Bounded Approximate Payoff Division for MC-nets Games

Hirayama, Katsutoshi

Okimoto, Tenda

(Citation)
IEICE Transactions on Information and Systems,E105.D(12):2085-2091

(Issue Date)
2022-12-01

(Resource Type)
journal article

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/0100477917

PAPER

Bounded Approximate Payoff Division for MC-nets Games

Katsutoshi HIRAYAMA[†a)], ***Member and*** **Tenda OKIMOTO**[†b)], ***Nonmember***

SUMMARY To the best of our knowledge, there have been very few work on computational algorithms for the core or its variants in MC-nets games. One exception is the work by [Hirayama, *et.al.*, 2014], where a constraint generation algorithm has been proposed to compute a payoff vector belonging to the least core. In this paper, we generalize this algorithm into the one for finding a payoff vector belonging to ε-core with pre-specified bound guarantee. The underlying idea behind this algorithm is basically the same as the previous one, but one key contribution is to give a clearer view on the pricing problem leading to the development of our new general algorithm. We showed that this new algorithm was correct and never be trapped in an infinite loop. Furthermore, we empirically demonstrated that this algorithm really presented a trade-off between solution quality and computational costs on some benchmark instances.

key words: *coalitional games, MC-nets, ε-core, constraint generation*

1. Introduction

Coalitional games have attracted a lot of attentions from researchers on multi-agent systems since they give a solid theoretical foundation for the analysis on cooperation among multiple agents. Recent studies reported that coalitional games are applied to some practical applications such as cloud manufacturing [1], ride sharing [2], and so on.

Coalitional games are denoted by (A, v), where A is a set of *agents* $\{1, 2, \ldots, n\}$ and $v : 2^A \rightarrow \Re$ is a *characteristic function* that computes a reward when a subset of agents forms a *coalition* to collaborate with each other.

The characteristic function was formerly assumed to be provided by the black-box function. A simple-table representation was enough to deal with the game having small number of agents, but totally far from practical to deal with the game having dozens or hundreds of agents. The *Marginal Contribution network* (MC-nets) is one of concise representations of characteristic functions, where it uses a set of logical rules to compute a value for a given coalition [3]. In this paper, the coalitional game whose characteristic function is represented by MC-nets rules is called *MC-nets game*.

In coalitional games, there are two major computational problems called the *coalition structure generation problem* and the *payoff division problem*. The coalition structure generation problem is related to finding a set partition over the agents that maximizes the social surplus, in other words, the total sum of characteristic function values. Regarding MC-nets games, the coalition structure generation problem is NP-hard and a number of algorithms have been proposed so far in the literature [4], [5].

The payoff division problem of coalitional games is to find a "reasonable" division of given payoff over the agents. Two major solution concepts, the *core* and the *Shapley values*, are well-known for this problem. These two solution concepts are based on different indicators (the *excess* for the core; the *marginal contribution* for the Shapley values) and evaluate payoff divisions from completely different perspectives. Therefore, it is important to be able to compute both for a given representation. Regarding MC-nets games, although the cost of computing the Shapley values is shown to be linear to the size of representation [3], finding the core or its variants is computationally challenging. For example, the problem of deciding *core-non-emptiness* is shown to be co-$\mathcal{NP}$-complete [6]. Therefore, it is very important to develop a practical approximate solution algorithm for finding the core or its variants in MC-nets games. However, unlike against the coalition structure generation problem, there have been very few work toward this goal.

One exception is the work in [7], where a *constraint generation algorithm* has been proposed to compute a payoff vector belonging to the least core for MC-nets games. Constraint generation, whose dual is *column generation*, is the technique to solve the linear programming problem with a potentially huge number of constraints. It alternates, until the candidate solution becomes feasible, solving the *restricted master problem* aiming to create a candidate solution to the original linear programming problem and the *pricing problem* aiming to create a constraint to be added to the restricted master problem. In this paper, we generalize this algorithm into the one for finding a payoff vector belonging to ε-core with pre-specified bound guarantee. Indeed, both the algorithm in [7] and our new generalized algorithm exploit the same restricted master problem and pricing problem, but one key contribution of this paper is to give a clearer view on the pricing problem. They have seen the pricing problem as the tool for checking feasibility in [7], while we will see it as the tool for computing the upper bound in this paper. This new view on the pricing problem allows us to develop a new constraint generation algorithm that works in more general context.

The remaining part of this paper will be organized as

Manuscript received March 7, 2022.
Manuscript revised June 11, 2022.
Manuscript publicized September 13, 2022.

[†]The authors are with Graduate School of Maritime Sciences, Kobe University, Kobe-shi, 658–0022 Japan.
a) E-mail: hirayama@maritime.kobe-u.ac.jp (Corresponding author)
b) E-mail: tenda@maritime.kobe-u.ac.jp

DOI: 10.1587/transinf.2022EDP7039

follows. Section 2 gives necessary background knowledge, which includes the definition of coalitional game, ε-core and the least core, and MC-nets, and discusses the significance of seeking the core or its variants in MC-nets games. Section 3 introduces our new constraint generation algorithm for ε-core that guarantees a pre-specified bound on solution quality after giving the overview of the algorithm in [7]. Next, Sect. 4 demonstrates that our new constraint generation algorithm can find a reasonably better solution with relatively small amount of computation costs for some problem instances involving up to 100 agents, and finally Sect. 5 concludes this work.

2. Background

This section gives necessary background and discusses the significance of seeking the core or its variants in MC-nets games.

2.1 Coalitional Game

Coalitional games are denoted by (A, v), where A is a set of *agents* $\{1, 2, \ldots, n\}$ and $v : 2^A \to \Re$ is a *characteristic function* that computes a reward when a subset of agents forms a *coalition* to collaborate with each other. We assume that the value of $v(\emptyset)$ is always zero. The following example will be used throughout the paper to illustrate our algorithm.

Example 1 (Characteristic function)**.** *We consider the following coalitional game (A, v), where $A = \{1, 2, 3, 4\}$ and $v(\cdot)$ as a simple-table representation: $v(\{1\}) = v(\{2\}) = v(\{4\}) = 0$; $v(\{3\}) = 3$; $v(\{1, 2\}) = v(\{2, 3\}) = v(\{2, 4\}) = 0$; $v(\{1, 4\}) = 1$; $v(\{1, 3\}) = v(\{3, 4\}) = 3$; $v(\{1, 2, 3\}) = v(\{2, 3, 4\}) = 0$; $v(\{1, 2, 4\}) = 3$; $v(\{1, 3, 4\}) = 4$; $v(\{1, 2, 3, 4\}) = 3$.*

One computational problem in coalitional games is the *coalition structure generation problem*. A *coalition structure* is a set partition over the agents and the reward made by a coalition structure is the sum of characteristic function values of coalitions constituting the coalition structure. A goal of this problem is to find an *optimal coalition structure*, denoted by CS^*, that achieves the maximum reward.

Example 2 (Optimal coalition structure)**.** *An optimal coalition structure of Example 1 is $\{\{1, 2, 4\}, \{3\}\}$ since its reward of 6 is maximum.*

The other computational problem in coalitional games is the *payoff division problem*, of which goal is to find a "reasonable" division of given payoff over the agents. We assume that such given payoff be the maximum reward, denoted by p_{max}, obtained by forming the optimal coalition structure. A division of given payoff is represented by payoff vector $x = (x_1, \ldots, x_n)$.

Two major solution concepts, the *core* and the *Shapley values*, are well-known for the payoff division problem. Regarding MC-nets games that will be explained in Sect. 2.3, although the cost of computing the Shapley values is shown to be linear to the size of representation [3], finding the core or its variants is computationally challenging. For example, the problem of deciding *core-non-emptiness* is shown to be co-$\mathcal{NP}$-complete [6]. Since there have been very few work on developing computational algorithms for finding the core or its variants in MC-nets games to the best of our knowledge, we would like to focus on this unexplored field.

2.2 ε-Core and Least Core

The core or its variants are based on the notion of *excess*, which represents the degree of "dissatisfaction" of coalition S against payoff vector x.

Definition 1 (Excess)**.** *The excess $e(x, S)$ of coalition S against payoff vector x is computed by*

$$e(x, S) \equiv v(S) - \sum_{i \in S} x_i.$$

For given payoff vector x, coalition S having a positive excess, i.e., $v(S) > \sum_{i \in S} x_i$, is called a *blocking coalition*. If there is a blocking coalition S against x, every member of S does not satisfy with x and has a common incentive to break off the optimal coalition structure. The core is a set of payoff vectors for which there is no blocking coalition, in other words, every possible coalition has non-positive excess. This idea can be generalized into ε-core.

Definition 2 (ε-core)**.** *A set of payoff vectors, denoted by C^ε, for which every possible coalition has the value of excess of ε or lower is called ε-core. Mathematically, C^ε is a set of payoff vectors that satisfy the following constraints.*

$$e(x, S) \leq \varepsilon, \ \forall S \subset A, S \neq \emptyset,$$
$$\textstyle\sum_{i \in A} x_i = p_{\max}.$$

Obviously, the core is C^0. Generally, when $\varepsilon' < \varepsilon$, we can see that $C^{\varepsilon'} \subset C^\varepsilon$. Namely, when reducing the value of ε, we observe that ε-core gradually gets smaller and eventually becomes empty.

Definition 3 (Least Core)**.** *The least core is the intersection C^{ε^*} of all of the non-empty ε-core. C^{ε^*} is identical with a set of optimal solutions to the following linear programming problem.*

$$\begin{aligned} \text{min.} \quad & \varepsilon \\ \text{s.t.} \quad & e(x, S) \leq \varepsilon, \quad \forall S \subset A, S \neq \emptyset, \\ & \textstyle\sum_{i \in A} x_i = p_{max}, \end{aligned} \tag{1}$$

Therefore, we need to compute an optimal solution to (1) if we would like to get a payoff vector belonging the least core. Similarly, we need to compute a feasible solution to (1) with a certain value for ε if we would like to get a payoff vector belonging to ε-core. However, since the number of constraints on excess in (1) grows exponentially with the number of agents, the naive computing method by just solving this LP model collapses empirically against problem instances having more than 20 agents in case of using a computer with 8 GB memory [7].

2.3 MC-nets

The characteristic function was formerly assumed to be provided by the black-box function. A simple-table representation was enough to deal with the game having small number of agents, but totally far from practical to deal with the game having dozens or hundreds of agents. Therefore, much research has been made recently on the concise representations of characteristic functions. Of these, *Synergy Coalition Group* (SCG) [9] and *Marginal Contribution networks* (MC-nets) [3] are basic concise representations which is simple and easy to handle.

SCG is a concise representation for *super-additive* games that describes only the values of coalitions in which there are positive "synergy" between agents belonging to those coalitions. The practical benefit of SCG is, once an input by SCG is given, to allow for an efficient procedure for checking whether a payoff vector is in the core. However, major drawback of SCG is that, in order to compute the value of a coalition, an optimal partition of its members into subgroups is needed to be found, which is generally very expensive.

On the other hand, MC-nets is a concise representation where it exploits a set of logical rules to compute a value for a given coalition. Each MC-nets rule r can be written by $(P_r, N_r) \to v_r$, where P_r is a set of agents that are expected to exist in the coalition and N_r is a set of agents that are expected not to exist in the coalition. We assume $P_r \cap N_r = \emptyset$ for any rule r. If both $P_r \subseteq S$ and $N_r \cap S = \emptyset$ are met, rule r can be applied to coalition S and the value of v_r is added to its characteristic function value. Note that the value of v_r can be positive or negative, but not zero. Suppose the set of rules that can be applied to some coalition S is denoted by R_S, the characteristic function value $v(S)$ is given by $v(S) \equiv \sum_{r \in R_S} v_r$.

Example 3 (MC-nets rules)**.** *A rule set* $R = \{r_1, r_2, r_3, r_4\}$*, where*

$$r_1 : (\{1,2\}, \emptyset) \to 2,$$
$$r_2 : (\{1,2\}, \{4\}) \to -2,$$
$$r_3 : (\{1,4\}, \emptyset) \to 1,$$
$$r_4 : (\{3\}, \{2\}) \to 3,$$

gives the same characteristic function value for any coalition in Example 1. For example, we have $v(\{1,2,4\}) = v_{r_1} + v_{r_3} = 2 + 1 = 3$ *since rules* r_1 *and* r_3 *can be applied to coalition* $\{1,2,4\}$.

MC-nets is known to be *fully-expressive*, that means any characteristic function can be described by MC-nets rules [3].

One of the games that can be represented by MC-nets is the weighted graph game [8]. In this game, we are given weighted graph $G = (V, E, w)$, where V is a set of vertices, $E = \{(i,j) \mid i, j \in V\}$ is a set of edges, and $w : E \to \Re$ is a function that returns a real value for any edge, and consider a vertex as an agent. For any coalition S, the value of the characteristic function can be computed by $v(S) = \sum_{(i,j) \in E[S]} w(i,j)$, where $E[S] = \{(i,j) \mid i, j \in S\}$.

2.4 MC-nets and Core

Regarding MC-nets games, although the cost of computing the Shapley values is shown to be linear to the size of representation [3], finding the core or its variants is computationally challenging. For example, the problem of deciding *core-non-emptiness* is shown to be co-$\mathcal{NP}$-complete [6].

On the other hand, there has been an argument that, since the core or its variants are known to be computed in polynomial time by using SCG, all we need to do is just selecting appropriate concise representation depending on the solution concept we want to compute. However, we consider that this approach would work in practice only if an efficient translation was possible between any pair of concise representations. At least regarding the translation from MC-nets to SCG, it is easy to prove that there would never be a polynomial-time translation algorithm as long as $\mathcal{P} \neq$ co-$\mathcal{NP}$.

Theorem 1. *If* $\mathcal{P} \neq$ co-$\mathcal{NP}$*, then MC-nets representation cannot be translated into SCG representation in polynomial time.*

Proof. Suppose that there exists such a polynomial-time translation algorithm. By using this algorithm, deciding the core-non-emptiness in MC-nets games becomes solvable in polynomial time, which results in $\mathcal{P} =$ co-$\mathcal{NP}$. □

Therefore, we consider that it is an important research issue to develop an algorithm for directly computing the core or its variant for MC-nets games without translating into another representation that might have some efficient algorithm.

3. Constraint Generation Algorithm

A *constraint generation algorithm* has been proposed to compute a payoff vector belonging to the least core [7]. Constraint generation, whose dual is *column generation*, is the technique to solve the linear programming problem with a potentially huge number of constraints. It alternates solving the *restricted master problem* and the *pricing problem* described below.

3.1 Restricted Master Problem

Generally, the restricted master problem is the one consisting of exactly the same set of variables and objective function as the original linear programming problem but having only a partial set of original constraints. Suppose $\mathcal{T}$ is any set of coalitions, the restricted master problem of (1) can be formulated as follows.

$$\begin{aligned} \text{min.} \quad & \varepsilon \\ \text{s.t.} \quad & e(x,S) \leq \varepsilon, \quad \forall S \in \mathcal{T}, S \neq \emptyset, \\ & \textstyle\sum_{i \in A} x_i = p_{max}. \end{aligned} \tag{2}$$

Example 4 (Restricted master problem)**.** *If $\mathcal{T}$ consists of only the individual coalitions in Example 1, i.e., $\mathcal{T} = \{\{1\},\{2\},\{3\},\{4\}\}$, the restricted master problem that corresponds to this $\mathcal{T}$ is*

$$\begin{array}{ll} \text{min.} & \varepsilon \\ \text{s.t.} & 0 - x_1 \le \varepsilon, \\ & 0 - x_2 \le \varepsilon, \\ & 3 - x_3 \le \varepsilon, \\ & 0 - x_4 \le \varepsilon, \\ & x_1 + x_2 + x_3 + x_4 = 6, \end{array}$$

of which the optimal value $\hat{\varepsilon}$ is -0.75 and an optimal solution $\hat{x}$ is $(0.75, 0.75, 3.75, 0.75)$.

Lemma 1. *The optimal value $\hat{\varepsilon}$ to the restricted master problem (2) gives a lower bound on the optimal value ε^* to (1).*

Proof. It is obvious since (2) is the same as (1) except for the fact that it includes only a partial set of constraints. □

3.2 Pricing Problem

Generally, the pricing problem is the one specially designed to check the feasibility of given variable assignments on the original linear programming problem. Even if the original linear programming problem has too many constraints to enumerate in practice, some information regarding the feasibility of given variable assignments can be obtained by solving the pricing problem.

Regarding the problem in (1), the feasibility of $\hat{x}$ and $\hat{\varepsilon}$ that are obtained by solving the restricted master problem of (2) is quite important since, if these are feasible, they are ensured to be an optimal solution to (1) due to the *complementary slackness theorem* of linear programming problem [7]. Therefore, we have presented the pricing problem described by formula (14) in [7], which was in a form of 0-1 integer minimization problem. However, we find the semantics of this previous pricing problem in [7] a bit complicated.

In this paper, we will present the following "new" formulation of the pricing problem. It will be shown later that this is exactly a tool for computing an upper bound on the optimal value to (1).

$$\begin{array}{ll} \text{max.} & \sum_{r\in R} v_r\beta_r - \sum_{i\in A} \hat{x}_i\alpha_i \\ \text{s.t.} & \sum_{i\in P_r} \alpha_i + \sum_{i\in N_r}(1-\alpha_i) \ge |P_r \cup N_r|\beta_r,\ \forall r \in R^+, \\ & \sum_{i\in P_r}(1-\alpha_i) + \sum_{i\in N_r} \alpha_i \ge 1 - \beta_r,\ \forall r \in R^-, \\ & \sum_{i\in A} \alpha_i \le |A| - 1, \\ & \sum_{i\in A} \alpha_i \ge 1, \\ & \alpha_i, \beta_r \in \{0,1\},\ \forall i \in A, \forall r \in R, \end{array} \tag{3}$$

where the meanings of its variables and constants are as follows:

- α : a set of binary *decision variables* that represents a coalition, whose element α_i takes one if agent i is in the coalition or zero otherwise.
- β : a set of binary variables that represents MC-nets rules that are applied for the coalition of α. Its element β_r takes one if MC-nets rule r is applied for the coalition or zero otherwise. This is actually a dependent variable whose value will be automatically determined by the values of α.
- $\hat{x}$: a payoff vector that are obtained by solving the restricted master problem of (2).
- A : a set of agents, $A = \{1, 2, \ldots, n\}$.
- R : a set of MC-nets rules, $R = \{r_1, r_2, \ldots, r_m\}$, but abbreviated by $R = \{1, 2, \ldots, m\}$ in this formulation.
- P_r : a set of agents that are expected to exist by MC-nets rule r.
- N_r : a set of agents that are expected not to exist by MC-nets rule r.
- v_r : increment of characteristic function value when MC-nets rule r is applied. Note that this value is positive or negative, but not zero.
- R^+ : a set of MC-nets rules whose values of v_r are positive.
- R^- : a set of MC-nets rules whose values of v_r are negative. Clearly, $R = R^+ \cup R^-$ and $R^+ \cap R^- = \emptyset$ holds.

Comparing formula (14) in [7] and this pricing problem of (3), they have exactly the same constraints but differ in their objectives. In (3), the objective function is multiplied by -1 while its direction is reversed from min to max. Namely, these two are essentially the same problem, but different only in notation. However, this small change in notation makes the semantics of pricing problem more simple and leads to the generalization of previous algorithm as shown in the next subsection.

Example 5 (Pricing problem)**.** *For the MC-nets rules in Example 3 and payoff vector $\hat{x} = (0.75, 0.75, 3.75, 0.75)$ in Example 4, we can create the following pricing problem.*

$$\begin{array}{ll} \text{max.} & 2\beta_1 - 2\beta_2 + \beta_3 + 3\beta_4 \\ & -0.75\alpha_1 - 0.75\alpha_2 - 3.75\alpha_3 - 0.75\alpha_4 \\ \text{s.t.} & \alpha_1 + \alpha_2 \ge 2\beta_1, \\ & (1-\alpha_1) + (1-\alpha_2) + \alpha_4 \ge 1 - \beta_2, \\ & \alpha_1 + \alpha_4 \ge 2\beta_3, \\ & \alpha_3 + (1-\alpha_2) \ge 2\beta_4, \\ & \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \le 3, \\ & \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \ge 1, \\ & \alpha_i, \beta_r \in \{0,1\}, \forall i \in A, \forall r \in R, \end{array}$$

whose optimal value z^* is 0.75 and optimal solution α^* is $(1, 1, 0, 1)$.

The semantics of pricing problem of (3) is, given a payoff vector $\hat{x}$, to find coalition α^* with maximum excess z^* against $\hat{x}$ among all possible coalitions with the size of less than n. This will be explained in detail using Example 5. The constraints of (3) is configured such that if we give any coalition α whose size is less than n, then they determine a set of MC-nets rules β that can be applied to that coalition. In Example 5, when α is set to $(1, 1, 0, 1)$ that means coalition $\{1, 2, 4\}$, β becomes inevitably $(1, 0, 1, 0)$ that indicates rules r_1 and r_3 are applied to coalition $\{1, 2, 4\}$ (see Example 3). The first term of the objective function of (3) computes the sum of the values of the applicable rules, namely,

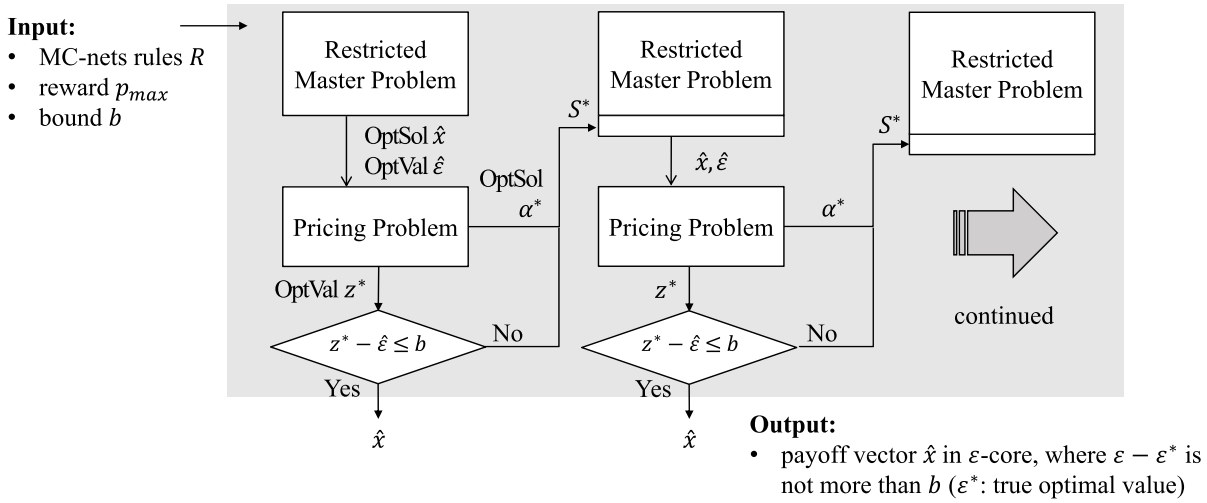

Fig. 1 Sketch of the behavior of CGεB

characteristic function value, for coalition α, while the second term of it computes the share of coalition α over a given payoff vector of $\hat{x}$. When α is set to $(1, 1, 0, 1)$ in Example 5, the first term of the objective function becomes 3, which is the value of $v(\{1, 2, 4\})$, and the second term becomes 2.25, which is the value of $\sum_{i\in\{1,2,4\}} \hat{x}_i$. By subtracting the second term from the first term, the objective function of (3) exactly computes the excess of coalition α against $\hat{x}$. We should notice that decision variable α can take any 0-1 vector except all-ones or all-zeros. Hence, given a payoff vector $\hat{x}$, the pricing problem of (3) computes coalition α^* with maximum excess z^* against $\hat{x}$ among all possible coalitions with the size of less than n.

Lemma 2. *The optimal value z^* to the pricing problem (3) with a given payoff vector $\hat{x}$ gives an upper bound on the optimal value ε^* to (1).*

Proof. Since $\hat{x}$ is an optimal solution to the restricted master problem of (2), it obviously satisfies the last constraint of (1). Furthermore, due to the semantics of the pricing problem (3), constraint $e(\hat{x}, S) \leq z^*$ is satisfied for every coalition in $\{S \mid S \subset A, S \neq \emptyset\}$. Therefore, assignment $(\hat{x}, z^*)$ for variables (x, ε) in (1) is a feasible solution to (1), which gives z^* as its objective function value. □

3.3 New Algorithm for ε-Core with Bound Guarantee

With Lemmas 1 and 2, we are now ready to present a new constraint generation algorithm for finding a payoff vector in ε-core with bound guarantee. The input of this algorithm is set R of MC-nets rules, maximum reward p_{max} to be divided over the agents, and absolute bound b (≥ 0) on the difference between the resulting value for ε and its true optimal value ε^*. The output of this algorithm is a payoff vector $\hat{x}$ in ε-core, where ε minus ε^* is guaranteed to be not more than b. Although we never know the true optimal value ε^* either before or after performing this algorithm, we can obtain such a payoff vector by this algorithm.

The detailed procedure of this algorithm, which we will call CGεB (Constraint Generation algorithm for ε-core with Bound guarantee), is shown in Algorithm 1. At line 1 of this algorithm, $\mathcal{T}$ is set with any set of coalitions. At line 3, the restricted master problem with this $\mathcal{T}$ is solved to obtain lower bound $\hat{\varepsilon}$ and candidate payoff vector $\hat{x}$ (see Example 4). At line 4, the pricing problem with $\hat{x}$ is solved to obtain upper bound z^* and coalition α^* that gives this max excess against $\hat{x}$ (see Example 5). At line 5, the difference between upper bound z^* and lower bound $\hat{\varepsilon}$ is checked whether not more than b or not. The procedure stops with the current payoff vector $\hat{x}$ if this is the case at line 6, or adds coalition S^* that corresponds to α^* to $\mathcal{T}$ otherwise at line 8.

Example 6 (Running example of CGεB)**.** *Suppose b is set to zero when we solve the ongoing example. Since upper bound z^* was 0.75 in Example 5 and lower bound $\hat{\varepsilon}$ was -0.75 in Example 4, formula $z^* - \hat{\varepsilon} \leq b$ is not satisfied in this case. Algorithm CGεB thus adds coalition $S^* = \{1, 2, 4\}$, which corresponds to α^* in Example 5, to $\mathcal{T}$ in Example 4.*

In this way, Algorithm CGεB repeats the loop from line 3 through 9 until the condition on bound is satisfied while generating and adding one constraint per every iteration. Figure 1 illustrates the behavior of CGεB.

The correctness and termination of CGεB are proven below.

Theorem 2 (Correctness)**.** *Algorithm CGεB returns a payoff vector in ε-core, where ε minus ε^* is not more than b.*

Proof. This obviously holds from Lemmas 1, 2, and the termination criteria at line 5 in Algorithm 1. □

Theorem 3 (Termination)**.** *Algorithm CGεB terminates within a finite time.*

Proof. Since the possible number of coalitions is at most $2^n - 1$, if a new coalition that does not appear in $\mathcal{T}$ is selected as S^* to be added to $\mathcal{T}$ every time in each iteration of the loop of CGεB, the restricted master problem of (2) will be eventually identical to (1). At that time, $\hat{\varepsilon}$ gets equal to ε^*

Algorithm 1 CGεB

Input: set R of MC-nets rules; maximum reward p_{max} to be divided into the agents; absolute bound b (≥ 0) on the difference between the resulting value for ε and its true optimal value ε^*

Output: payoff vector $\hat{x}$ in ε-core, where ε minus ε^* is guaranteed to be not more than b

1: set $\mathcal{T}$ with any set of coalitions.
2: **loop**
3: solve the restricted master problem of (2) with $\mathcal{T}$ to find its optimal value $\hat{\varepsilon}$ and optimal solution $\hat{x}$
4: solve the pricing problem of (3) with $\hat{x}$ to find its optimal value z^* and optimal solution α^*
5: **if** $(z^* - \hat{\varepsilon} \leq b)$ **then**
6: print $\hat{x}$ and stop
7: **else**
8: add coalition S^* that corresponds to α^* into $\mathcal{T}$
9: **end if**
10: **end loop**

and thus CGεB will terminate. Hence, our goal is to show that such a coalition is selected every time in each iteration. Assume that, for current payoff vector $\hat{x}$, coalition S^*, which was already selected in a previous iteration, is again selected at line 8. Since $z^* = v(S^*) - \sum_{i \in S^*} \hat{x}_i$, we have

$$v(S^*) - \sum_{i \in S^*} \hat{x}_i - \hat{\varepsilon} > b \tag{4}$$

due to the fact that line 8 is running. On the other hand, since S^* was already selected in a previous iteration and must be in $\mathcal{T}$, current payoff vector $\hat{x}$ should satisfy

$$v(S^*) - \sum_{i \in S^*} \hat{x}_i \leq \hat{\varepsilon} \tag{5}$$

due to the fact that $\hat{x}$ is feasible for (2). Since $b \geq 0$, (4) and (5) contradict each other. Therefore, the assumption is denied, in other words, coalition S^* has never been selected in a previous iteration. □

4. Experimental Evaluation

To the best of our knowledge, there is no comparable algorithm to CGeB except for the one in [7]. Therefore, in this section, we will experimentally demonstrate whether the generalization of previous algorithm works in practice to produce expected results and performance.

In order to observe the actual performance of CGεB algorithm, we have conducted an experiment on MC-nets instances that were also used in [4] and [7]. These MC-nets instances have been created using the *decay distribution* under the CATS framework [10] as detailed in [4]. In these instances, the number $|R|$ of MC-nets rule and the number $|A|$ of agents is the same and varies from 10 to 100 in steps of 10. For each number of $|A|$ or $|R|$, 100 instances were randomly created to produce 1000 instances in total.

In this experiment, we set the value of bound b as $k \times p_{max}$, where k ranges over $\{0.001, 0.005, 0.01, 0.05, 0.1\}$. The smaller the value of k is, the closer to the least core the payoff vector found by this algorithm is. The CGεB algorithm that uses $k \times p_{max}$ as b is denoted by CGεB(k).

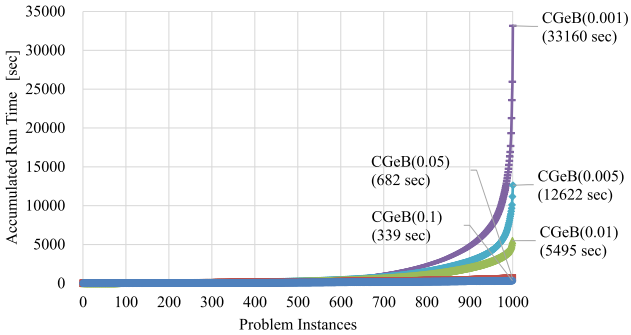

Fig. 2 Cactus plots of accumulated run-time for CGεB(k)

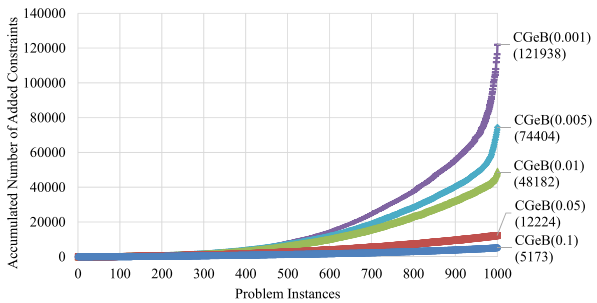

Fig. 3 Cactus plots of accumulated number of added constraints for CGεB(k)

Before applying CGεB(k) for MC-nets games, we need the value of p_{max} that will be divided over the agents. This value should be the maximum reward that can be obtained by solving the coalition structure generation problem for MC-nets games. In this experiment, we have actually used those maximum rewards that were reported by the authors of [4].

All of the experiments were conducted on Intel Core i7-3960X (3.30 GHz, 12 CPUs) with 32 GB memory, Windows 7 Pro 64 bit, and IBM ILOG CPLEX 12.6.3 with Java 8 API.

We have set a cut-off time to 2 hours (7200 sec) for each run of algorithm. If a run exceeds this cut-off time, the algorithm was forced to be terminated and recorded as completed at the cut-off time. We report that there existed only one such run that was cut off for CGεB(0.001).

The *cactus plots* of accumulated run-time and accumulated number of added constraints for each CGεB(k) algorithm are shown in Figs. 2 and 3, respectively. The cactus plot for some specific algorithm is the one, where the x-axis indicates the whole set of problem instances and the y-axis indicates the accumulated costs (run-time or the number of added constraints in this experiment). The problem instances on the x-axis are all sorted by the ascending order of costs depending on the algorithm used. In the cactus plot representation, it shows that an algorithm with a smaller degree of increase in the line plot has better performance. We show the accumulated data of 1000 problem instances in parentheses under the label of each algorithm in both Figs. 2 and 3.

As these two figures shows, if we increase the value of b (the value of k), both the run-time and the number of added

constraints are drastically reduced as we have expected. On the other hand, a payoff vector belonging to the least core can be obtained by CGεB(0) (by setting the value of k to 0). Although not shown in the graph, the accumulated run-time of 1000 instances of such CGεB(0) exceeds 55,000 seconds, which is much more expensive compared to CGεB(k) with $k > 0$.

5. Conclusion

In this paper, we generalize the algorithm in [7] into the one, which is called CGεB, for finding a payoff vector belonging to ε-core with pre-specified bound guarantee. Considering the fact that there are very few algorithms for finding the core or its variants for MC-nets games at present, we believe that this study can be regarded as a valuable first step in the field.

It might need to be discussed as to whether the approximate payoff division, rather than the optimal one, is acceptable to the agents. For example, some agents who are most dissatisfied with a current approximate payoff division may refuse to accept it and argue to continue further computation for a better payoff division, but some other agents may be satisfied with that approximate one. The question is whether to terminate the algorithm with that approximate payoff division under these situations. To answer it properly, we need to develop a new solution concept involving not only the degree of dissatisfaction but also the attitude toward the consumption of computational resources, which we consider is beyond the scope of this paper though. Devising more sophisticated termination conditions for CGεB that reflect the above new solution concept, rather than just cutting off using the bound, is one of our interesting future work.

Other possible future work will include applying CGεB algorithm to weighted graph games, exploring how to generate a bunch of constraints instead of one at each iteration of CGεB, creating more realistic and meaningful benchmark instances for MC-nets games, and others.

Acknowledgements

The authors would like to thank Jun Akagi for helping us conduct computational experiments and prepare an early version of this paper.

References

[1] J. Chen, G.Q. Huang, J-Q. Wang, and C. Yang, "A cooperative approach to service booking and scheduling in cloud manufacturing," European Journal of Operational Research, no.3, vol.273, pp.861–873, 2019. DOI: 10.1016/j.ejor.2018.09.007.

[2] F. Bistaffa, A. Farinelli, G. Chalkiadakis, and S.D. Ramchurn, "A cooperative game-theoretic approach to the social ridesharing problem," Artificial Intelligence, vol.246, pp.86–117, 2017. DOI: 10.1016/j.artint.2017.02.004.

[3] S. Ieong and Y. Shoham, "Marginal contribution nets: a compact representation scheme for coalitional games," Proc. 6th ACM Conference on Electronic Commerce (EC-2005), pp.193–202, 2005. DOI: 10.1145/1064009.1064030.

[4] S. Ueda, T. Hasegawa, N. Hashimoto, N. Ohta, A. Iwasaki, and M. Yokoo, "Handling negative value rules in MC-net-based coalition structure generation," Proc. 11th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2012), pp.795–802, 2012. https://dl.acm.org/doi/abs/10.5555/2343776.2343810.

[5] X. Liao, M. Koshimura, K. Nomoto, S. Ueda, Y. Sakurai, and M. Yokoo, "Improved WPM encoding for coalition structure generation under MC-nets," Constraints, vol.24, no.1, pp.25–55, 2019. DOI: 10.1007/s10601-018-9295-4.

[6] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello, "On the complexity of core, kernel, and bargaining set," Artificial Intelligence, vol.175, no.12-13, pp.1877–1910, 2011. DOI: 10.1016/j.artint.2011.06.002.

[7] K. Hirayama, K. Hanada, S. Ueda, M. Yokoo, and A. Iwasaki, "Computing a payoff division in the least core for MC-nets coalitional games," Proc. 17th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2014), pp.319–332, 2014. DOI: 10.1007/978-3-319-13191-7_26.

[8] X. Deng and C.H. Papadimitriou, "On the complexity of cooperative solution concepts," Mathematics of Operations Research, vol.19, no.2, pp.257–266, 1994. http://www.jstor.org/stable/3690220.

[9] V. Conitzer and T. Sandholm, "Complexity of constructing solutions in the core," Artificial Intelligence, vol.170, pp.607–619, 2006. DOI: 10.1016/j.artint.2006.01.005.

[10] T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," Artificial Intelligence, vol.135, no.1-2, pp.1–54, 2002. DOI: 10.1016/S0004-3702(01)00159-X.

Katsutoshi Hirayama received the Ph.D. in Engineering from Osaka University in 1995. During 1995–2003, he worked for Kobe University of Mercantile Marine as a research assistant, lecturer, and associate professor. After becoming an associate professor of Kobe University in 2003, he has been a professor of Kobe University since 2013. During 1999–2000, he stayed in the Robotics Institute of Carnegie Mellon University as a visiting scientist. His research interests are multi-agent systems, constraint satisfaction, combinatorial optimization, and others. He received the IFAAMAS Influential Paper Award in 2010, the 2014 JSSST Basic Research Award in 2015, and the Funai Best Paper Award in 2019. He is a member of the Information Processing Society of Japan (IPSJ), the Japanese Society of Artificial Intelligence (JSAI), the Operations Research Society of Japan (ORSJ) and the Japan Society for Software Science and Technology (JSSST).

Tenda Okimoto is an Associate Professor at Kobe university since 2014. Before joining Kobe university, he worked as an assistant professor in Transdisciplinary Research Integration Center (TRIC)/National Institute of Informatics (NII). He got his Dilpom from Freiburg University in 2008 and his PhD degree (Informatics) from Kyushu University in 2012. His research interests include Artificial Intelligence (AI), Game Theory, Operations Research (OR), and Disaster Research. He is a member of the Japanese Society of Artificial Intelligence (JSAI).