, "Dependence based coalitions and contract net: A comparative analysis," in *Proc. Int. Joint 7th Ibero-American Conf. Artificial Intelligence and 15th Brazilian Symp. Artificial Intelligence*, 2000, pp. 106–115.

[17] B. Katzy and G. Sung, "State-of-the-art of virtual organization modeling," in *Building the Knowledge Economy: Issues, Applications, Case Studies*. Tokyo, Japan: Ohmsha, 2003, pp. 959–966.
[18] B. R. Katzy, "Design and implementation of virtual organizations," in *Proc. 31st Hawaii Int. Conf. System Sciences*, Jan. 1998, pp. 142–151.
[19] B. R. Katzy and M. Dissel, "A toolset for building the virtual enterprise," *J. Intell. Manufactur.*, vol. 12, no. 2, pp. 121–131, 2001.
[20] M. Klusch and A. Gerber, "Dynamic coalition formation among rational agents," *IEEE Intell. Syst.*, vol. 17, no. 3, pp. 42–47, May/Jun. 2002.
[21] S. Kraus, O. Shehory, and G. Taase, "The advantages of compromising in coalition formation with incomplete information," in *Proc. 3rd Int. Joint Conf. Autonomous Agents and Multiagent Systems*, 2004, pp. 588–595.
[22] S. Kraus, O. Shehory, and G. Tasse, "Coalition formation with uncertain heterogeneous information," in *Proc. 2nd Int. Joint Conf. Autonomous Agents and Multiagent Systems*, 2003, pp. 1–8.
[23] N. Krothapalli and A. Deshmukh, "Design of negotiation protocols for multi-agent manufacturing systems," *Int. J. Prod. Res.*, vol. 37, no. 7, pp. 1601–1624, 1999.
[24] L. Morgado and G. Graca, A Social Reasoning Mechanism Based on a New Approach for Coalition Formation, Instituto Superior de Engenharia de Lisboa, 2000, Tech. Rep.
[25] C. Petrie and C. Bussler, "Service agents and virtual enterprises: A survey," *IEEE Internet Comput.*, vol. 7, no. 4, pp. 68–78, Jul./Aug. 2003.
[26] G. Putnik, M. M. Cunha, R. Sousa, and P. Avila, "Virtual enterprise integration: Challenges of a new paradigm," in *Virtual Enterprise Integration: Technological and Organizational Perspectives*. Hershey, PA: IGI, 2005, pp. 1–30.
[27] T. Rahwan and N. R. Jennings, "Distributing coalitional value calculations among cooperating agents," in *Proc. 20th Nat. Conf. Artificial Intelligence*, 2005.
[28] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme, "Coalition structure generation with worst case guarantees," *Artif. Intell.*, vol. 111, no. 1–2, pp. 209–238, 1999.
[29] T. W. Sandholm and V. Lesser, "Coalitions among computationally bounded agents," *Artif. Intell.* (Special Issue on Economic Principles of Multi-Agent Systems), vol. 94, no. 1, pp. 99–137, 1997.
[30] L. S. Shapley, "A value for n-person game," in *Contributions to the Theory of Games*. Princeton, NJ: Princeton Univ. Press, 1953.
[31] O. Shehory and S. Kraus, "Coalition formation among autonomous agents: Strategies and complexity," in *From Reaction to Cognition*. New York: Springer Verlag, 1995, pp. 57–72.
[32] O. Shehory and S. Kraus, "A kernel-oriented model for autonomous-agent coalition-formation in general environments: Implementation and results," in *Proc. 11th Nat. Conf. Artificial Intelligence*, Aug. 1996, pp. 134–140.
[33] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artif. Intell.*, vol. 101, no. 1–2, pp. 165–200, 1998.
[34] J. S. Sichman, "Depint: Dependence-based coalition formation in an open multi-agent scenarios," *J. Artif. Soc. Social Sim.*, vol. 1, no. 2, Mar. 1998. [Online]. Available: http://jasss.soc.surrey.ac.uk/1/2/3.html.
[35] J. S. Sichman and R. Conte, "Multi-agent dependence by dependence graphs," in *Proc. 1st Int. Joint Conf. Autonomous Agents and Multiagent Systems*, Jul. 2002, pp. 483–492.
[36] J. S. Sichman, R. Conte, Y. Demazeau, and C. Castelfranchi, "A social reasoning mechanism based on dependence networks," in *Proc. 16th Eur. Conf. Artificial Intelligence*, 1994, pp. 188–192.
[37] L. Soh and X. Li, "An integrated multilevel learning approach to multiagent coalition formation," in *Proc. 18th Int. Joint Conf. Artificial Intelligence*, 2003, pp. 619–624.
[38] B. Travica, "Virtual organization and electronic commerce," *DATA BASE Adv. Inf. Syst.*, vol. 36, no. 3, pp. 45–68, Jul./Aug. 2005.
[39] G. Vauvert and A. E. Fallah-Segrouhni, "Coalition formation among strong autonomous and weak rational agents," in *Proc. 10th Eur. Workshop Modelling Autonomous Agents in a Multi-Agent World*, May 2001.
[40] C. S. K. Yeung, A. S. Y. Poon, and F. F. Wu, "Game theoretical multiagent modelling of coalition formation for multilateral trades," *IEEE Trans. Power Systems*, vol. 14, no. 3, pp. 929–934, Aug. 1999.
[41] G. Zlotkin and J. S. Rosenschein, "Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains," in *Proc. 9th Nat. Conf. Artificial Intelligence*, 1994, pp. 432–437.

**Bo An** received the B.Sc. and M.Sc. degrees in computer science from Chongqing University, Chongqing, China, in 2003 and 2006, respectively. Currently, he is pursuing the Ph.D. degree in the Computer Science Department, University of Massachusetts, Amherst.

His research interests are in autonomous agent, multiagent systems, and electronic commerce, especially in coalition formation and automated negotiation.

**Zhiqi Shen** received the B.Sc. degree in computer science from Peking University, Peking, China, and the Ph.D. degree in computer engineering from the Information Communication Institute, Nanyang Technological University, Singapore.

His research interests include goal-oriented modelling, cognitive modelling, intelligent software agent, software engineering, semantic web/grid, sensor network and their applications in education innovation, integrated bio-manufacturing services, interactive games, and various e-services. His current research focuses on infusing agent technology into autonomic service-oriented computing and invisible computing. He has worked in both university and industry in many large funded R&D projects in China, Singapore, and Canada.

**Chunyan Miao** received the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University, Singapore.

She is currently an Assistant Professor with the School of Computer Engineering, Nanyang Technological University. Her major research interest includes machine learning, intelligent software agent, agent mediated semantic web/grid, and agent-oriented software engineering.

**Daijie Cheng** received the B.Sc. degree in mathematics from Beijing University of Aeronautics and Astronautics, Beijing, China, in 1958.

Currently, he is a Professor at the College of Computer Science, Chongqing University, Chongqing, China. His research focuses on parallel computing, networks, intelligent e-commerce, and data mining.