INTELLIGENT AUTOMATION & SOFT COMPUTING, 2017
http://dx.doi.org/10.1080/10798587.2016.1267238

A Complex Networked Method of Sorting Negotiation Demand Based on Answer Set Programs

Hui Wang, Liang Li, Long-yun Gao and Wu Chen

College of Computer and Information Science, Southwest University, Chongqing, China

ABSTRACT

With the development of big data science, handling intensive knowledge in the complex network becomes more and more important. Knowledge representation of multi-agent negotiation in the complex network plays an important role in big data science. As a modern approach to declarative programming, answer set programming is widely applied in representing the multi-agent negotiation knowledge in recent years. But almost all the relevant negotiation models are based on complete rational agents, which make the negotiation process complex and low efficient. Sorting negotiation demands is the most key step in creating an efficient negotiation model to improve the negotiation ability of agents. Traditional sorting method is not suitable for the negotiation in the complex network. In this paper, we propose a complex networked negotiation, which can show the relationships among demands, and then a sorting method of negotiation demands is proposed based on demand relationships. What's more, we use the betweenness of literals and the boundary co-efficient of rules to evaluate the importance of demands and rules.

KEYWORDS

Complex network; Big data; Multi-agent negotiation; Answer set program; Demand sorting

1. Introduction

With the development of big data science, lots of intensive knowledge in complex networks needs to be represented. Handling common sense in the complex network becomes more and more popular. Negotiation plays an important role in multi-agent system to solve conflict, realize coordination and cooperation. In real society, negotiation in the complex network is really common among multiple agents and it is a multi-round process. Agents will make some deals in each negotiation round, and the negotiation will not end until the final agreement is reached or negotiation fails. Negotiation is one of the issues, which has been studied for a long time, and various formalized methods have been proposed (Chen, Zhang, & Wu, 2009, 2013; Meyer, Foo, Kwok, & Zhang, 2004; Son, Pontelli, Nguyen, & Sakama, 2014), such as a sequential model for reasoning about bargaining in logic programs. In recent years, a formalized method based on belief revision has been proposed (Wu, Zhang, & Zhang, 2011; Zhang, Foo, Meyer, & Kwok, 2004). Its basic idea is that when two agents negotiate, both sides put forward some demands and use a formula set of logic language to represent these demands. Each agent makes decisions depending on, which demands are acceptable by guessing counterpart's demands. In some cases, each agent can give up part of its demands, which have been proposed at the beginning and accept some of counterpart's demands in order to make a deal. These demands are called beliefs of agent and they are the basis of the rational mechanisms of giving up and receiving demands. Results of each round of negotiations are the results of belief revision among agents. As an important method of knowledge representation, answer set program (*ASP* for short) plays a significant role in AI (Baral, 2003; Gelfond & Lifschitz, 1991). In recent years, some researchers began to apply *ASP* to multi-agent negotiation, which opened a new way for multi-agent theory. Answer set programming is a form of declarative programming oriented towards difficult (primarily NP-hard) search problems. It is based on the stable model (answer set) semantics of logic programming. All the agents in researches of negotiation based on *ASP* are rational agents, which have a common limitation of making the negotiation process inefficient. In their researches, they have not put the relations and importance among negotiation demands into consideration.

To solve the problems above, we put forward a method of demand sorting based on relationships of negotiation demands. Each agent is modeled as an *ASP*, and all the answer sets of the *ASP* are the whole demands of corresponding agent. In each round of negotiation, each agent selects one answer set from its own sets as current demand and then changes its program by accepting part of counterpart's literals or by giving up some of its own literals. Both sides get into next round of negotiation with modified programs and the negotiation will end with making a deal or failure. In order to solve the inequality of negotiation results, we propose a rational sorting method of demands. Based on graph theory (West, 2000) and the researches about vulnerability of complex networks (Albert & Barabasi, 2002; Boccaletti et al., 2007; Crucitti, Latora, & Marchior, 2004, Jiang, Wu, Xu, & Yuan, 2013; Marrone et al., 2013; Mishkovski, Biey, & Kocarev, 2011), we translate an *ASP* into a relation network of demands, and compute the degree of the importance of each negation demand by analyzing each node's position in the whole negotiation network, and predict

CONTACT Wu Chen chenwu@swu.edu.cn

the result of negotiation through initial demands. Complex network has many valuable applications in different fields such as economy, society and population, and classification of data mining is an important measurement to study problems in these fields. Meanwhile, community detection of the complex network is the same as classification and clustering in essence. Thus, to some extent, we provide with a new research way from Answer Set Program to data mining using the method we proposed in the paper.

2. Preliminaries

In this section, we recapitulate the basic concepts of *ASP* in order to describe our negotiation model. The notations we use follow the convention in the literatures (Gelfond & Lifschitz, 1991).

Assume that L is a propositional language with a finite number of propositional symbols (atoms). A literal can be either a positive atom, say *a*, or a negative atom, say¬*a*.*a* and ¬*a* are called complementary literals. A *rule* is a formula

$$L_0 \leftarrow L_1, \ldots, L_m, notL_{m+1}, \ldots, notL_n (0 \le m \le n)$$

where each $L_i (0 \le i \le n)$ is a literal, *not* is *negation as failure*. Its head, positive body and negative body is written as, $Head(r) = \{L_0\}$,$Pos(r) = \{L_1, \ldots, L_m\}$ and $Neg(r) = \{L_{m+1}, \ldots L_n\}$ respectively. *r* is called a *fact* if $Pos(r) = \varnothing$ and $Neg(r) = \varnothing$.*r* is a *constraint* if $Head(r) = \bot$. An *ASP* is a finite set of *rules*. Given an *ASP* $\prod$, we wrote $Head(\prod) = \bigcup_{r\in\prod} Head(r)$, $Pos(\prod) = \bigcup_{r\in\prod} Pos(r)$, $Neg(\prod) = \bigcup_{r\in\prod} Neg(r)$.

An *ASP* $\prod$ is called *a basic program* if $Neg(\prod) = \varnothing$. For *a basic program*, let *Lit* be the set of all literals in the underlying language of $\prod$. A set of literals is the *answer set of a basic program*$\prod$ if it is the smallest subset *S*of *Lit* such that:

(1) For any $r \in \prod$, if $Pos(r) \subseteq S$, then $Head(r) \subseteq S$;
(2) If *S* contains a pair of complementary literals, then $S = Lit$.

The answer set of a basic program $\prod$ is designated as $C_n(\prod)$.

Now let $\prod$ be *a logic program*. *Lit* again denotes the set of all literals in the language of $\prod$. For any set $S \subset Lit$, let $\prod^S$ be the *basic program* obtained from $\prod$ by removing

(1) Each $r \in \prod$ if $Neg(r) \cap S \neq \varnothing$;
(2) All formulas of the form *notL* in the remaining rules.

Which is called *Gelfond-Lifschitz reduction* (Gelfond & Lifschitz, 1991). *S* is an *answer set* of $\prod$ if and only if $S = C_n(\prod^S)$. *S* is a consistent answer set of an *ASP* $\prod$ if *S* is an answer set and consistent, i.e., the answer set does not contain complementary literals. *ASP* ($\prod$) is used to denote the set of all the consistent answer sets of an *ASP* $\prod$.

3. The Sorting Method of Demands

In the negotiation model among agents, the most important step is how to accept and give up demands. In this section, we propose a new sorting method according to the priority of demands.

3.1 How to Build the Sort over Demands

We regard all literals as negotiation demands and then divide them into two parts; basic demands and extended demands according to their importance. Basic demands are the bases of negotiation. Each agent has to satisfy counterpart's basic demands. Extended demands are the demands that need to be negotiated. The process of negotiation of extended demands is dynamic. In order to make a deal, agent can give up some of own extended demands and accept some of counterpart's. Here accepting is selective. Agent wants to gain maximum benefit through a good strategy. The division of basic and extended demands is as follows:

Basic demands: $BD = \{l_i | l_i \leftarrow \in \prod\}$
Extended demands: $ED = \{l_i | l_i \in L/BD\}$

L represents all the literals in an *ASP*. Basic demands have the highest priority. For extended demands, we choose the method introduced below to accept selectively.

In this paper, we realize the negotiation process by simplifying the rules in *ASP* and by adding and deleting nodes and edges. Basic demands will not be given up in the updating of program, so we mainly talk about how to sort extended demands in order to accept selectively. Before introducing four sorting methods, we introduce some knowledge of the graph.

A graph is an ordered pair $G = (V, E)$, where *V* represents the set of nodes and *E* represents the set of edges. In negotiation, *V* stands for the demands (literals) of negotiation and *E* represents the relation of inference among literals. $V = \{v_1, v_2, v_3, \ldots, v_n\}$(Where n is the number of nodes), $E = \{e_1, e_2, e_3, \ldots, e_m\}$(where m is number of edges), $e = (v_i, v_j)$ $(1 \le i \le n, 1 \le j \le n)$.

These methods are based on the relation of inference in negotiation, so we need to create a relation network based on literals in *ASP* that we call negotiation network in this paper. Here are some examples to introduce how to create negotiation network and further analyze the feature of negotiation network. We first introduce the method of representation of facts through Example 1:

Example 1. Consider the following *ASP* $\prod_1$:

$$a \leftarrow$$
$$b \leftarrow .$$

$\prod_1$ is represented as Figure 1.

We take the following Example 2 to illustrate that an *ASP* $\prod_2$ contains positive literal only:

Example 2.
$\prod_2$:

$$a \leftarrow$$
$$b \leftarrow$$
$$c \leftarrow b$$
$$d \leftarrow c$$
$$f \leftarrow e.$$

$\prod_2$ is captured as Figure 2.

We introduce *not* into negotiation model as follows:

Example 3. Consider the following *ASP* $\prod_3$:

$$a \leftarrow$$
$$b \leftarrow$$
$$c \leftarrow b$$
$$d \leftarrow not\ c$$
$$f \leftarrow not\ e.$$

$\prod_3$ is represented as Figure 3.

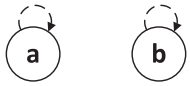


Figure 1. The Representation of Facts.

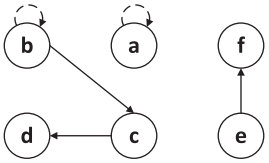


Figure 2. The Representation of an ASP Containing Positive Literal Only.

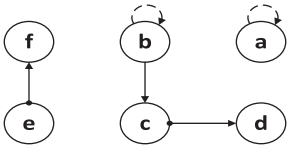


Figure 3. The Representation of ASP Containing Not. Not is Represented as a Circle with a Black Dot.

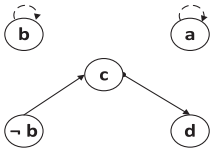


Figure 4. The Representation of an ASP Containing Classical Negation Symbol¬.

We introduce classical negation symbol ¬ in an *ASP* $\prod_4$:

Example 4.

$\prod_4$:

$$a \leftarrow$$
$$b \leftarrow$$
$$c \leftarrow \neg b$$
$$d \leftarrow not\ c.$$

$\prod_4$ is represented as Figure 4.

Now we introduce the representation of a more complex *ASP* $\prod_5$ in Example 5.

Example 5.

$\prod_5$:

$$a \leftarrow$$
$$b \leftarrow$$
$$c \leftarrow \neg b$$
$$d \leftarrow b, not\ c$$
$$e \leftarrow d, c, not\ a.$$
$$d \leftarrow not\ f.$$
$$g \leftarrow not\ e.$$

$\prod_5$ is represented as Figure 5.

In Figure 5, for any rule *r*, a red circle stands for one process. Each process connects two literals in Body(r), or a literal inBody(r) with another previous process of *r*. And further it will point to the head literal of *r*. We design an algorithm to create such graph to represent the model of negotiation network as follows:

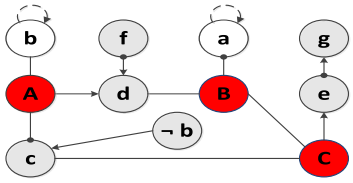


Figure 5. The Representation of a Complex ASP Containing Not and ¬.

Input: rules set $R, r \in R$
Output: G(V, E) \\ Vis set of nodes, *E* is set of edges
$V = \emptyset, E = \emptyset$ \\ Initialization
For each R **do**
  Nodesize = R.Bodysize; nodes = R.Body.literal;
  for i = 0 to nodesize-2 **do**
    $Vp(n) = CreateVp(node[i], node[i+1])$;
    $V.add(Vp(n))$; // add a virtual process.
    $E.add(node[i] \rightarrow Vp(n))$; \\ n is number of process
    $E.add(Vp(n) \rightarrow node[i+1])$;
    $E.add(node[i+1] \rightarrow Vp(n))$;
    $E.add(Vp(n) \rightarrow node[i])$;
    $nodes.remove(node[i])$; $nodes.remove(node[i+1])$;
    $nodes.add(0, Vp(n))$; \\ put process in the first place to continue calculate
  end
  $E.add(Vp(n) \rightarrow R.Head)$; \\ add links to *E*
  $V.addall(R.Body.literal)$;$V.add(R.Head.literal)$; \\ add literals into *V*
end

Algorithm 1: Generalized Graph Creation Algorithm Based On ASP

According to Algorithm 1, we can easily create a graph based on *ASP*. For instance, in Example 5, we can obtain rules set from *ASP* Π_5, and for each $r \in R$, the first element is $a \leftarrow$, nodesize = 0 and $r.Body.literal = \emptyset$, $r.Head.literal = \{a\}$, so at the end of the first round we can get a graph $G_{r1} = (V_{r1}, E_{r1})$, $V_{r1} = \{a\}$, $E_{r1} = \emptyset$. For rule $d \leftarrow b, notc$, its nodesize = 2 and we can also find $r.Body.literal = \{b, notc\}$, $r.Head.literal = \{d\}$, it needs to enter the second loop, create a virtual process $Vp(1)$, and then add link between $Vp(1)$ and $node[0]$, and link between $Vp(1)$ and $node[1]$, $Vp(1)$isn't a node, but we treat them as a transitional state while calculating, then, we get graph $G_{r4} = (V_{r4}, E_{r4})$, $V_{r4} = \{b, notc, d\} E_{r4} = \{b \rightarrow Vp(1), not\ c \rightarrow Vp(1), Vp(1) \rightarrow d\}$. Finally, the result of $\Pi_5$is $G = G_{r1} \cup G_{r2} \cup G_{r3} \cup G_{r4} \cup G_{r5} \cup G_{r6} \cup G_{r7} = (V, E)$, where $V = \{a, b, c, d, e, g, \neg b, not\ f\}, E = \{c \rightarrow Vp(1), Vp(1) \rightarrow b, Vp(1) \rightarrow d, f \rightarrow d, \neg b \rightarrow c, d \rightarrow Vp(2), a \rightarrow Vp(2), Vp(2) \rightarrow Vp(3), Vp(3) \rightarrow e, e \rightarrow g\}$.

In the following section, we introduce four sorting methods. Two are based on the importance of nodes in negotiation network, and the other two are about the importance of edges.

(1) Sort demands by degree of node.

Each literal has some relations with others in negotiation model. The degree of node reflects the degree of relationship of literals in the whole negotiation. Firstly, sort all literals by their degree. The larger the degree of literal is, the more important the demand is. Agent chooses the demand whose degree of node is the largest to negotiate with its counterpart, and gives up demand whose degree of node is small enough.

(2) Sort rules by weight of edge.

Agent will delete rules and literals in negotiation model, that is to say, edges and nodes will be deleted in negotiation network. When choose to delete edges, we sort them by their importance in advance. When choose to give up rules, agent gives up rules with smaller weight. The computing method of weight of rules is as follows:

For rules whose body contain more literals, the computing method is as follows:

$$we(e) = \sqrt{k_1 \bullet \frac{k_2 + k_3 + \ldots + k_n}{n-1}} \tag{1}$$

where k_1 is the degree of the head literal, $k_2 + k_3 + \ldots + k_n$ represents the degrees of all nodes in the body of the rule, and n stands for number of literals contained in the body of the rule.

For example, in rule$\{c \leftarrow d, e, not\ f\}$, k_d,k_e and k_f represent degrees of node of demands d,e and f respectively. The degree of the rule is$\sqrt{k_c \bullet \frac{k_d+k_e+k_f}{3}}$.

When choosing rules, it needs to work out the weight of each rule first. Rules with larger weight play a more important role in network. Agent chooses rules with larger degree to negotiate and gives up rules whose degrees are smaller in order to make maximum benefit.

(3) Sort demands by node betweenness (Mishkovski et al., 2011)

Betweenness of node reflects the tightness of the node with other nodes in negotiation network. The larger betweenness the node is, the node is more important in negotiation network, and the influence after deleting it is bigger. The computing method of betweenness of demand is as follows:

$$D(v) = \sum_{i \neq j} \frac{n_{ij}(v)}{n_{ij}} \tag{2}$$

where $D(v)$ represents the betweenness of node (demand or literal), n_{ij} is the number of shortest paths between node I and node j . $n_{ij}(v)$ stands for number of shortest paths between node i and node j including node v. Agent chooses demands with larger betweenness to negotiate and give up demand with smaller betweenness in order to make a deal.

(4) Sort rules by boundary coefficient b_l is the boundary coefficient of edge l:

$$b_l = \sum_{i \neq j} \frac{n_{ij}(l)}{n_{ij}} \tag{3}$$

Where n_{ij} represents the number of shortest paths between node I and node j,$n_{ij}(l)$ stands for number of shortest paths between node I and node j including edge l. The larger the boundary coefficient is, the more important the rule is in negotiation network.

The definition of average boundary coefficient is as follows:

$$b(G) = \frac{1}{|E|} \sum_{l \in E} b_l \tag{4}$$

where $|E|$ is the number of edges.

$$b(G) = \frac{1}{|E|} \sum_{l \in E} \sum_{i,j \in V} \frac{n_{ij}(l)}{n_{ij}} = \frac{1}{|E|} \sum_{l \in E} \frac{1}{n_{ij}} \left(\sum_{i,j \in V} n_{ij}(l) \right) \tag{5}$$

Let S_{ij} be the set of all shortest paths between node i and node j:

$$n_{ij}(l) = \sum_{g \in S_{ij}} C_g(l) \tag{6}$$

where

$$C_g(l) = \begin{cases} 1 & l \in g; \\ 0 & \text{otherwise} \end{cases}$$

We can make the following operation (Albert & Barabasi, 2002; Marrone, Nardone, & Tedesco, 2013):

$$b(G) = \frac{1}{|E|} \sum_{l \in E} \frac{1}{n_{ij}} \left(\sum_{i,j \in V} n_{ij}(l) \right) = \frac{1}{|E|} \sum_{l \in E} \frac{1}{n_{ij}} \left(\sum_{g \in S_{ij}} \sum_{l \in E} C_g(l) \right) \tag{7}$$

And then,

$$b(G) = \frac{1}{|E|} \sum_{l \in E} \frac{1}{n_{ij}} \left(\sum_{g \in S_{ij}} d_{i,j} \right) = \frac{n(n-1)}{2|E|} L(G) \tag{8}$$

Where $L(G)$ represents the characteristic path length of network, $d_{i,j}$ is the degree between node i and node j. See from the formula, we can know that use $b(G)$ to measure the vulnerability of network can reflect the nature of negotiation network better than using characteristic path length.

Considering some limitations, the method of measuring vulnerability of negotiation network from multiple dimensions is as follows (Boccaletti et al., 2007; Mishkovski et al., 2011):

$$b_p(G) = \left[\frac{1}{|E|} \sum_{l \in E} \left(\sqrt{k_1 \bullet k_2} b_l \right)^p \right]^{1/|p|} \tag{9}$$

where k_1, k_2 are the degrees of nodes connected with edge l, $p > 0$. In order to compare these two negotiation networks G_1 and G_2, give p the value of 1 and calculate the value of $b_1(G_1)$. If $b_1(G_1) < b_1(G_2)$,G_1 is more robust than G_2; if $b_1(G_1) > b_1(G_2)$,G_2 is more robust than G_1; if $b_1(G_1) = b_1(G_2)$, then choose $p > 1$ and work out $b_p(G)$ until $b_p(G_1) \neq b_p(G_2)$.

We find some nodes have really large degrees from experiments, but these nodes are at the outskirts of network so they are not important. And the method based on betweenness can solve this problem efficiently. Also, we find some edges with large weight are at the outskirts of network and some edges with small weight are at the center of network. And the method based on boundary coefficient can solve this problem. So we use the method based on betweenness to determine the importance of nodes (literals) and use the method based on boundary coefficient to determine the importance of edges (rules).

Consider the following *ASP* $\prod$:

1:$a \leftarrow .$

2:$b \leftarrow .$

3:$c \leftarrow \neg b.$

4:$d \leftarrow not\ c.$

5:$f \leftarrow not\ e.$

The answer sets = $\{a, b, d, f\}$. $D(c) = 3$, $D(d) = 2$, $D(e) = 1$, $D(f) = 1$. The relation of the importance of demands is $\{a = b \succ c \succ d \succ f \succ e\}$. Here the demands user wants are $\{a, b, c, f\}$ and the demand proposed by user are $\{a, b, f\}$. So the sort of demand is $\{a = b \succ f\}$.

The boundary coefficients of rule 3, 4 and *5* are; b(3) = 2, b(4) = 2, b(5) = 1 and the relation of importance of rules is {1 = 2 ≻ 3 = 4 ≻ 5} according to the boundary coefficient of edges. In the process of negotiation, agent will give up demand *f* (rule *5*) if necessary in order to maximize benefit.

3.2. Analyzing Negotiation Network

We can compute the importance of demands by the sorting methods above and we introduce how to accept and give up demands using these methods in the following part.

In complete graph, every two nodes are connected by an edge. There are N(N-1) edges in a complete graph with N nodes. So the vulnerability of complete graph is as follows (Mishkovski et al., 2011):

$$b_p(G_{complete}) = 1 \tag{10}$$

Where $G_{complete}$ stands for complete graph G.

Path diagram is a kind of simplest graph. It contains nodes whose degrees are 1 or 2 only, and only two nodes' degrees are 1. There are N-1 edges in a path diagram with N nodes. So the vulnerability of path diagram is as follows:

$$b_p(G_{path}) = \frac{N(N+1)}{6} \tag{11}$$

where G_{path} stands for path diagram G.

If the number of nodes is no less than 2 in network, we have the relation below;

$$b_p(G_{complete}) < b_p(G) < b_p(G_{path}) \tag{12}$$

In order to measure the vulnerability of negotiation network more clearly, we regard the normalized average boundary to evaluate the vulnerability.

$$b_{nor}(G) = \frac{b(G) - b(G_{complete})}{b(G_{path}) - b(G_{complete})} = \frac{b(G) - 1}{\frac{N(N+1)}{6} - 1} \tag{13}$$

where N represents the number of nodes in negotiation network. So we can find that $0 \le b_{nor}(G) \le 1$. When $b_{nor}(G)$ is close to 1, the negotiation network G is more vulnerable, similarly, G is more robust when $b_{nor}(G)$ approaches 0.

Assume that G' is the graph we get after sorting G by the degrees of node and adding or deleting some nodes, we can add some important nodes and delete some unimportant nodes:

$$KI_{nor}(G) = \frac{b_{nor}(G') - b_{nor}(G)}{b_{nor}(G)} \tag{14}$$

Assume that G'' is the graph we get after sorting G by the degrees of node and adding or deleting some edges, we can add some important edges and delete some unimportant edges:

$$KI_{edge}(G) = \frac{b_{nor}(G'') - b_{nor}(G)}{b_{nor}(G)} \tag{15}$$

The larger the values of $KI_{nor}(G)$, $KI_{edge}(G)$ are, the worse the vulnerability of negotiation network after giving up this literal is.

When receiving demands, the counterpart might have several schemes. Agent adds these demands or rules into own program, compute $KI_{nor}(G)$ or $KI_{edge}(G)$, and choose the scheme, which has the smallest $KI_{nor}(G)$ or $KI_{edge}(G)$.

4. Related Work

We have studied some papers based on *ASP* and negotiation, and found that some researchers do consider the importance of negotiation demands, but they have not proposed the computing method of priority of demands. Still, a very few researchers have proposed the computing method. Here is the comparison of methods with this paper.

Zhang and *Foo*, *Brewka* and *Eiter* have done detailed work about the priority of rules in *ASP* respectively, and proposed mature theories (Brewka & Eiter, 1999; Zhang & Foo, 1997). In their work, there is a partial order between any two rules in *ASP.* The method of determining the priority of rules based on preference is qualitative. In this paper, we compute the priority based on the relation of rules in *ASP,* which has been already established. The method is quantitative and it can be used to compute the priority of combinations of rules. In the researches of *Zhang* and *Foo* (Zhang & Foo, 1997), the priority of rules remains unchangeable after it has been given, but the priority in our work is dynamic. With the negotiation approaches, the relation of inference among rules changes dynamically, thus the priority changes accordingly, which is more suitable in the real situation.

The method proposed by *Zhang* and *Foo*, *Brewka* and *Eiter* can solve many issues in negotiation, but it behaves insufficiently in following instance:

$$1{:}a \leftarrow .$$
$$2{:}b \leftarrow .$$
$$3{:}c \leftarrow a.$$
$$4{:}d \leftarrow b.$$
$$5{:}f \leftarrow d.$$

We can find that rule 4 is more important than rule 3. However, rule 3 is as important as rule 4 using the method of theirs, which has some difference from a real situation. In our work, we get that rule 4 is more important than 3 by the method of a boundary coefficient.

Above all, the sort methods based on boundary coefficient and betweenness can compute priority quantitatively and solve the priority among combinations. And the priority is dynamic during negotiation, which is more close to real situation. Also, the method can be used to predict the result of negotiation.

5. Conclusion and Future Work

In this paper, each literal is considered as a demand and demands can be divided into two parts; the basic demands and the extended demands. Negotiation mainly solves the issues of accepting and giving up demands. We regarded *ASP* as a negotiation relation of agent based on answer set programming and use answer set to represent negotiation demand. We translated *ASP* into negotiation network. In particular, we proposed sort methods of the importance of rules or literals based on the degree of node, the weight of edge and betweenness and we proposed the method of measuring the influence after accepting and giving up rules or literals. Also, we proposed the method of predicting the results of negotiation. The sort method of negotiation demands based on the relation of nodes solved the problem of unbalanced negotiation. Future work is to use these sort methods in accepting and giving up demands and bring research of the priority of *ASP* into multi-agent negotiation. Meanwhile, sorting itself defined in the rules of logic program is important and worthy to study and doing some research about classification

and clustering methods in data mining combining this complex networked method will be interesting and meaningful.

Disclosure statement

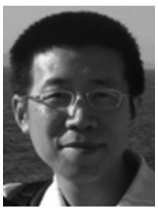
No potential conflict of interest was reported by the authors.

Notes on contributors

Hui Wang received a BS degree from Southwest University, China, in 2015. Currently, he is studying in College of Computer and Information Science of Southwest University in China as a postgraduate student. His main research interests include logic programming, automated negotiation, knowledge representation and reasoning.

Liang Li received a BS degree from Southwest University, China, in 2015. Currently, he is working as a software engineer in H3C Technologies Co. Limited.

Long-yun Gao received a BS degree from Southwest University, China, in 2014. Currently, he is studying in College of Computer and Information Science of Southwest University in China as a postgraduate student. His main research interests include bargaining, logic programming, knowledge representation and reasoning.



Wu Chen received BS and MS degrees from Southwest Normal University in 1998 and 2004 respectively and a PhD from Guizhou University in 2009. Now he is a professor with the College of Computer and Information Science, Southwest University, China. His main research interests include bargaining, logic programming, automated negotiation, knowledge representation and reasoning.

Acknowledgement

We should thank the anonymous referees for their comments. This work was supported in part by the Major Project of National Social Science of China (14ZDB016).

References

Albert, R., & Barabasi, A.-L. (2002). Statistical Mechanics of complex networks. *Reviews of Modern Physics, 74*, 47–97.

Baral, C. (2003). *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge: Cambridge University Press.

Boccaletti, S., Buldu, J., & Herrero, R. (2007). Multiscale vulnerability of complex networks. *Chaos, 17*, 043110.

Brewka, G., & Eiter, T. (1999). Preferred answer sets for extended logic programs. *Artificial Intelligence, 109*, 297–356.

Chen, W., Zhang, M., & Wu, M. N. N. (2009). A Logic-program-based Negotiation Mechanism. *Journal of Computer Science and Technology, 24*, 753–760.

Chen, W., Zhang, D., & Wu, M. (2013). *A Sequential Model for Reasoning about Bargaining in Logic Programs*. Proceedings of the 12th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR2013). Springer, (pp. 239-244).

Crucitti, P., Latora, V., & Marchior, M. (2004). Model for cascading failures in complex networks. *Physics Review E, 69.045104*.

Gelfond, M., & Lifschitz, V. (1991). Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing, 9*, 365–386.

Jiang, C., Wu, L., Xu, F., & Yuan, J. (2013). Characteristics and Reliability Analysis of the Complex Network in Guangzhou Rail Transit. *Intelligent Automation and Soft Computing, 19*, 217–225.

Marrone, S., Nardone, R., & Tedesco, A. (2013). Vulnerability Modeling and Analysis for Critical Infrastructure Protection Applications. *International Journal of Critical Infrastructure Protection, 6*, 217–227.

Meyer, T., Foo, N., Kwok, R., & Zhang, D. (2004). *Logical foundations of negotiation strategies and preferences*. In Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning(KR'04) AAAI Press 2004, (pp. 311–318).

Mishkovski, I., Biey, M., & Kocarev, L. (2011). Vulnerability of Complex Networks. *Communications in Nonlinear Science and Numerical Simulation*, *16*, 341–349.

Son, T. C., Pontelli, E., Nguyen, N.-h., & Sakama, C. (2014). Formalizing Negotiations Using Logic Programming. *ACM Transactions on Computational Logic*, *15*(2), 1–30.

West, D. B. (2000). *Introduction to Graph Theory*. New Jersey: Prentice Hall.

Wu, M., Zhang, D., & Zhang, M. (2011). *Language Splitting and Relevance-based Change in Horn Logic*. In the Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI-11), AAAI Press 2011.

Zhang, Y. , & Foo, N. (1997). *Answer Sets for Prioritized Logic Programs*. In Proceedings of International Logic Programming Symposium.

Zhang, D., Foo, N., Meyer, T., & Kwok, R. (2004). *Negotiation as mutual belief revision*. Proceedings of the 19th National Conference on Artificial Intelligence (AAAI04), AAAI Press / The MIT Press 2004, (pp. 317–323).