2012 IEEE 24th International Conference on Tools with Artificial Intelligence

Solving the Coalition Structure Generation Problem with MaxSAT

Xiaojuan Liao
Graduate School of Information
Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819–0395
Email: liao@ar.inf.kyushu-u.ac.jp

Miyuki Koshimura
Faculty of Information
Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819–0395
Email: koshi@inf.kyushu-u.ac.jp

Hiroshi Fujita
Faculty of Information
Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819–0395
Email: fujita@inf.kyushu-u.ac.jp

Ryuzo Hasegawa
Faculty of Information
Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819–0395
Email: hasegawa@inf.kyushu-u.ac.jp

***Abstract*—Coalition Structure Generation (CSG), a main research issue in the domain of coalition games, involves partitioning agents into exhaustive and disjoint coalitions so that the social welfare is optimized. The advent of compact representation schemes, such as marginal contribution networks (MC-nets), promotes the efficiency of solving the CSG problem.**

In this paper, inspired by the dramatic speed-up of Boolean Satisfiability Problem (SAT) solvers, we make the first step towards a study of applying MaxSAT solvers to the CSG problem. We set out to encode the MC-nets into propositional Boolean logic and utilize an off-the-shelf MaxSAT solver as an optimization tool for solving the CSG problem. Specifically, based on the previous works, we encode rule relations and their constraints into weighted partial MaxSAT formulas and show that MaxSAT solvers are useful in solving the CSG problem. Furthermore, we put forward a brand-new method based on agent relations which specify whether two agents of a rule are in the same coalition. Experimental evaluations show that our methods outperform other state-of-the-art algorithms.

I. Introduction

Coalitional games have been proved highly influential in the research of multi-agent systems as they capture opportunities for cooperation by explicitly modeling the ability of the agent to take joint actions as primitives [1]. Coalition Structure Generation (CSG) is one of the main research issues in the use of coalitional games in multi-agent systems [2]. The CSG problem involves partitioning a set of agents so that the social welfare is maximized. The challenge of this problem is that the number of possible solutions grows exponentially as the number of agents increases.

Besides solving the CSG problem, designing compact representation schemes of characteristic functions is also an active research area. Traditionally, to represent characteristic functions, the naive solution is to enumerate the payoffs to each set of agents, thus requiring space exponential in the number of agents in the game [3]. This is prohibitive when the number of agents is large. Recently, more compact representation schemes were put forward [3], [4], [5], [6]. Ohta et al. [7] utilized these compact representation schemes to solve the CSG problem more efficiently. Then, Ueda et al. [8] further extended the work in [7] to handle negative value rules as well as externalities among coalitions.

In the last decade we have witnessed a dramatic speed-up of SAT solvers: problems with thousands of variables are now solved in milliseconds by state-of-the-art SAT solvers. MaxSAT is an optimization version of SAT. An interesting question is whether MaxSAT solvers can contribute to solving the CSG problem with higher efficiency. We try to answer this question by presenting and evaluating two methods for the CSG problem based on a reduction to propositional logic. Specifically, we make the following contributions:

- We encode the rule relation-based approach in [7], [8] into propositional logic. Our method can solve the CSG problem faster than previous works, i.e., problem instances with 150 agents/ rules can be solved within 14 seconds on average.
- We set out to design a new method by encoding agent relations into propositional logic. By encoding agent relations, the MaxSAT solver can solve problem instances with 150 agents/ rules within 11 seconds on average.

II. Preliminary

A. *Coalition Structure Generation (CSG)*

Coalition Structure Generation (CSG) refers to partitioning agents into exhaustive and disjoint coalitions so that the social welfare is optimized. Given a finite set of agents A, let 2^A denote the set of all subsets of A, then C is a *coalition* if

$C \in 2^A$. The value of a coalition C is given by a *characteristic function* v. A characteristic function $v : 2^A \to \mathbb{R}$ assigns a real-valued payoff to each coalition $C \subseteq A$.

A *coalition structure* CS is an exhaustive set of mutually disjoint coalitions over A, i.e., CS satisfies the following conditions: $\forall i, j\,(i \neq j)\,, C_i \cap C_j = \emptyset,\ \bigcup\limits_{C_i \in CS} C_i = A$. The value of a coalition structure CS is called *social welfare*, denoted as $V(CS)$, given by $V(CS) = \sum\limits_{C_i \in CS} v(C_i)$. We denote by $\Pi(A)$ the set of all coalition structures over A. The objective of coalition structure generation is to find an optimal coalition structure, so that the social welfare is maximized, i.e., given A, find CS^* such that $\forall CS \in \Pi(A)\,, V(CS^*) \geq V(CS)\,.$

B. MC-nets

Ieong et al. [3] developed a concise representation using sets of rules, called *marginal contribution network (MC-net)*, which largely reduces the space necessary for representation.

Definition 1. *(MC-nets). An MC-net consists of a set of rules R. Each rule $r_i \in R$ is of the syntactic form $I_i \to w_i$, where $w_i \in \mathbb{R}$ and I_i is a conjunction of literals over A, formally expressed by $\{a_1 \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \neg a_{m+2} \wedge \cdots \wedge \neg a_{m+n}\}$. For each rule r_i, we call P_i positive literals where $P_i = \{a_j\}_{j=1}^m$, and N_i negative literals where $N_i = \{a_{m+k}\}_{k=1}^n$. A rule r_i is said to apply to a coalition C if $P_i \subseteq C$ and $N_i \cap C = \emptyset$, i.e., all agents in P_i are in C, and none of agents in N_i are in C. For a coalition C, $v(C) = \sum\limits_{r_i \in R'} w_i$, where R' is the set of rules that apply to C. Thus, the value of a coalition structure CS is given as $\sum\limits_{C \in CS} \sum\limits_{r_i \in R'} w_i$. Without loss of generality, we assume each rule has at least one positive literal.*

Hence, the problem of finding CS^* is equivalent to finding a set of rules that apply to some coalition $C \in CS^*$, so that the sum of values in the set of rules is maximized.

Example 1. For $A = \{a_1, a_2, a_3, a_4\}$, assume there are four rules: $r_1 : a_1 \wedge a_2 \to 2$, $r_2 : a_1 \wedge a_2 \wedge \neg a_4 \to -2$, $r_3 : a_1 \wedge a_4 \to 1$, $r_4 : a_3 \wedge \neg a_2 \to 3$. In this case, $P_1 = \{a_1, a_2\}$, $N_1 = \emptyset$, $P_2 = \{a_1, a_2\}$, $N_2 = \{a_4\}$, $P_3 = \{a_1, a_4\}$, $N_3 = \emptyset$, $P_4 = \{a_3\}$, $N_4 = \{a_2\}$. r_1 and r_3 apply to coalition $\{a_1, a_2, a_4\}$, but r_2 and r_4 do not. Thus, $v(\{a_1, a_2, a_4\}) = 2 + 1 = 3$. After enumerating all possible coalition structures, we can calculate the optimized social welfare: $V(CS^*) = V(\{a_1, a_2, a_4\}\{a_3\}) = 2 + 1 + 3 = 6$.

C. Weighted Partial MaxSAT (WPM)

Boolean Satisfiability Problem (SAT) was the first problem shown to be NP-Complete [9], which is used to solve Boolean formulas that only contain logic operations *and*, *or* and *not*, called *propositional Boolean formula*. Typically, a propositional Boolean formula is expressed in *Conjunctive Normal Form (CNF)*, which consists of a conjunction (logic *and*) of one or more clauses. A *clause* is a disjunction (logic *or*) of one or more literals, and a *literal* is an occurrence of a Boolean variable or its negation. In this paper, we identify a set of clauses with a conjunction of clauses.

SAT determines whether there exists a variable assignment that makes a propositional Boolean formula evaluate true. A Boolean MaxSAT problem is an optimization problem related to SAT. Also, in practice, the problem instance is typically expressed as a CNF.

Solving *MaxSAT* problem amounts to finding a variable assignment that maximizes the number of satisfied clauses [10]. In *partial MaxSAT* instance, some clauses are *soft* and the rest are *hard*. The problem is to find an assignment that satisfies all the hard clauses and the maximum number of soft clauses. *Weighted MaxSAT* is a MaxSAT problem where each clause has a bounded positive numerical *weight*. The problem is to find an assignment that maximizes the sum of weights of satisfied clauses. *Weighted Partial MaxSAT (WPM)* is the combination of partial MaxSAT and weighted MaxSAT. In WPM, each soft clause is given a positive numerical value. Solving WPM corresponds to finding an assignment that satisfies all the hard clauses and maximizes the sum of weights of satisfied soft clauses.

Formally, we denote a WPM formula by $\varphi = \{(C_1, w_1), \ldots, (C_m, w_m), C_{m+1}, \ldots, C_{m+m'}\}$, where the first m clauses are soft and the rest are hard. With each soft clause C_i, a Boolean variable b_i is associated such that $b_i = 1$ if clause C_i is satisfied and $b_i = 0$, otherwise. Solving WPM instance φ amounts to finding an assignment that satisfies all $C_{m+1}, \ldots, C_{m+m'}$ and maximizes $\sum_{i=1}^m w_i b_i$.

III. Revisiting Existing Works

In an MC-net, a coalition game is represented by a set of rules. Each rule r_i is of the form $I_i \to w_i$. If w_i is positive, we call the rule *positive value rule*. If w_i is negative, we call it *negative value rule*. This section exhibits the existing works on handling positive and negative value rules in MC-nets.

Definition 2. *[7]. (Feasible rule set). A set of rules $R' \subseteq R$ is feasible if there exists CS where each rule $r \in R'$ applies to some $C \in CS$.*

Thereby, the problem of finding CS^* is equivalent to finding a feasible rule set R', so that $\sum\limits_{r_i \in R'} w_i$ is maximized.

Definition 3. *[7]. The possible relations between two rules r_i and r_j can be classified into the following four non-overlapping and exhaustive cases:*

(1) *Compatible on the same coalition: $P_i \cap P_j \neq \emptyset$ and $P_i \cap N_j = P_j \cap N_i = \emptyset$.*
(2) *Compatible on different coalitions: $P_i \cap P_j = \emptyset$ and ($P_i \cap N_j \neq \emptyset$ or $P_j \cap N_i \neq \emptyset$).*
(3) *Incompatible: $P_i \cap P_j \neq \emptyset$ and ($P_i \cap N_j \neq \emptyset$ or $P_j \cap N_i \neq \emptyset$).*
(4) *Independent: $P_i \cap P_j = \emptyset$ and $P_i \cap N_j = P_j \cap N_i = \emptyset$.*

Based on rule relations, an MC-net thereby can be considered as a graph, in which each vertex is a rule, and between any two vertices, there exists an edge whose type is one of the four cases described above. Ohta et al. [7] presented the following constraints and proved they are enough for finding out a feasible rule set among positive value rules.

Theorem 1. *[7]. A set of rules R' is feasible if and only if R' satisfies the following conditions.*

(1) *R' includes no pair of rules/vertices connected by an "incompatible" edge, and*
(2) *if two rules in R' have the relation of "compatible on different coalitions", then they are not reachable via "compatible on the same coalition" edges in R' .*

When all rules have positive values, making more rules apply to $C \in CS$ never decreases the social welfare, thus it is enough to just investigate the conditions in which the rules cannot be selected at the same time. The solver in [7] thereby only selects as many rules as possible and would simply ignore rules with negative values because they decrease the social welfare. However, when a negative value rule involved, constraints are required to specify the conditions in which the negative value rule should be selected because in some cases, in order to choose more positive value rules, the solver is forced to select some negative value rules first, although intuitively, choosing those negative value rules would decrease the social welfare.

Ueda et al. [8] demonstrated the advantage of using negative value rules in terms of significantly decreasing the number of rules required for representing a coalition game, and proposed a method called *direct encoding* to handle negative value rules. They created several dummy rules with the value of zero for each negative value rule. A negative value rule is incompatible with all of its dummy rules. Furthermore, they added a constraint to complement Theorem 1 by specifying the condition in which negative value rules must be selected, shown in Theorem 2. Readers may refer to the proof in [8].

Theorem 2. *[8]. A negative value rule applies to a coalition in coalition structure CS if and only if none of its dummy rules apply to any coalition in CS.*

IV. Rule Relation-based WPM

In this section, we set out to apply rule relations and the technique of direct encoding to WPM. First, we consider a special case that a rule contains only one agent. In this case, the rule is always selected because no matter how a CS is structured, a rule with a single agent always applies to a coalition in CS. Without loss of generality, in the rest of the paper, we assume each rule contains at least two agents.

A. Handling Positive Value Rules by Rule Relations

Let $R = \{r_1, r_2, \ldots, r_n\}$ be a set of positive value rules. For each rule r_i, we introduce a new Boolean variable B_i ($i = 1, 2, \ldots, n$). We use $\mathcal{B}$ as the set of all such Boolean variables, i.e., $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$. Intuitively, $B_i = 1$ means r_i is in a feasible set of rules.

In order to deal with reachability mentioned in Theorem 1, we introduce a Boolean variable $S(i,j)$ for a pair of rules r_i and r_j where $1 \leq i < j \leq n$. Intuitively, $S(i,j) = 1$ means both r_i and r_j are in a feasible rule set, and the relation between them is "compatible on the same coalition". We need the following hard clauses which represent transitive laws to accomplish the reachability where $1 \leq i < j < k \leq n$: $\neg S(i,j) \vee \neg S(j,k) \vee S(i,k)$, $\neg S(i,j) \vee \neg S(i,k) \vee S(j,k)$, $\neg S(i,k) \vee \neg S(j,k) \vee S(i,j)$. The number of hard clauses for representing the transitive laws is $n \cdot (n-1) \cdot (n-2)/2$.

Definition 4. *(WPM encoding of positive value rules). Let R be a set of positive value rules. For each positive value rule $(r_i : I_i \rightarrow w_i) \in R$, $w_i > 0$, a soft clause (B_i, w_i) is introduced. The possible relations between two rules r_i and r_j can be encoded into the following clauses:*

(1) *Compatible on the same coalition: $\neg B_i \vee \neg B_j \vee S(i,j)$, $\neg S(i,j) \vee B_i$, and $\neg S(i,j) \vee B_j$.*
(2) *Compatible on different coalitions: $\neg B_i \vee \neg B_j \vee \neg S(i,j)$.*
(3) *Incompatible: $\neg B_i \vee \neg B_j$.*
(4) *Independent: no clause is generated.*

In the followings, $Hard(R)$ denotes a set of all hard clauses introduced by Definition 4 and all hard clauses representing the transitive laws.

Lemma 1. *Let R be a set of positive value rules, $R' = \{r'_1, r'_2, \ldots, r'_l\} \subseteq R$ be a feasible rule set, and $\mathcal{B}' = \{B'_1, B'_2, \ldots, B'_l\}$ be the corresponding Boolean variable set. Then, $\mathcal{B}' \cup Hard(R)$ is satisfiable.*

Lemma 2. *Let R be a set of positive value rules and $\mathcal{B}' = \{B'_1, B'_2, \ldots, B'_l\}$ be a subset of $\mathcal{B}$. If $\mathcal{B}' \cup Hard(R)$ is satisfiable, then $\{r'_1, r'_2, \ldots, r'_l\}$ is a feasible rule set.*

We omit the proof of Lemma 1 and Lemma 2 here due to the lack of space.

The above two lemmas indicate that if R' is a feasible rule set, then $\mathcal{B}' \cup Hard(R)$ is satisfiable, and vice versa. Therefore, the following theorem holds.

Theorem 3. *The encoding given by Definition 4 with the transitive laws leads MaxSAT solvers to output the correct results of the CSG problem.*

Proof: We need to prove the soft clause encoded in Definition 4 is correct, i.e., we show the result output by the MaxSAT solver is equal to the value of the feasible rule set R'. Given a feasible rule set $R' = \{r'_1, r'_2, \ldots, r'_l\}$ and the corresponding Boolean variable set $\mathcal{B}' = \{B'_1, B'_2, \ldots, B'_l\}$, according to Lemma 1, $\mathcal{B}' \cup Hard(R)$ is satisfiable. This means $\forall B'_i \in \mathcal{B}'$, (B'_i, w_i) is satisfied, thus the result output by the MaxSAT solver is $\sum_{i=1}^{l} w_i$, which is equal to the value of the feasible rule set R'. As an alternative, we can also prove the sum of weights of satisfied soft clauses is equal to the value of the feasible rule set R' by Lemma 2. ■

Example 2. For $A = \{a_1, a_2, a_3, a_4\}$, assume there are four positive value rules: $r_1 : a_1 \wedge a_2 \rightarrow 2$, $r_2 : a_1 \wedge a_2 \wedge \neg a_4 \rightarrow 2$, $r_3 : a_1 \wedge a_4 \rightarrow 1$, $r_4 : a_3 \wedge \neg a_2 \rightarrow 3$.

We introduce four soft clauses $(B_1, 2)$, $(B_2, 2)$, $(B_3, 1)$ and $(B_4, 3)$, respectively. Their rule relations are captured by the following clauses:

- r_1 and r_2 are compatible on the same coalition: $\neg B_1 \vee \neg B_2 \vee S(1,2)$, $\neg S(1,2) \vee B_1$, $\neg S(1,2) \vee B_2$.
- r_1 and r_3 are compatible on the same coalition: $\neg B_1 \vee \neg B_3 \vee S(1,3)$, $\neg S(1,3) \vee B_1$, $\neg S(1,3) \vee B_3$.
- r_1 and r_4 are compatible on different coalitions: $\neg B_1 \vee \neg B_4 \vee \neg S(1,4)$.
- r_2 and r_4 are compatible on different coalitions: $\neg B_2 \vee \neg B_4 \vee \neg S(2,4)$.
- r_2 and r_3 are incompatible: $\neg B_2 \vee \neg B_3$.

B. Handling Negative Value Rules by Rule Relations

Encoding negative value rules into WPM formulas is not as straightforward as translating positive value rules. First, negative values have to be converted to positive ones that MaxSAT solvers can deal with. In addition, dummy rules and their constraints [8] should be encoded into propositional Boolean formulas.

Definition 5. *(WPM encoding of negative value rules). Let R be a set of negative value rules. We introduce a soft clause $(\neg B_x, -w_x)$ for each negative value rule $(r_x : I_x \rightarrow w_x) \in R$ $(w_x < 0)$ where $I_x = \{a_1 \wedge a_2 \wedge \cdots \wedge a_k \wedge \neg a_{k+1} \wedge \neg a_{k+2} \wedge \cdots \wedge \neg a_m\}$ and x is the sequence number of the rule. r_x generates $m-1$ dummy rules, following two types:*

(i) $r_x^i : a_1 \wedge \neg a_{i+1} \rightarrow 0$ *where* $1 \leq i \leq k-1$,
(ii) $r_x^j : a_1 \wedge a_{j+1} \rightarrow 0$ *where* $k \leq j \leq m-1$,

where r_x^i denotes the i-th dummy rule created by r_x. The constraints on r_x and its dummy rules are specified by the following hard clauses.

(1) $B_x \vee \bigvee_{i\in\{1,\ldots,m-1\}} B_x^i$, *and*
(2) $\neg B_x^i \vee \neg B_x$, *for* $i = 1, \ldots, m-1$,

where B_x^i is a Boolean variable. $B_x^i = 1$ if r_x^i applies to a coalition C in CS and $B_x^i = 0$, otherwise.

The final social welfare after encoding is $(-W_{neg})$ larger than the original one, where W_{neg} is the sum of values of negative value rules.

Theorem 4. *The encoding given by Definition 4 and Definition 5 with transitive laws leads MaxSAT solvers to output the correct results of the CSG problem.*

Proof: Given a negative value rule r_k, if r_k is in a feasible rule set, i.e., $B_k = 1$, the payoff generated by $(\neg B_k, -w_k)$ and $I_k \rightarrow w_k$ is 0 and w_k, respectively. If r_k is not in a feasible rule set, i.e., $B_k = 0$, the corresponding payoff of $(\neg B_k, -w_k)$ and $I_k \rightarrow w_k$ is $-w_k$ and 0, respectively. Clearly, the payoff of $(\neg B_k, -w_k)$ is constantly $(-w_k)$ larger than that of $I_i \rightarrow w_i$. Considering a coalition game of n negative value rules, the social welfare after encoding negative value rules is $(-W_{neg})$ larger than the original one, where $W_{neg} = \sum_{k=1}^{n} w_k$.

Hard clauses introduced by Definition 5 (1) and (2) are encoded from Theorem 2. Now, we prove the encoding is correct. Let R' be a feasible rule set and $\mathcal{B}'$ be the corresponding Boolean variable set. Let $D(r_x) = \{r_x^1, r_x^2, \ldots, r_x^{m-1}\}$ be the set of dummy rules created by a negative value rule r_x, and $B_{D(r_x)} = \{B_x^1, B_x^2, \ldots, B_x^{m-1}\}$ be the set of corresponding Boolean variables. If $\forall r_x^i \in D(r_x)$, $r_x^i \notin R'$, i.e., $B_x^i \notin \mathcal{B}'$, we have to make $B_x \in \mathcal{B}'$ according to the hard clause introduced by Definition 5 (1), i.e., $r_x \in R'$. This agrees with the "if" part of Theorem 2. If $r_x \in R'$, i.e., $B_x \in \mathcal{B}'$, then $\forall B_x^i \in B_{D(r_x)}$, we have to make $B_x^i \notin \mathcal{B}'$ according to the hard clauses introduced by Definition 5 (2), i.e., $\forall r_x^i \in D(r_x)$, $r_x^i \notin R'$. This is consistent with the "only if" part of Theorem 2.

Under the above encoding, the CSG problem has been encoded to the one with only positive value rules. The remaining work is the same as introduced in Definition 4. Detailed proof of this part is mentioned in Lemma 1, 2, and Theorem 3. ■

Note that the construction of dummy rules guarantees that each negative value rule is incompatible with all of its dummy rules, and such "incompatible" relation has been captured by Definition 4. Therefore, in practice, hard clauses in Definition 5 (2) is redundant and can be omitted safely.

Example 3. In example 1, $r_2 : a_1 \wedge a_2 \wedge \neg a_4 \rightarrow -2$ is encoded into $(\neg B_2, 2)$. r_2 generates two dummy rules: $r_5 : a_1 \wedge \neg a_2 \rightarrow 0$, $r_6 : a_1 \wedge a_4 \rightarrow 0$. Their constraint is specified by $B_2 \vee B_5 \vee B_6$. (Note that $\neg B_2 \vee \neg B_5$ and $\neg B_2 \vee \neg B_6$ are redundant.)

The original social welfare is recovered by subtracting $(-W_{neg}) = 2$ from the result output by the solver.

V. Encoding Agent Relations into WPM

In the rule relation-based WPM approach, a single rule is the smallest unit for finding a feasible rule set, and the constraints are specified based on rule relations. In this section, we present a brand-new method on the basis of agent relations, in which the smallest unit is an agent.

A. Agent Relation

Let $\mathcal{A}_r = \{a_1, a_2, \ldots, a_n\}$ be a set of agents appearing in a rule r. Based on the assumption that each rule contains at least one positive literal, we define a Boolean variables $\mathcal{C}(i,j)$ for a pair of agents a_i and a_j, where a_i is the first positive literal in a rule. If a_i and a_j are in the same coalition, i.e, a_j is also a positive literal in r, $\mathcal{C}(i,j) = 1$, otherwise, $\mathcal{C}(i,j) = 0$.

Apparently, the relation that two agents are in the same coalition is transitive, i.e., any three agents a_i, a_j, and a_k $(1 \leq i < j < k \leq n)$ satisfy the following transitive laws: $\neg\mathcal{C}(i,j) \vee \neg\mathcal{C}(j,k) \vee \mathcal{C}(i,k)$, $\neg\mathcal{C}(i,j) \vee \neg\mathcal{C}(i,k) \vee \mathcal{C}(j,k)$, $\neg\mathcal{C}(i,k) \vee \neg\mathcal{C}(j,k) \vee \mathcal{C}(i,j)$, where n is the number of agents. The number of hard clauses for representing the transitive laws is $n \cdot (n-1) \cdot (n-2)/2$.

Definition 6. *(Agent relation-based MC-nets). An agent relation-based MC-net consists of a set of rules R. Each rule $r_i \in R$ is of the syntactic form $\mathcal{S}_i \rightarrow w_i$, where $\mathcal{S}_i$ is a conjunction of agent relations, formally expressed by $\mathcal{C}(1,2) \wedge \mathcal{C}(1,3) \wedge \cdots \wedge \mathcal{C}(1,m) \wedge \neg\mathcal{C}(1,m+1) \wedge \neg\mathcal{C}(1,m+2) \wedge \cdots \wedge \neg\mathcal{C}(1,m+n)$. A rule r_i is said to apply to a coalition C if $\mathcal{C}(1,2), \mathcal{C}(1,3), \ldots, \mathcal{C}(1,m)$ are all true, and $\mathcal{C}(1,m+1), \mathcal{C}(1,m+2), \ldots, \mathcal{C}(1,m+n)$ are all false. For a coalition C, $v(C) = \sum_{r_i \in R'} w_i$, where R' is the set of rules that apply to C. Thus, the value of a coalition structure CS is given as $\sum_{C\in CS} \sum_{r_i \in R'} w_i$.*

Hence, the problem of finding CS^* is equivalent to finding a feasible rule set by assigning the truth value to each $\mathcal{C}(i,j)$ $(i<j)$, in which the sum of the values is maximized.

Example 4. In agent relation-based MC-net, r_1, r_2, r_3, and r_4 in example 1 are expressed as: $r_1 : \mathcal{C}(1,2) \rightarrow 2$, $r_2 : \mathcal{C}(1,2) \wedge \neg\mathcal{C}(1,4) \rightarrow -2$, $r_3 : \mathcal{C}(1,4) \rightarrow 1$, $r_4 : \neg\mathcal{C}(2,3) \rightarrow 3$.

B. Handling Positive Value Rules by Agent Relations

Recall a WPM formula is expressed as $\varphi = \{(C_1, w_1), \ldots, (C_m, w_m), C_{m+1}, \ldots, C_{m+m'}\}$ where each C_i for $i = 1, \ldots, m+m'$ is a clause, i.e., disjunction of one or more literals. In contrast, in a rule $r_i : \mathcal{S}_i \rightarrow w_i$, $\mathcal{S}_i$ is expressed as a conjunction of Boolean variables $\mathcal{C}(i,j)$. In the following, we introduce a new Boolean variable u_i and encode the conjunction formula to clauses.

Definition 7. *(WPM encoding for positive value rules). For a positive value rule $r_i : \mathcal{S}_i \rightarrow w_i$ $(w_i > 0)$, where $\mathcal{S}_i = \mathcal{C}(1,2) \wedge \mathcal{C}(1,3) \wedge \cdots \wedge \mathcal{C}(1,m) \wedge \neg\mathcal{C}(1,m+1) \wedge \neg\mathcal{C}(1,m+2) \wedge \cdots \wedge \neg\mathcal{C}(1,m+n)$, the following clauses are introduced, where u_i is a new Boolean variable.*

(1) *a weighted soft clause: (u_i, w_i), and*
(2) *$(m+n-1)$ hard clauses: $\neg u_i \vee C(1,2)$, $\neg u_i \vee C(1,3)$, $\ldots$, $\neg u_i \vee C(1,m)$, $\neg u_i \vee \neg C(1,m+1)$, $\ldots$, $\neg u_i \vee \neg C(1,m+n)$.*

Theorem 5. *The encoding given by Definition 7 with the transitive laws leads MaxSAT solvers to output the correct results of the CSG problem.*

Proof: If there is a coalition structure CS, then we can make an assignment $\mathcal{A}$ which satisfies all hard clauses for the transitive laws so as to agree with CS. Reversely, if there is an assignment $\mathcal{A}$ which satisfies all hard clauses for the transitive laws, then we obtain a coalition structure CS which agrees with $\mathcal{A}$. Thus, there is a one-to-one correspondence between a set of coalition structures and a set of assignments which satisfy all hard clauses for the transitive laws.

Let CS be a coalition structure. We can make an assignment $\mathcal{A}$ which satisfies all hard clauses for the transitive laws so as to agree with CS.

If r_i applies to a coalition $C \in CS$, then $\mathcal{A}$ satisfies $\mathcal{S}_i$. This implies $\mathcal{A}$ satisfies all the corresponding hard clauses, and the corresponding soft clause (u_i, w_i) is not restricted by any hard clauses. In such a case, MaxSAT solvers prefer $u_i = 1$ to $u_i = 0$ because the payoff w_i can be gained.

If r_i does not apply to any coalition $C \in CS$, then $\mathcal{A}$ does not satisfy $\mathcal{S}_i$. This implies $\mathcal{A}$ does not satisfy at least one of the corresponding hard clauses unless making $u_i = 0$. Thus, the corresponding soft clause (u_i, w_i) is unsatisfied, that is, the payoff w_i is 0.

In either case, MaxSAT solvers calculate the correct social welfare. ■

Example 5. In example 2, $r_2 : a_1 \wedge a_2 \wedge \neg a_4 \rightarrow 2$ can be encoded into one soft clause $(u_2, 2)$ and two hard clauses $\neg u_2 \vee \mathcal{C}(1,2)$, $\neg u_2 \vee \neg\mathcal{C}(1,4)$.

C. Handling Negative Value Rules by Agent Relations

If a rule is a negative value rule $r_i : \mathcal{S}_i \rightarrow w_i$ $(w_i < 0)$, as explained in previous sections, extra transformation is needed to convert the negative weight w_i to positive, so that the MaxSAT solvers can be applied.

Definition 8. *(WPM encoding for negative value rules). For each negative value rule $r_i : \mathcal{S}_i \rightarrow w_i$ $(w_i < 0)$, the following clauses are introduced, where u_i is a new Boolean variable.*

(1) *a weighted soft clause: $(\neg u_i, -w_i)$, and*
(2) *a hard clause: $\neg\mathcal{S}_i \vee u_i$.*

The social welfare after encoding is $(-W_{neg})$ larger than the original one, where W_{neg} is the sum of negative weights.

Theorem 6. *The encoding given by Definition 7 and Definition 8 with the transitive laws leads MaxSAT solvers to output the correct results of the CSG problem.*

The proof of Theorem 6 is similar to that of Theorem 5. Because o the lack of space, we omit the proof here.

Example 6. In example 1, $r_2 : a_1 \wedge a_2 \wedge \neg a_4 \rightarrow -2$ can be encoded into a soft clause $(\neg u_2, 2)$ and a hard clause $\neg\mathcal{C}(1,2) \vee \mathcal{C}(1,4) \vee u_2$. The social welfare is 2 larger than that before encoding.

VI. Evaluation

We experimentally evaluated the performance of our rule relation-based WPM approach (RWPM) and agent relation-based WPM approach (AWPM). We carried out tests on a Core i5-2540 2.6GHz processor with 8GB RAM and used SAT4j MaxSAT [11] as our solver. In comparison, direct encoding algorithm [8] was run on an Xeon E5540 2.53GHz processor with 24GB RAM and used ILOG CPLEX as a general-purpose mixed integer programming package.

We show the performance with problem instances for the case that rules can be either positive or negative. The method of generating the instances was described in [8]. We set $\#rules = \#agents$ and vary $\#agents$ from 70 to 150 in Fig. 1, and vary $\#agents$ from 30 to 150 in Fig. 2, 3, 4. For all $\#rules$ of these Figures, 100 problem instances are generated. Each data point depicted in these Figures is the average of 100 data points.

Fig. 1 depicts the average computation time for solving the generated problem instances by the direct encoding algorithm in [8] and our MaxSAT-based approaches. It is clear that our MaxSAT-based approaches are far superior to the direct encoding algorithms. Especially, when $\#agents$ (equal to $\#rules$) increases over 110, the computation time of direct encoding goes up sharply, while the computation time of our approaches increases much slowly.

Fig. 2 magnifies the dimension of Fig. 1, showing that AWPM is more time efficient than RWPM. Especially, when $\#rules$ reaches 150, the computation time for solving problem instances by AWPM approach is less than 11 seconds, while RWPM needs more than 13 seconds. We explain the reason by investigating the number of Boolean variables and clauses in these two approaches, shown in Fig. 3 and 4.

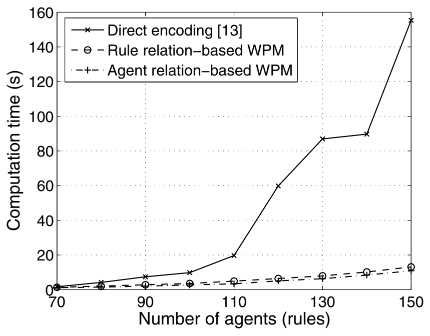

Fig. 1. Computation time of three approaches

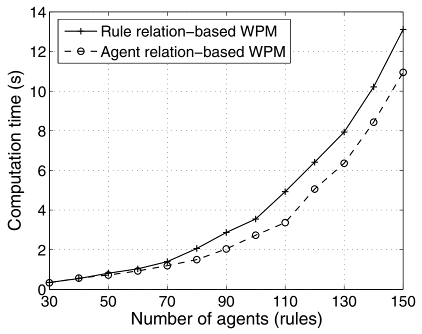

Fig. 2. Computation time of our approaches

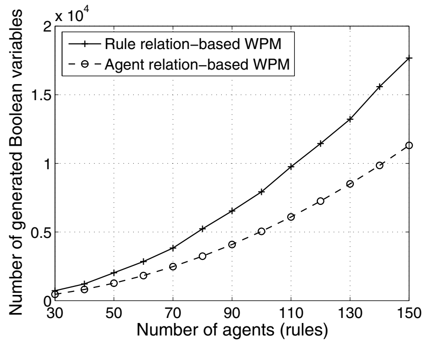

Fig. 3. Number of generated Boolean variables in our approaches

It is obvious that both the numbers of variables and clauses in RWPM are greater than that in AWPM. Generally, the computation time goes up as the number of variables and clauses increases. That is why RWPM needs more computation time than AWPM in our experiment.

VII. Conclusion

In this paper, we made the first step towards a study of applying MaxSAT solvers to the CSG problem. We show that the CSG problem can scale up significantly when MaxSAT solvers are utilized. Furthermore, we developed a new approach to capture more fine-grained relationship between pairs of agents and further speeds up the process of solving the problem.

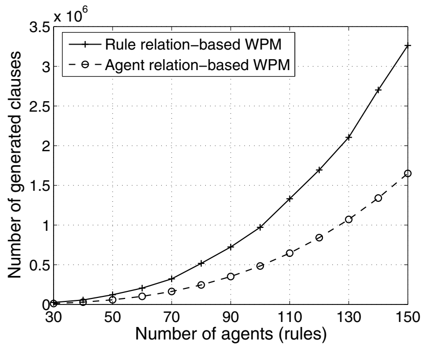

Fig. 4. Number of generated clauses in our approaches

In the future work, we would like to examine the way of handling the CSG problem with externalities, i.e., the performance of a coalition may be affected by other coalitions.

Acknowledgment

We wish to thank Prof. M. Yokoo and his students for introducing the MC-nets to the CSG problem and providing the experimental results of direct encoding. This work is partially supported by JSPS KAKENHI (21300054).

References

[1] M. J. Osborne and A. Rubinstein, *A course in Game theory*. Cambridge, MA: MIT Press, 1994.

[2] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme, "Coalition structure generation with worst case guarantees," *Artificial Intelligence*, vol. 111, pp. 209–238, 1999.

[3] S. Ieong and Y. Shoham, "Marginal contribution nets: a compact representation scheme for coalitional games," in *Proc. ACM Conference on Electronic Commerce (ACM EC'05)*, 2005, pp. 193–202.

[4] V. Conitzer and T. Sandholm, "Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains," in *Proc. Conference on Artificial Intelligence (AAAI'04)*, 2004, pp. 219–225.

[5] T. Michalak, D. Marciniak, M. Szamotulski, T. Rahwan, M. Wooldridge, P. McBurney, and N. R. Jennings, "A logic-based representation for coalitional games with externalities," in *Proc. Autonomous Agents And MultiAgent Systems (AAMAS'10)*, 2010, pp. 125–132.

[6] V. Conitzer and T. Sandholm, "Complexity of constructing solutions in the core based on synergies among coalitions," *Artificial Intelligence*, vol. 170, no. 6, 2006.

[7] N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, and M. Yokoo, "Coalition structure generation utilizing compact characteristic function representation," in *Proc. Principles and Practice of Constraint Programming (CP'08)*, 2008.

[8] S. Ueda, T. Hasegawa, N. Hashimoto, N. Ohta, A. Iwasaki, and M. Yokoo, "Handling negative value rules in mc-net-based coalition structure generation," in *Proc. Autonomous Agents And MultiAgent Systems (AAMAS'12)*, 2012, pp. 795–802.

[9] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. IEEE Symposium on the Foundations of Computer Science*, 1971, pp. 151–158.

[10] A. Biere, M. Heulu, H. Maaren, and T. Walsh, *Handbook of satisfiability*. IOS Press, 2009, ch. 19.

[11] D. L. Berre and A. Parrain, "The sat4j library, release 2.2," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 7, pp. 59–64, 2010.