IEICE TRANS. INF. & SYST., VOL.E88–D, NO.12 DECEMBER 2005

PAPER IEICE/IEEE Joint Special Section on Autonomous Decentralized Systems

A Coalition Formation Framework Based on Transitive Dependence

Bo AN[†a)], **Chunyan MIAO**[††b)], *and* **Daijie CHENG**[†], *Nonmembers*

SUMMARY Coalition formation in multi-agent systems (MAS) is becoming increasingly important as it increases the ability of agents to execute tasks and maximize their payoffs. Dependence relations are regarded as the foundation of coalition formation. This paper proposes a novel dependence theory namely transitive dependence theory for dynamic coalition formation in multi-agent systems. Transitive dependence is an extension of direct dependence that supports an agent's reasoning about other social members during coalition formation. Based on the proposed transitive dependence theory, a dynamic coalition formation framework has been worked out which includes information gathering, transitive dependence based reasoning for coalition partners search and coalition resolution. The nested coalitions and how to deal with incomplete knowledge while forming coalitions are also discussed in the paper.

key words: *coalition formation, transitive dependence, multi-agent systems*

1. Introduction

Autonomous agents situated in dynamic, open, and complicated multi-agent environments may need to cooperate to achieve their goals. Thus agents may form coalitions to achieve goals that can't be accomplished by individuals. Cooperation among autonomous agents may be mutually beneficial even if the agents are selfish and try to maximize their own expected payoffs [11]. Mutual benefit may arise from resource sharing and task redistribution. Coalition formation is important for agent cooperation in multi-agent environment. A number of coalition mechanisms have been successfully proposed and applied into many areas [9]–[13], [17].

Social reasoning refers to agents' reasoning about others. The social reasoning mechanism is considered to be an essential building block of autonomous agents. Although there are various kinds of relations in multi-agent systems (MAS) as what in a real society, the dependence relation is believed to be one of the most crucial ones. With dependence relations, an agent is able to dynamically choose a goal to pursue and a plan to achieve as it is sure every skill it needs to accomplish the selected plan is available in MAS.

Social reasoning mechanisms, especially dependence based methods, play an important role in coalition formation in MAS. Dependence relations are regarded as the foundation of coalition formation [2]. There have been some research efforts with respect to dependence based coalition formation (e.g., [4], [8], [14], [15]). However, transitive dependence has not yet been considered in the existing research. For instance, if an agent ag_i depends on an agent ag_j about an action a_1, and the agent ag_j depends on an agent ag_k about an action a_2, is there dependence relation between the agent ag_i and the agent ag_k? The answer is definitely not. Then under what condition will there exist dependence between the agent ag_i and the agent ag_k? If there is a dependence relation derived from the two direct dependence relations, this kind of dependence is called *transitive dependence*. This research advocates the importance of transitive dependence, an extension of direct dependence, in social reasoning in MAS and proposes a coalition formation framework based on transitive dependence. Moreover, search for potential partners is a crucial problem in dynamic coalition formation. This research also presents a transitive dependence based method for searching coalition partners including dependence tree generation, dependence tree reduction, plan optimization and action optimization.

The remainder of this paper is organized as follows. Section 2 presents the transitive dependence theory. In Sect. 3, we propose the transitive dependence based coalition formation mechanism. Nested coalitions and coalition formation with incomplete information will be discussed in Sect. 4. Related work will be given in Sect. 5. In the final section, some conclusions are presented and ideas for future work are outlined.

2. Transitive Dependence Theory

A multi-agent system is composed of agents, agent environments, agent organizations, agent interaction and various relationships between agents. The organization structure in MAS includes issues such as the organization style, relations between members, which evolve during agents' cooperation problem solving.

As social reasoning mechanisms are based on agents' information about the others, an agent must explicitly represent certain properties concerning other agents [14]. For the agents ag_i and ag_j, $D_{ag_i}(ag_j)$ is the set of desires, $A_{ag_i}(ag_j)$ is the set of actions, $R_{ag_i}(ag_j)$ is the set of resources, $P_{ag_i}(ag_j)$ is the set of plans the agent ag_i believes the agent ag_j has. A plan consists of a sequence of actions with its associated resources needed to accomplish them. $DP_{ag_i}(ag_j)$ is the set of

Manuscript received March 31, 2005.
Manuscript revised June 28, 2005.

[†]The authors are with the College of Computer Science, Chongqing University, 400044 Chongqing, P.R. China.
[††]The author is with the School of Computer Engineering, Nanyang Technological University, 639497 Singapore.
a) E-mail: anbolangzhong@sohu.com
b) E-mail: ascymiao@ntu.edu.sg

DOI: 10.1093/ietisy/e88–d.12.2672

action dependence relations the agent ag_i believes the agent ag_j has. The definition of action dependence will be discussed later.

We call a depending agent the depender, and the agent who is depended upon the dependee. The object around which the dependence relationship centres is called the dependum [16]. A depender (or dependee) can be any member in a multi-agent system, e.g., a single agent, an organization, etc. According to dependence property, we divide dependence relations into strong dependence and weak dependence.

Definition 1: (Strong dependence) Suppose an agent ag_i tries to achieve a goal g. $p(ag_k, g) = \{a_1, a_2, \ldots, a_n\}$ is a plan of the agent ag_k for the goal g. The agent ag_i has no ability to achieve an action $a_i \in p(ag_k, g)$, but it believes that the agent ag_j has ability to achieve the action a_i, then the agent ag_i strongly depends on the agent ag_j about the action a_i, i.e., $Sdep(ag_i, ag_j, p(ag_k, g), a_i)$.

Definition 2: (Weak dependence) An agent ag_i tries to achieve a goal g. There is a plan of an agent ag_k for the goal g, $p(ag_k, g) = \{a_1, a_2, \ldots, a_n\}$. The agent ag_i can achieve the action $a_i \in p(ag_k, g)$ by itself, but it also believes that the agent ag_j has ability to achieve the action a_i if it pays $offer_{ag_i \to ag_j}(a_i)$ to the agent ag_j. If the agent ag_i achieves the action a_i by itself, it should spend $cost_{ag_i}(a_i)$, and $cost_{ag_i}(a_i) > offer_{ag_i \to ag_j}(a_i)$, then the agent ag_i weakly depends on the agent ag_j about the action a_i, i.e., $Wdep(ag_i, ag_j, p(ag_k, g), a_i)$.

The definition of strong dependence is the same as it is in most related work. Although weak dependence has not been addressed in previous research, weak dependence relations exist in multi-agent systems, as well as in real society. For example, a robot who is good at washing clothes can also wash dishes. Nevertheless it has to spend much more time than letting a dishwasher robot do it. Therefore, the clothes-washer robot can request the dishwasher robot to wash dishes, and it can wash clothes for the dishwasher robot. In this way, it helps to save the cost and to increase efficiency through cooperation between the dishwasher robot and the robot who is good at washing clothes. Another simple example is the Tireworld [17], where every agent can move tiles. After analysis of weak dependence, agents can save cost through cooperation.

Although there is some research work about dependence relations, little research work has been reported with respect to transitive dependence relations. Due to the fact that transitive dependence relationship is an important form of dependence relationships in MAS, this paper aims to identify, define and model such relationships so that agents in MAS can mutually benefit each other. Here we give formal definitions related to transitive dependence. First we define action dependence relations.

Definition 3: (Action dependence) Suppose the agent ag_i can achieve the action a_i, but it wants any other agent who depends on it about the action a_i to do an action a_j for it, and it has no ability to achieve the action a_j, then the agent ag_i has an action dependence on the action a_j about the action a_i, i.e., $Adep(ag_i, a_i, a_j)$.

Let A be a set of actions. The agent ag_i can achieve the action a_i. If it wants any other agent who depends on it about the action a_i to do all the actions in A, we call this kind of action dependence *and-action dependence*. If it wants any other agent who depends on it about the action a_i to do any one action in A, we call this kind of action dependence *or-action dependence*.

Now we discuss transitive dependence. For the agents ag_i, ag_j and ag_k, there are two dependence relations: $Dep(ag_i, ag_j, p(ag_m, g_a), a_i)$ and $Dep(ag_j, ag_k, p(ag_q, g_b), a_j)$, where Dep can be $SDep$ or $WDep$. Although the agent ag_j can achieve the action a_i, but it has an action dependence on the action a_j about the action a_i, i.e., $Adep(ag_j, a_i, a_j)$. We can find that the agent ag_i transitively depends on the agent ag_k about the action a_j.

In contrast to the transitive dependence, we call the dependence relations defined in definition 1 and definition 2 direct dependence.

Dependence chain is used to describe the transition process of transitive dependence relations. A dependence chain has a head and a tail. For the agents ag_i, ag_j and ag_k, $Dep(ag_i, ag_j, p(ag_m, g_a), a_i)$, $Dep(ag_j, ag_k, p(ag_q, g_b), a_j)$, and $Adep(ag_j, a_i, a_j)$, the dependence chain from the agent ag_i to the agent ag_k is $Dpc = ag_i \frac{p(ag_m, g_a), a_i}{} ag_j \frac{p(ag_q, g_b), a_j}{} ag_k$.

For a dependence chain Dpc, $Head(Dpc)$ represents the agent at the head of the dependence chain, and $Tail(Dpc)$ represents the agent at the tail of the dependence chain. For an agent ag_i in a dependence chain Dpc, $ToDep\_act(ag_i, Dpc)$ and $Deped\_act(ag_i, Dpc)$ represent the actions that the agent ag_i depends on and is depended upon respectively†. For the dependence chain Dpc in the last paragraph, $Head(Dpc) = ag_i$, $ToDep\_act(ag_i, Dpc) = a_i$, $Tail(Dpc) = ag_k$, and $Deped\_act(ag_k, Dpc) = a_j$.

Definition 4: Transitive dependence can be described as *Tdep(Depender, Dependee, Dependence chain)*. For the sake of simplicity, the direct dependence is regarded as a kind of special transitive dependence, i.e., the agent ag_i's dependence on the agent ag_j about the action a_i that belongs to the plan $p(ag_k, g)$ can be described as $Tdp(ag_i, ag_j, ag_i \frac{p(ag_k, g), a_i}{} ag_j)$. Transitive dependence can be recursively defined as:

For the agents ag_i, ag_j and ag_k, $Tdep(ag_i, ag_j, Dpc_1) \wedge Tdep(ag_j, ag_k, Dpc_2) \wedge ADep(ag_j, Deped\_act(ag_j, Dpc_1)), ToDep\_act(ag_j, Dpc_2)) \Rightarrow Tdep(ag_i, ag_k, Dpc_1 + Dpc_2)$, where $Dpc_1 + Dpc_2$ represents the connection of Dpc_1 and Dpc_2.

According to dependence property, transitive dependence can be divided into strong transitive dependence and weak transitive dependence. $TSdep(ag_i, ag_j, Dpc)$ means the agent ag_i strongly transitively depends on the agent ag_j

†An agent may appear more than once in a dependence chain.

and $TWdep(ag_i, ag_j, Dpc)$ represents the agent ag_i weakly transitively depends on the agent ag_j.

According to definition 3, the transitive dependence relation $Tdep(ag_j, ag_k, Dpc_2)$ in definition 4 should be strong transitive dependence relation. For the agents ag_i, ag_j, ag_k, $Tdep(ag_i, ag_j, Dpc_1)$, $Tdep(ag_j, ag_k, Dpc_2)$ and $Tdep(ag_i, ag_k, Dpc_1 + Dpc_2)$. If the agent ag_i strongly transitively depends on the agent ag_j, the agent ag_i strongly transitively depends on the agent ag_k, otherwise the agent ag_i weakly transitively depends on the agent ag_k.

3. Coalition Formation Framework

In order to enable an agent to co-ordinate 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. In order to make the creation of mutually beneficial coalitions possible, we make the following two assumptions:

Assumption 1: *Complete information*

We assume that agents' information about the other members in MAS is complete. The information is stored in the data structure described in section 2.

Assumption 2: *Personal rationality*

We assume that each agent in the environment has personal rationality, i.e., it joins a coalition only if it can benefit at least as much within the coalition as it could benefit by itself or by joining in other coalitions. For instance, an agent ag_i can do an action a_i, if another agent ag_j ask for its help with the action a_i, and the agent ag_j can pay more than it wants and other agents will pay, the agent ag_i will join the coalition to do the action for the agent ag_j.

Basically, agents interact and form coalitions because they depend on one another in order to achieve their own goals. In our coalition formation framework, the coalition formation consists of three stages: information gathering, transitive dependence based reasoning, and coalition resolution.

3.1 Information Gathering Phase

It's obvious that agents must have some information about other members in the multi-agent system before reasoning about possible coalition partners. This kind of information is acquired during an initial information gathering phase and can be acquired and updated dynamically. The information can be acquired by:

- Passive receiving. When an agent joins a multi-agent system, it must present itself to the others, sending some information to introduce itself, e.g., capabilities, resources, goals, etc. While an agent leaves the multi-agent system, it has to tell the other members about this.
- Active inquiring. When an agent wants to know some information about a member in agent society, it can inquire directly about the member or ask for other members' help.
- Internal reasoning. Agents can get information about the other members by internal reasoning.

3.2 Transitive Dependence Based Reasoning Phase

The Transitive dependence based reasoning is the most important phase in the three stages of coalition formation. In this phase, the depender reasons about potential partners based on dependence relations between the potential partners and itself. The dependence relations are got by analyzing the information gathered in the first phase. For example, agents can reason out transitive dependence relations based on direct dependence relations and action dependence relations.

In the transitive dependence based reasoning phase, after analysis of the dependence relations, the reasoning process for potential coalition partners includes the following steps: 1) *dependence tree generation*; 2) *dependence tree reduction*; 3) *plan optimization*; and 4) *action optimization*.

Assume an agent ag_i has a goal g, and $p(ag_k, g)$ is a plan of the agent ag_k for the goal g. If the agent ag_i has no ability to achieve an action $a_i \in p(ag_k, g)$, the agent ag_i strongly depends on other agents about the action a_i that it can't achieve; If the agent ag_i has ability to achieve the action a_i but it has to spend more than what it pays for some other agents when it asks for the help of them, the agent ag_i weakly depends on other agents about the action a_i. The kind of direct dependence and transitive dependence in solving a goal are represented with a dependence tree.

Definition 5: A dependence tree DPT is an ordered triple $(V(DPT), E(DPT), \Psi(DPT))$ consisting of a nonempty set $V(DPT)$ of nodes, a set $E(DPT)$, disjoint from $V(DPT)$, of edges and an incidence function $\Psi(DPT)$ that associates with each edge of DPT an ordered pair of (not necessarily distinct) vertices of DPT.

1. The set $V(DPT) = Root \cup V_{plan}(Root) \cup V_{ag}(DPT) \cup V_{act}(DPT)$ is the union of four disjoint sets. *Root* is the root node of the dependence tree, which represents the goal of the depending agent, $V_{plan}(Root)$ is the set of plans the agents may use to achieve the goal of the *Root* node, $V_{ag}(DPT)$ is the set of agents, and $V_{act}(DPT)$ is the set of actions.
2. The set $E(DPT)$ is a set of edges.
3. The function $\Psi_{DPT} : E(DPT) \rightarrow V(DPT) \times V(DPT)$ is defined as follows:
   a. $\Psi_{DPT}(e_1) = (Root, p_i)$ associates an edge e_1 with an ordered pair of vertices $(Root, p_i)$, and represents the fact that the goal of the *Root* node can be achieved by the plan p_i.
   b. $\Psi_{DPT}(e_2) = (p_i, a_i)$ associates an edge e_2 with an ordered pair of vertices (p_i, a_i), and represents the

fact that the plan p_i needs the action a_i and the action can't be achieved by the depender or the depender can achieve it but has to spend more.

c. $\Psi_{DPT}(e_3) = (a_i, ag_i)$ associates an edge e_3 with an ordered pair of vertices (a_i, ag_i), and represents the fact that the action a_i can be performed by the agent ag_i.

d. $\Psi_{DPT}(e_4) = (ag_i, a_i)$associates an edge e_4 with an ordered pair of vertices (ag_i, a_i), and represents the fact that the agent ag_i has an action dependence on the action a_i, i.e., $ADep(ag_i, a_k, a_i)$, where the action a_k is the origin action node of the agent ag_i.

Here we define two notations for the dependence tree. Let a node x be any node but the *Root* node in a dependence tree DPT. $father(x)$ represents the father node of the node x, and $sons(x)$ represents the set of nodes whose father node is the node x. Each node x has only a father node $father(x)$, and may has more than one son node, i.e., $|sons(x)| \geq 0$.

In a dependence tree, an agent node or an action node may appear more than once. But if an agent node appears two times in a road from the *Root* node, their father nodes should not be the same (because in this case the depending agent can not evaluate the dependence chain since there is a cycle). A dependence tree has the following characteristics:

- For an agent node x in a dependence tree, its father node, an action node, $a_i = father(x)$ and any son node $a_j \in sons(x)$ (if $|sons(x)| \geq 1$) belongs to an action dependence relation $ADep(x, a_i, a_j)$.
- Let the node x is an action node in a dependence tree, for its father node $ag_i = father(x)$(if it's an agent node) and any son node, an agent node, $ag_j \in sons(x)$ (if $|sons(x)| \geq 1$), it's obvious that the maximum offer of the agent ag_i about the action node x is no less than the reserve price of the agent ag_j.
- For an agent node x in a dependence tree, its father node is $a_i = father(x)$ and the set of its son nodes is $sons(x)$ ($|sons(x)| \geq 1$). The node x is an "or" node if the agent x or-depends on the set $sons(x)$ of actions about the action a_i, and the node x is an "and" node if the agent and-depends on the set $sons(x)$ of actions about the action a_i.

To illustrate the dependence tree, a simple example as follows is given and the corresponding dependence tree is in Fig. 1[†]. An agent ag_1 has a goal g_1 and it has two alternative plans, $p_{11} = a_0, a_1, a_2$ and $p_{12} = a_0, a_3$ for the goal. Suppose that the agent ag_1 can only perform the action a_0, and the actions a_1 and a_2 can be performed respectively by the set of agents $\{ag_2, ag_3\}$ and $\{ag_4\}$. The agent ag_5 can achieve the action a_3. An action dependence relation of the agent ag_2 is $ADep(ag_2, a_1, a_4)$. An or-action dependence relation of the agent ag_3 is $ADep(ag_3, a_1, a_5 \vee a_6)$. An and-action dependence relation of the agent ag_5 is $ADep(ag_5, a_3, a_7 \wedge a_8)$. The action a_5 can be performed by the agent ag_6. The action a_7 can be performed by the agents ag_7 and ag_8. The action a_8 can be performed by the agent ag_9. The actions a_4 and a_6 can't be performed by any agent.

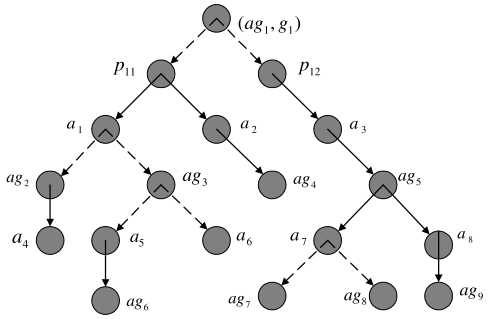

Fig. 1 A simple example of dependence tree.

Definition 6: (Feasible action) For an action node a_i in a dependence tree, it's a feasible action if it can be achieved, which can be defined by:

If the prior node of the action node a_i is a plan node, and the depender has ability to achieve the action a_i, the action a_i is a feasible action.

If an edge e associates with an ordered pair of vertices (a_i, ag_j) and the node ag_j is a leaf node, the action a_i is feasible.

If an edge e associates with an ordered pair of vertices (a_i, ag_j), the node ag_j is an "or" node, and there is more than one ordered pair of vertices (ag_j, a_k), in which the action a_k is feasible, then the action a_i is a feasible action.

If an edge e associates with an ordered pair of vertices (a_i, ag_j), the node ag_j is an "and" node, and for every ordered pair of vertices (ag_j, a_k), the action a_k is feasible, then the action a_i is a feasible action.

Definition 7: (Feasible plan) For a plan p_i in a dependence tree, if for every ordered pair of vertices (p_i, a_i), the action a_i is feasible, the plan p_i is a feasible plan.

After deletion of the all actions and plans that are not feasible from a dependence tree, the reduced tree is called a *reduced dependence tree*.

Take the dependence tree in Fig. 1 as an example. Assume that the agent ag_2 has no ability to achieve the action a_4, and the agent ag_3 has no ability to achieve the action a_6. Figure 2 shows the reduced dependence tree after dependence tree reduction.

For a reduced dependence tree, if the node *Root* is a leaf node, then the goal is not achievable; otherwise, it's a feasible goal.

In a reduced dependence tree, there may be more than one feasible plan for the goal of the *Root* node. Similarly, for an action node, there may be more than one agent which can achieve the action. Thus, the depending agent should make a decision on choosing the best plan for the goal and best agent for an action. Best here refers to the plan (action)

[†]The or-dependence relations are represented with doted lines, and the and-dependence relations are represented with solid lines

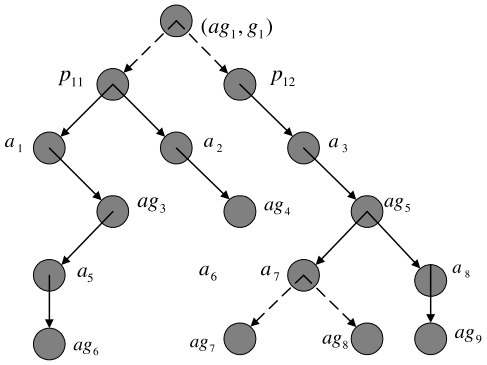

Fig. 2 A reduced dependence tree.

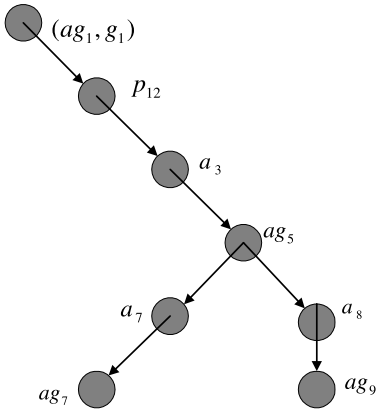

Fig. 3 A coalition tree.

can bring about the most favorable utility for the depending agent. It's obvious that both the two problems are of high complexity and there is a strong dependence between the two problems. To avoid the high complexity of the two problems, here we utilize a solution of bounded rationality. The two problems with mutual influence are divided into two independent problems: *plan optimization* and *action optimization*, which will reduce the computational complexity of finding the best plan and the best cooperation partner, but, accordingly, the coalition result is not optimal.

For a feasible goal, there may be more than one feasible plan, and we choose the most favorable plan to form coalition. This is called Plan optimization. Similarly, for a feasible action, there may be more than one agent which can achieve it, and the agent that will result in the most favorable outcome is chosen to achieve the action. This is called Action optimization.

For a reduced dependence tree, after plan optimization and action optimization, the reduced dependence tree is called a *coalition tree*.

Definition 8: (Dependence based coalition formation) Given a framework $F =< AG, G, Plan, Can, Dp >$, an agent $ag_i \in AG$ has a goal $g \in G$, according to the set of agents' ability Can and the set of dependence relations Dp, does there exist a coalition tree over F that is feasible for the goal g?

Theorem 1: The dependence based coalition formation problem is NP-complete.
Proof: Proof can be found in Appendix. □

It is well known that many problems in graph theory are NP hard on general graphs. Fortunately in most cases we can limit the graphs to a restricted class to reduce the computation complexity. In this paper, we give an anytime algorithm [5] to find the best solution in the allowed time range. Anytime algorithms has been regarded as novel effective methods for solving classic planning problems which are NP complete.

Algorithm Suppose Index() is the index function to evaluate a solution. It returns a numeric value. The smaller number indicates a better solution. The minimum index value indicates the optimal solution.

1. if there is no feasible plan/action, return;
2. select a feasible plan/action;
3. compute its index value using index function;
4. if it is the first solution, keep the solution and the index value;
5. else if the index value is less than the kept one, replace the solution with the current one and keep the index value;
6. else go to 1. □

From the anytime algorithm, we can find that the best solution so far will be chosen as the final solution when the deadline approaches. Thus, at any time, there is a dynamic coalition can be formed.

Take the reduced dependence tree is in Fig. 2 as an example. Let us assume that the agent ag_1 has to pay much more if it adopts the plan p_{11} than that if it adopts the plan p_{12}, and the reserve price of the agent ag_7 is less than that of the agent ag_8 for the action a_7. Figure 3 shows the coalition tree after plan optimization and action optimization.

3.3 Coalition Resolution Phase

In this phase, the depending agent invites all the agents in the coalition tree to form a coalition. According to assumption 1 and assumption 2, all the agents in the coalition tree will agree to join in the coalition for cooperative problem solving.

4. Discussions

4.1 Nested Coalitions

The coalition formation framework in this paper is based on transitive dependence, in which the coalition formation will result in nested coalitions [8].

When reasoning about the other agents to establish a coalition, an agent can adopt two different perspectives: a global perspective, which corresponds to a reasoning about all the dependence relationships among the agents that can participate in the coalition (including transitive dependence

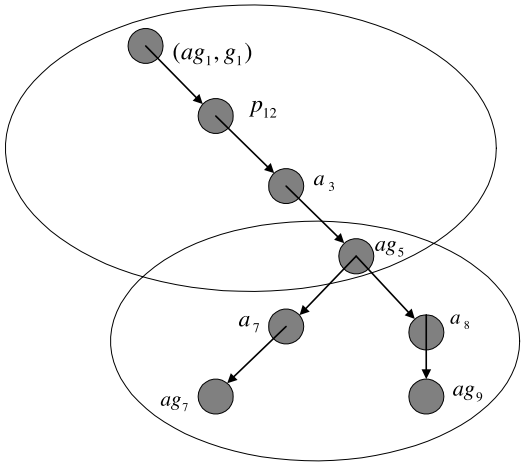

Fig. 4 Nested coalitions.

and direct dependence), and a local perspective, in which the agent only reasons about direct dependence.

In our framework, every agent node in the dependence tree adopts the global perspective to reason about the fulfillment of its goal based on transitive dependence, and it adopts the local perspective to form nested coalitions. The achievement of the goals of nested coalitions will results in the success of the goal of their father agent node.

With an approach based on a local perspective, the achievement of a goal for which an agent is not autonomous may originate not only one coalition, but a group of nested coalitions. The nested coalitions of the coalition tree in Fig. 3 are showed in Fig. 4.

4.2 With Incomplete Information

In dynamic, distributed and heterogeneous multi-agent systems, the information may be incomplete. Now if we relax the assumption 1, i.e., the information may be not complete, does our coalition formation framework based on transitive dependence still work? If so, how to form coalitions with incomplete information?

In this situation, each agent in the dependence tree has incomplete knowledge of all the social members, which means that each agent doesn't clearly know all the details about the accomplishment of the intended goal. Therefore, the agent has to take the local perspective. 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.

5. Related Work

Distributed artificial intelligence (DAI) is concerned with problem solving in which several agents interact in order to execute tasks. Coalition formation and coalition planning is a well-investigated area in the field of DAI. Coalition formation can be categorized into two approaches: utility-based and complementary-based [6]. Most classic methods for forming stable coalitions among rational agents follow the utility-based approach and rely on derived concepts from cooperative game theory. However, game theoretical approaches are typically centralized and computationally infeasible. MAS researchers, using game theory concepts, have developed algorithms for coalition formation in MAS environments. However, many of them suffer from a number of important drawbacks, for example: they only consider super-additive environments [11], [17]. Sandholm and Lesser have developed a coalition formation model for bounded rational agents and present a general classification of coalition games [9]. Shehory and Kraus present several solutions to the problem of task allocation among autonomous agents [12]. Sandholm *et al*. present an algorithm that establishes a tight bound within this minimal amount of search [10]. [13] presents algorithms for coalition formation and payoff distribution in non-super-additive environments. Compared with the game theory based coalition formation mechanisms, our method seems more intuitive since dependence is the reason for forming coalitions. Moreover, game theory based algorithms generally can only be applied to super-additive environments and are of high computational complexities.

An important aim in the field of Multi-agent Systems is to study emergent social structures, such as groups and collectives. Some approaches aimed in exploring social relations like power and dependence are based on Decision-theoretic techniques [14]. Boella *et al*. distinguish four structural viewpoints on multi-agent systems in an abstraction hierarchy: mind view, power view, dependence view and coalition view [2]. From the point view of dependence property, Alonso divides dependence relation into two kinds: strong dependence and weak dependence [1]. Dependence relations are also used in coalition formation [14]. According to the nature of the dependum, Yu et al. divides dependence relation into goal dependency, task dependency, resource dependency, and soft-goal dependency [16]. The related research on dependence theory ignored the importance of transitive dependence. In contrast, this research proposes a set of formal definitions of transitive dependence.

Dependence relations, to our knowledge, are firstly used in coalition formation in [14]. David *et al*. present a utility-driven rationality and a complementary-driven rationality based model, relative to multiple partner coalitions. Morgado and Graca present a social reasoning mechanism that extends previous works [8]. In [7], a comparison analysis of a utility-driven method and a complementary-driven method is introduced. The proposed mechanism in this paper advocates the importance of transitive dependence and discusses the process of searching the potential formation partners.

6. Conclusions and Future Work

There are two major questions concerning coalition formation [11]: 1) how should a group of autonomous agents form a coalition? and 2) among all possible coalitions, what coalition will form, and what reasons and processes will lead the agents to form that particular coalition? The theory of direct dependence and transitive dependence tries to answer the first question, and it has been the direction followed by some previous works as presented in [4], [7], [15].

The main contributions of this paper include: 1) We identify, define and model a novel dependence relationship called transitive dependence relation. It extends the related research work on dependence based social reasoning by considering transitivity in dependence relations. Dependence relationships among agents are important for autonomous agents in MAS [3], [4], [7], [14], [15]. However, to date little work has been reported in defining the transitive relationships. Transitive dependence helps to agents' representing and reasoning social relations in MAS. Moreover, transitive dependency can help agents to reach a goal more fast with less cost since agents are goal oriented; 2) With the proposed transitive dependence theory, a novel coalition formation mechanism is worked out. Although there are some efforts using dependence relations in coalition formation, but transitive dependence has not been identified and utilized in the related work though it can bring great mutual benefit to agents in MAS. Moreover, transitive dependence based reasoning is proposed for the search of potential partners in coalition formation, which can answer the second question concerning coalition formation. Compared with other coalition formation methods, the transitive dependence based ones are more intuitive. With the anytime algorithm, the depending agent can find the best solution so far, which means that, at any time, there is a dynamic coalition can be formed; and 3) Nested coalition formation and how to deal with incomplete information in coalition formation are also discussed in this paper.

The proposed framework based on the proposed transitive dependence theory consists of information gathering, coalition formation reasoning and coalition formation. The framework is inspiring for dynamic coalition formation. Although of high complexity (which is also the problem of other proposed methods), with the anytime algorithm, the proposed coalition formation framework can be applied in many application domains (especially for applications with relatively small scale), such as, the application in autonomic and service oriented computing, dynamic web/Grid service composition, VOs (Virtual Organizations) formation in service oriented Grid where all the entities can be treated as autonomous agents, supply chain management, workflow, enterprise integration, etc. Our on-going research is focused on meeting the balance between optimal rationality and the large complexity of the transitive dependence based reasoning, which will be reported in the future papers.

Acknowledgments

The authors would like to thank the Associate Editor and two anonymous referees for their comments and suggestions for improving this paper. The authors also would like to thank Zhiqi Shen, Guido Boella, Leon van der Torre, Tom Mitchell, Lianggui Tang, Shuangqing Li, and Luigi Sauro for their helpful suggestions on this research.

References

[1] E. Alonso, "An individualistic approach to social action in multi-agent systems," J. Experimental and Theoretical Artificial Intelligence, vol.11, no.4, pp.519–530, 1999.

[2] G. Boella, L. Sauro, and L. van der Torre, "Social viewpoints on multiagent systems," Proc. 3rd International Joint Conference on Autonomous Agents and Multiagent Systems, pp.1358–1359, 2004.

[3] C. Castelfranchi, A. Cesta, and M. Miceli, "Dependence relations in multi-agent systems," in Decentralized AI-3, ed. Y. Demazeau and E. Werner, Elsevier, North-Holland, 1992.

[4] N. David, J.S. Sichman, and H. Coelho, "Extending social reasoning to cope with multiple partner coalitions," Proc. 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, vol.1647, pp.175–187, 1999.

[5] 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.

[6] M. Klusch and A. Gerber, "Dynamic coalition formation among rational agents," IEEE Intell. Syst., vol.17, no.3, pp.42–47, May/June 2002.

[7] M. Ito and J.S. Sichman, "Dependence based coalitions and contract net: A comparative analysis," Proc. International Joint Conference 7th Ibero-American Conference on Artificial Intelligence (IBERAMIA'00) and 15th Brazilian Symposium on Artificial Intelligence (SBIA'00), pp.106–115, 2000.

[8] L. Morgado and G. Graca, "A social reasoning mechanism based on a new approach for coalition formation," Proc. 2nd International Symposium "From Agent Theory to Agent Implementation," Vienna, Austria (EU), April 2000.

[9] T. Sandholm and V. Lesser, "Coalitions among computationally bounded agents," Artif. Intell., vol.94, no.1, pp.99–137, 1997.

[10] 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.

[11] O. Shehory and S. Kraus, "Coalition formation among autonomous agents: Strategies and complexity," in Reaction to Cognition, ed. C. Castelfranchi and J. Muller, pp.57–72, Springer Verlag, Heidelberg, 1995.

[12] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," Artif. Intell., vol.101, no.1-2, pp.165–200, 1998.

[13] O. Shehory and S. Kraus, "Feasible formation of stable coalitions among autonomous agents in non-superadditive environments," Comput. Intelligence, vol.15, pp.218–251, 1999.

[14] J.S. Sichman, R. Conte, Y. Demazeau, and C. Castelfranchi, "A social reasoning mechanism based on dependence networks," Proc. 16th European Conference on Artificial Intelligence, pp.188–192, 1994.

[15] J.S. Sichman, "DEPINT: Dependence-based coalition formation in an open multi-agent scenarios," J. Artificial Societies and Social Simulation, vol.1, no.2, 1998.

[16] E. Yu and J. Mylopoulos, "Understanding "Why" in software process modeling, analysis, and design." Proc. 16th IEEE International Conference on Software Engineering, pp.159–168, Sorrento, Italy,

California, 1994.
[17] G. Zlotkin and J.S. Rosenschein, "Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains," Proc. AAAI 94, pp.432–437, 1994.

Appendix: Proof of Theorem 1

Suppose that an agent ag_i has a goal g_i. There are k plans for the goal g_i and each plan consists of a set of actions. To find the optimal plan, the depending agent should get the least cost of each plan. To get the the least cost of each plan, the depending agent should compute the least cost of each action in the plan that the agent depends on other agents. To get the least cost of an action in a plan, the agent should get the least cost when it asks an agent to achieve the action. For instance, in Fig. 1, to decide which plan to use, the agent ag_1 should get the least cost of the plans p_{11} and p_{12}. To get the least cost of the plan p_{11}, the agent ag_1 should get the least cost of the actions a_1 and a_2. To get the least cost of the action a_1, the cost of asking the agents ag_2 and ag_3 to achieve the action a_1 should be calculated first. Obviously, to prove that the dependence based coalition formation problem is NP-complete, we just need to prove that the computation of the least cost of asking an agent to achieve an action in a plan (CLCAA) is NP-complete.

To get the least cost of asking an agent to achieve 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. We introduce a directed *search graph* to describe the process of computing the least cost of asking an agent to achieve an action. Suppose that there are n agents in MAS and each agent can achieve at most m actions, there are at most $n \times m$ valid nodes and one invalid node in the search graph. Each valid node, represented as (ag_i, a_j), represents that the agent ag_i can achieve the action a_j. If an agent ag_i has an action dependence relation on the action a_j about the action a_i, and an agent ag_j can achieve the action a_j, then there is an edge from the node (ag_i, a_i) to the node (ag_j, a_j). But if there is no agent can achieve the action a_j, then there is an edge from the node (ag_i, a_i) to the invalid node.

To get the least cost of asking an agent to achieve an action (represented by a node in the search graph), the depending agent should search all the feasible paths from the node and find the least cost of all the feasible paths. A feasible path here means that 1) there are no repeated nodes in the path, 2)the tail of the path isn't the invalid node, and 3) there is no edge from the tail of the path.

Membership of NP is easy: given an instance, simply guess a path is feasible in a search graph. The size of the path is bounded above 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 path in a search graph can be done in polynomial time.

For completeness, we must show that CLCAA 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 problem can be transformed into an instance of $\tau(I)$ of CLCAA 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 CLCAA, we define a reduction from a version of the wellknown 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 $v_i > 0$ 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 A· 1. The most valuable set of items in this example is $\{2, 3\}$. We transform the example into an instance $\tau(Knapsack - 01)$ of CLCAA. To do this, we first create all the nodes for the search graph. We create two nodes for each item i: i_0 and i_1. We also create a *Root* node and an invalid node. The edges between the nodes are defined as: 1) there is one edge from the *Root* node to the two nodes of item 1; 2) for each node of item i ($i = 1, 2, 3$), there is an edge from the node to the nodes $(i + 1)_0$ and $(i + 1)_1$ respectively. There is no edge connected with the invalid node.

The search graph transformed from the example Knapsack-01 is showed in Fig. A· 1. Then the depending agent begin to search all the feasible paths from the *Root*

Table A· 1 A 0-1 Knapsack problem instance.

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

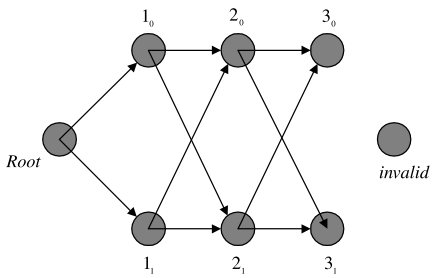

Fig. A· 1 Search graph for example Knapsack-01.

Let N be the number of items of different values and volumes

For each item i, create two nodes i_0 and i_1.
Create a *Root* node and an invalid node

Draw one edge from the *Root* node to the two nodes of item 1
For each node i_0 or i_1, $i < n$
Draw **one edge from the node to the nodes** $(i + 1)_0$ **and** $(i + 1)_1$ **respectively**

Fig. A· 2 Reducing the 0-1 Knapsack problem to CLCAA.

node, for example, $Root \rightarrow 1_0 \rightarrow 2_1 \rightarrow 3_0$. The node i_0 means that the item i isn't in the selection of items and the node i_1 means that the item i is in the selection of items. From all the feasible paths from the *Root* node, 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 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 CLCAA is entirely automatic (see Fig. A· 2), and is polynomial.

Bo An received a BSc degree in Computer Science from Chongqing University, China, in 2003. Currently he is a MSc student at the College of Computer Science, Chongqing University. His research interests are in autonomous agent and multi-agent systems, especially in coalition formation and automated negotiation.

Chunyan Miao received her PhD from School of Computer Engineering, Nanyang Technological University Singapore. She is currently an Assistant Professor in the same school. Her major research interest includes machine learning, intelligent software agent, agent mediated semantic web/grid, and agent oriented software engineering.

Daijie Cheng received a BSc degree in Mathematics from Beijing University of Aeronautics and Astronautics, China, in 1958. Currently he is a professor at the College of Computer Science, Chongqing University, China. His research focuses on parallel computing, networks, intelligent e-commerce, and data mining.