ICIC Express Letters
Part B: Applications
Volume **13**, Number **11**, November **2022**
 ISSN 2185-2766
pp. **1143–1153**

COALITION FORMATION OF BUYERS IN REAL WORLD AGRICULTURE DOMAIN

Chattrakul Sombattheera

Multi-agent Intelligent Simulation Laboratory (MISL)
Department of Information Technology
Faculty of Informatics
Mahasarakham University
Khamriang Sub-District, Kantarawichai District, Mahasarakham 44150, Thailand
chattrakul.s@msu.ac.th

Received March 2022; accepted May 2022

Abstract. *This work presents algorithms for coalition formation of buyers in agriculture domain, where farmers use an application on their mobile phones to buy fertilizers (or other resources) in group and benefit from fair distribution of discounts. We propose algorithms that i) help farmers to form optimal coalitions of buyers, and ii) calculate Shapley value for fair distribution of discount among farmers in timely fashion. Since computing Shapley value can be intractable for even a small input, most researches tend to approximate the value. We compare our algorithm for computing Shapley value with other recent works and find that ours performs better in most cases. Particularly, our algorithm is faster for cases of 5-15 agents, which are more often. In cases of 19-20 agent, which seldom happen, ours is slightly slower but is still within seconds. This is due to the fact that ours works in non-structured data, which is more difficult to compute, while the others expect structured data. This shows that our algorithm can deliver results in timely fashion. To our best knowledge, there is no other system that serves individual farmers with this advanced technique. In addition, other related parties can also benefit from this system.*
Keywords: Fairness, Shapley value, Coalition formation, Agriculture

1. **Introduction.** Applying Artificial Intelligence (AI) research in real world domain has been prosperous recently. It would be very challenging and beneficial to bring about advances in AI research to help farmers in developing countries. The government of Thailand reacts quickly to this prosperity by announcing that AI is one of the country's major policies. Among many initiatives, the Thai government has been working on promoting cooperation among farmers, namely "Plaeng Yai" (Big Farm), so that they can benefit from economy of scale. The Department of Agriculture Extension (DOAE), Ministry of Agriculture, of Thailand is enthusiastic about this. It wants to utilize the immense amount of agricultural data, collected from millions of Thai farmers over three decades, with advances in AI and benefit Thai farmers directly and in the simplest possible way. DOAE develops an ambitious AI-based system, namely, Personalized Data (PD), to realize this idea. PD is equipped with many advances in AI, machine learning and cooperative game theory (also known as *coalition formation*) [1], in particular. Coalition formation is used to help Thai farmers benefit from economy of scale, increasing their negotiation power and decreasing costs. The challenge of the system is to make the information analyzed by advanced AI techniques understandable, accessible and acceptable by farmers.

This paper presents details on how the benefits accruing from coalition formation can be calculated and distributed among farmers fairly. In PD, farmers can form coalitions in various forms, such as group buy. The discount is the benefit and can be distributed among

DOI: 10.24507/icicelb.13.11.1143

themselves. The most well known approach for distributing the benefit fairly is Shapley value [2]. However, the technical problem with computing Shapley value is that it can be intractable for a small size of input and may not be appropriate for directly applying in real world domains. This work proposes an approach to help compute distribution of discounts for farmers fairly and quickly. In order to provide convenient access to farmers, they can use the system via their mobile phones. DOAE has already developed and deployed many mobile phone applications for farmers. This project is a continuation on this basis.

The paper is structured as follows. We review advances in research in coalition formation, with respect to fairness, and its applications in agricultural domains. We then discuss calculation for coalition values and fair payoffs. We also provide an example for it. We then discuss about experiments and results, and then conclude, lastly.

2. **Related Works.** Coalition formaiton has been adopted to help solve real world problems on various aspects. Important solution concepts include efficiency (core [3] and optimal coalition structure [4]), stability (kernel [5]), and fairness (Shapley value). Here we shall review previous works related to Shapley value.

The computation complexity for Shapley value is NP-Hard [6] because it exhaustively computes every contribution of agents in all permutations. This can become intractable very easily. To achieve results in timely fashion, it is typically either estimated or computed with strict assumptions. For the former, most works are in theoretical domain. Corder and Decker [7] cluster coalitions of agents, based on their features and similarities, and structure them in hierarchical order before estimating Shapley value. The execution time rises sharply when the number of agents exceeds 15 agents and up to 20 agents. However, the execution time for estimation is well below 1 second. Their research is in a similar domain with ours and will be used as a benchmark. Maleki et al. [8] estimate Shapley value in cooling load domains, whose objective is to distribute work load on several coolers. Not much advanced technique is really used to compute Shapley value. Despite of this, we can also use as our benchmark. Lupia et al. [9] apply Shapley value in an allocation problem by estimating lower and upper bounds of the exact values. Their work differs from ours because we propose an algorithm that provides exact results. Landinez-Lamadrid et al. [10] extend Shapley value into large supply chain in free trade environment, assuming certain structure of coalitions. Their work differs from ours that we do not expect any existing structure. Fu et al. [11] also extend the concept into supply chain of shell-egg industry in China. They do not present any new algorithm for computing nor estimating the value. It is used to model the behavior of parties in the chain. The computation task is not heavy because of small size of input. Their work proposes nothing new, whereas our work proposes one. Ren et al. [12] come closer to our work that it analyzes profit distribution in agricultural supply chain. Unfortunately, their work focuses mainly on administrative aspect. They do not propose any new algorithm.

There are more works related to Shapley value but cannot be compared with ours. Rozemberczki and Sarkar [13] approximate Shapley value in ensemble game, which requires that agents are pre-trained binary classifier. Although the results are satisfactory, such requirement does not exist in our domain. Li et al. [14] apply Shapley value to decision making situations which can happen in various domains, such as crime judgement, financial analysis, and medical diagnosis. It assumes mining evidence for necessary information. This condition does not hold in our environment. Li et al. [15] divide payoffs fairly based on Shapley value in multi-agent reinforcement learning, where centralized training with decentralized execution has been widely adopted. Unfortunately, it is not applicable to our environment. Touati et al. [16] use Bayesian Monte Carlo for computing Shapley value in weighted voting and bin packing games. Their work expects monotonicity in the coalition members. The algorithm depends on the associated sampling which does not

exist in our domain. Okhrati and Lipani [17] estimate Shapley value by assuming multilinear extension sampling. Again, this sampling is not applicable to our environment. Patel et al. [18] do not really compute Shapley value but rather apply the concept to helping select vocabulary. As we can see, none of these work aims at benefiting individual farmers. In addition, none of them propose algorithm that guarantees exact result, while taking polynomial time.

3. System Overview.

Because of many outside factors involvement, applying a theoretical concept in real world domains, such as agriculture, is very challenging. We discuss such an environment below.

3.1. Real world domain.

DOAE is ambitious but is also pragmatic. Among many possibilities, it wants to focus on a single mobile application to help farmers. Every year, Thai farmers spend hundreds of billions Baht to buy fertilizers and carry them home individually. They receive tiny discounts sometimes. This looks alright from a perspective of an individual farmer. However, it could be a massive loss, from macroscopic perspective of economy, because farmers can collectively save a lot of money by buying in groups, also known as *bulk buy*. Bulk buy can save sellers a fair amount of money in many ways, including shorter (or probably no) stock keeping, more predictable planning for their supply chains, etc. Such cost reduction can be offered to buyers as more discounts. It will be very helpful if farmers can use mobile phones to buy cheaper fertilizers, or other resources, and then share the discounts, as payoffs for farmers, fairly and quickly.

Fortunately, the advances of communication and information technologies nowadays provide us enough foundation to develop a new system for this purpose. In addition, modern shops, suppliers and producers already offer discounts based on economy of scale. Discounts are offered in various forms, such as money value, gifts, and delivery. These sellers normally offer free delivery when the order meets a certain quantity. Sellers deliver the goods to a certain location for each buyer. The new system should allow farmers to define their requirements, e.g., fertilizer description, number of participants, and due time of the group. Farmers participating in the group will be divided into coalitions of buyers, each of which will be represented by a farmer for collecting good from the seller. The coalition representative is a member whose location has the lowest collective cost for other members to come and carry the good back home. The discount for the coalition should be traded off with the cost for collecting them and is the value of the coalition. Lastly, the coalition values and payoffs for farmers should be computed quickly and be distributed fairly among the members.

In such an environment, the followings must be taken into account. 1) *Instant result.* Farmers expect to see results instantly. Waiting time should not exceed 1 minute. In some seasons, the calculation can be background job. Farmers can wait up to the deadline for joining members. The execution on the server can take some time, up to over night. The results will be proposed to farmers once the calculation is done. 2) *Real world data.* In addition to DOAE's databases, there are two outside sources of map data, i.e., Google map and Open street map. The former offers more complete data but costs dearly. The latter is free but offers limited data. Although DOAE has access to Google map, accessing the data in real time is not instant. This limits the size of the group to 20-25 farmers. Also, more members affect the previous point directly. 3) *Pragmatics.* This system is to be used by farmers, whose daily lives are busy and frustrating enough. Many delicate issues in theoretical research may not be taken into account. Kernel, for instance, considers a tiny fraction of a number to decide whether the payoff vector is stable. This obviously is not helpful to farmers. The optimal coalition structure is needed for global efficiency. Fairness, i.e., Shapley value, is needed because this directly benefits farmers and encourages other farmers and sellers to use the system even more.

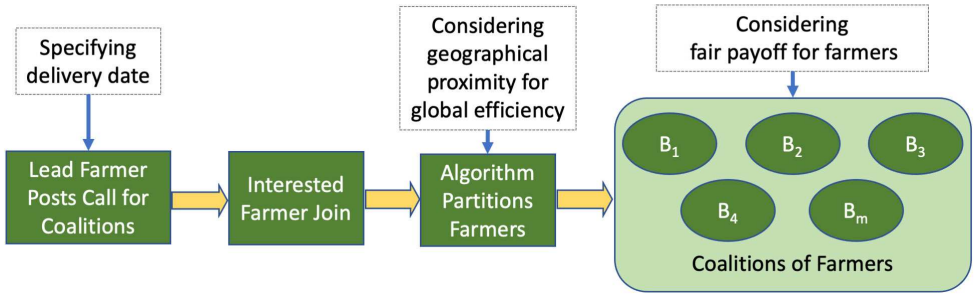

FIGURE 1. Overview process of forming coalition of farmers

3.2. **Process in the system.** PD is a very large system, composed of *i*) machine learning libraries for predicting key factors, including yields, prices and costs of crops, and *ii*) game theoretic, both non-cooperative and cooperative, libraries for strategic analysis and decision making. In this paper, we focus on how farmers can cooperate, i.e., buy in groups. An overview picture of this part of the system is shown in Figure 1. The processes are as the following.

1) A leading farmer posts an invitation for other farmers to participate in the group.
2) Interested farmers join the group up to a certain time or number of farmers.
3) The system calculates coalition values.
4) The system calculates the optimal coalition structure.
5) The system calculates fair payoffs for agents.

The system has been developed and is to be promoted after the COVID-19 pandemic situation is improved. The system is composed of a mobile application for farmers, the configuration system for administrative officers and the dashboard and report for executive officers. In addition, the most important part is the AI engine that receives input data from users, executes important algorithms on the server and outputs the results to users. Due to limited space, only issues that matter the most in the system, i.e., computing coalition values and Shapley values, are to be presented in sections below.

4. **Computing Coalition Values.** Computing coalition values, i.e., the discount accruing from all members, is the first and foremost. The main algorithm, although not presented here, goes through each group, and generates all coalitions. For each of them, it calls the $findCoalitionCenter$ for minimal logistics cost to agents, and the $getCheapestSeller$ for maximal discounted amount. The coalition value can be computed from the relation of *Non-Discounted Amount* (NDA), *Discounted Amount* (DA) and *Logistics Cost* (LC), as the following

$$v(S) = NDA - DA - LC. \tag{1}$$

Here, we present two important algorithms in the system, i.e., the $findCoalitionCenter$ and $getCheapestSeller$, involved in this task.

4.1. **Algorithm to find coalition center.** In practice, ordered goods will be delivered to a customer, who is the representative of the group in our environment. The coalition value depends on the transportation for distributing goods to individual farmers. This algorithm examines for the total distance between each farmer to every other farmer. The farmer whose total distance, $coalDist$, is minimal will be regarded as the coalition center, $coalCenter$. The algorithm firstly takes a *coalition* of buyers, generated from all interested farmers. The total distance from the center to every member, $coalDist$, is set to 0. Then each member b_f of *coalition* is considered being the coalition center from where the goods will be delivered to every member $b_t \neq b_f$ in *coalition*. If the transportation cost, $tranCost(b_f, b_t)$, for carrying the good from b_f back to b_t is more than the cost of buying the good, $baseCost(b_t)$ by b_t, alone, $coalDist$ is set to maximal and returns. This

will prevent the coalition from being in the optimal coalition structure. The *tempDist* is used to accumulate the distance between each pair (b_f, b_t). If the total distance *coalDist* is more than *tempDist*, the new center is found and both *coalCenter* and *coalDist* are set accordingly. Below is the high level algorithm for conducting such task.

Algorithm 1 Algorithms *findCoalitionCenter* for computing coalition center (left) and *getCheapestSeller* for finding the cheapest seller (right)

```
Require: a set of farmers F
  coalCenter ← null
  coalDist ← Int.MAX
  for each b_f ∈ F do
    tempDist ← 0
    tempCenter = b_f
    for each b_t ∈ S, b_t ≠ b_f do
      if tranCost(b_t) > baseCost(b_f) then
        coalDist ← Int.MAX
        return
      end if
      dist ← findDist(b_f, b_t)
      tempDist ← tempDist + dist
    end for
    if coalDist > tempDist then
      coalCenter ← b_k
      coalDist ← tempDist
    end if
  end for
```

(a) Alg. *findCoalitionCenter*

```
Require: a set of sellers S
  bestInitPrice ← Int.MAX
  bestPrice ← 0
  bestSeller ← null
  for each s ∈ S do
    priceRange = s.getPriceRange()
    if priceRange.initPrice < bestInitPrice then
      bestInitPrice ← p.initPrice
      for each p ∈ priceRange do
        if demand < p.quantity then
          if p.price < bestPrice then
            bestPrice ← p.price
            bestSeller ← s
          end if
        end if
      end for
    end if
  end for
```

(b) Alg. *getCheapestSeller*

4.2. **Algorithm to find the cheapest seller.** In practice, sellers may locate anywhere in the country. They try to attract customers by offering their best prices possible. In most cases, the initial offer is to deliver goods with the highest-rate transportation costs. If the ordered quantity reaches the next level, the offer is to reduce the transportation cost rate, up to the point where transportation is free. Then the seller may offer further discounts, or other goods, mostly gold, which can be regarded as further discounts. Given the list of all sellers, S, the algorithm sets the initial values of best initial price, $bestInitPrice$, best price, $bestPrice$, and best seller, $bestSeller$. Then, the algorithm scans each seller, $s \in S$, for the list of price range, $priceRange$. If the initial price, $priceRange.initPrice$, of the range is lower than the current $bestInitPrice$, it further scans each item p in the range. If the demand of buyers is within the quantity limit, $p.quantity$, of the item and the current price, $p.price$, of the item is lower than $bestPrice$, then the best agent, $bestAgent$, and the best price, $bestPrice$, are found and are set accordingly.

Once the coalition values for the group are computed, an algorithm for computing optimal coalition structure is called for optimality and economy of scale. There are many of such well developed algorithms. Therefore, we do not discuss about this process here.

5. Computing Agent Payoffs.

5.1. **Classical Shapley value approach.** Classical Shapley [2] value requires the total sum of marginal contribution of each agent in each of the $n!$ permutations, each of which designates how the grand coalition can be formed. The algorithm, NP-Hard complex, is shown in Algorithm 2, expressing how the total sum of marginal of each agent can be calculated.

Here, we consider a simple example of characteristic function game. Let $N = \{a_1, a_2, a_3\}$. A coaliition, $S \subset N$ is a non-empty subset of N. The set of all coalitions is denoted by **S**. The coalition values are $v(a_1) = 0$, $v(a_2) = 2$, $v(a_3) = 1$, $v(a_1, a_2) = 2$, $v(a_1, a_3) = 3$, $v(a_2, a_3) = 4$, $v(a_1, a_2, a_3) = 5$, respectively. The classical approach firstly

Algorithm 2 Algorithm for computing classical Shapley value (left) and an example to compute it (right)

```
for each p ∈ P do
    S ← ∅
    S′ ← ∅
    for each a_i ∈ p do
        S ← S ⋃ a_i
        cont(a) ← cont(a) + v(S) − v(S′)
        S′ ← S
    end for
end for
```

Permutation	a_1	a_2	a_3
a_1, a_2, a_3	0	2	3
a_1, a_3, a_2	0	2	3
a_2, a_1, a_3	0	2	3
a_2, a_3, a_1	1	2	2
a_3, a_1, a_2	2	2	1
a_3, a_2, a_1	1	3	1
Sum	4	13	13
Payoff	$\frac{4}{6} = 0.67$	$\frac{13}{6} = 2.165$	$\frac{13}{6} = 2.165$

generates all permutations of n agents, representing all possible orders the coalition can be formed. Each permutation strictly specifies how much each agent contributes to the existing coalition by bringing about addition value to the existing one. This can be done by $v(S) - v(S \setminus a_i)$. For each permutation, we compute contribution value n times for all agents. This is not too bad because it is obviously linear. However, there are $n!$ permutations for n agent. This makes computing Shapley intractable very easily. In typical desktop computers, computing Shapley value for 15 agents can take days. As a reference, we compute marginal contribution and agent payoffs according to traditional Shapley [2] as shown in the right part of Algorithm 2. Note that there are $3! \times 3 = 18$ computation for agents' marginal contributions.

5.2. **Our algorithm to compute Shapley value.** While coalition values are known a priori in coalitional game theory, we have to compute coalition values one by one in real world domain. In addition, we have learned in previous section that computing coalition values is time consuming because the distance between each pair of locations of agents is only derived by calling external service, taking account of complexity of real world road network. Since $n!$ is exponential, computing Shapley value can be intractable even for a small number of agents. In practice, computing Shapley value for 20 agents on a typical computer can take a few days to complete the execution. In real world domain, this is not acceptable. In this section we present our approach to rapidly compute Shapley value acceptable to real world applications.

To improve the existing algorithm for computing Shapley value, we do not have to go through all permutations. Given a permutation of coalitions, as we proceed through each S_m, where $1 \leq m \leq n$, we can divide coalitions into three parts: 1) prefix coalition, $|S_{m-1}|$, 2) the coalition, S_m itself, and 3) suffix coalition, $|S_{n-m}|$. Given a coalition S_m, we simply go through each coalition, compute the marginal contribution of each agent in the coalition and multiplied with the permutations of prefix coalition, $|S_{m-1}|$, and the permutations of suffix coalition, $|S_{n-m}|$. To help speed up the calculation and cope with anytime result demand, we rank the coalitions by the agent ratio contribution,

$$r(S) = \frac{v(S)}{|S|}, \tag{2}$$

in ascending order when the coalition value is realized and it is being added to the list.

Given the time limitation enforced by real world demand, we might not have enough time to complete the calculation. However, ranking the coalitions and computing agents marginal contribution help us to have agent payoffs at anytime. The time of computing agents' marginal contribution is

$$\sum_{S \in \mathbf{S}} \sum_{a \in S} (|S| - 1)! \times (|N| - |S|)! \tag{3}$$

5.2.1. *Example of our approach.* Our approach improves the efficiency in computing approximation Shapley value. Firstly, we need to compute the coalition contribution ratio and rank coalitions accordingly in descending order. The ranking is a very simple computation and can be done in polynomial time. The ranking is shown in Table 1.

TABLE 1. Ranking coalitions – Coalitions are ranked based on their contribution ratio

Coalition values	Contribution ratio	Ranking
$v(a_2) = 2$	$r(a_2) = 2$	1st
$v(a_2, a_3) = 4$	$r(a_2, a_3) = \frac{4}{2} = 2$	2nd
$v(a_1, a_2, a_3) = 5$	$r(a_1, a_2, a_3) = \frac{5}{3} = 1.67$	3rd
$v(a_1, a_3) = 3$	$r(a_1, a_3) = \frac{3}{2} = 1.5$	4th
$v(a_3) = 1$	$r(a_3) = 1$	5th
$v(a_1, a_2) = 2$	$r(a_1, a_2) = \frac{2}{2} = 1$	6th
$v(a_1) = 0$	$r(a_1) = 0$	7th

Let $ct(a)$ be the contribution function of agent a. Let $pm(a)$ be the permutation of a set of agent(s) a. We compute the marginal contribution from ranks 1 to 7 below. From top to bottom, we simply follow our approach. The search space can be reduced massively. Hence, the computation time is also reduced. Examples of such computations are shown below.

1st $\{a_2\}$
$ct(b) = 0 + (v(b) - v(\emptyset)) \times pm(\emptyset) \times pm(a, c) = 0 + (2) \times 1 \times 2 = 4$
2nd $\{a_2, a_3\}$
$ct(b) = 4 + (v(b, c) - v(c)) \times pm(c) \times pm(a) = 4 + (4 - 1) \times 1 \times 1 = 4 + 3 = 7$
$ct(c) = 0 + (v(b, c) - v(b)) \times pm(b) \times pm(a) = 0 + (4 - 2) \times 1 \times 1 = 0 + 2 = 2$
3rd $\{a_1, a_2, a_3\}$
$ct(a) = 0 + (v(a, b, c) - v(b, c)) \times pm(b, c) \times pm(\emptyset) = 0 + (5 - 4) \times 2 \times 1 = 0 + 2 = 2$
$ct(b) = 7 + (v(a, b, c) - v(a, c)) \times pm(a, c) \times pm(\emptyset) = 7 + (5 - 3) \times 2 \times 1 = 7 + 4 = 11$
$ct(c) = 2 + (v(a, b, c) - v(a, b)) \times pm(a, b) \times pm(\emptyset) = 2 + (5 - 2) \times 2 \times 1 = 2 + 6 = 8$
4th $\{a_1, a_3\}$
$ct(a) = 2 + (v(a, c) - v(c)) \times pm(c) \times pm(b) = 2 + (3 - 1) \times 1 \times 1 = 2 + 2 = 4$
$ct(c) = 8 + (v(a, c) - v(a)) \times pm(a) \times pm(b) = 8 + (3 - 0) \times 1 \times 1 = 8 + 3 = 11$
5th $\{a_3\}$
$ct(c) = 11 + (v(c) - v(\emptyset)) \times pm(\emptyset) \times pm(a, b) = 11 + (1 - 0) \times 1 \times 2 = 11 + 1 \times 1 \times 2$
$= 11 + 2 = 13$
6th $\{a_1, a_2\}$
$ct(a) = 4 + (v(a, b) - v(b)) \times pm(b) \times pm(c) = 4 + (2 - 2) \times 1 \times 1 = 4 + 0 = 4$
$ct(b) = 11 + (v(a, b) - v(a)) \times pm(a) \times pm(c) = 11 + (2 - 0) \times 1 \times 1 = 11 + 2 = 13$
7th $\{a_1\}$
$ct(a) = 4 + (v(a) - v(\emptyset)) \times pm(\emptyset) \times pm(b, c) = 4 + (0 - 0) \times 1 \times 2 = 4 + 0 = 4$

This algorithm helps compute Shapley value in timely fashion and can be used in real world applications. As in PD, farmers can buy in groups, earn more discount, and use our algorithm to help distribute the discount fairly to farmers.

6. **Experiments and Results.** There are two activities of coalition formation related to this work, i.e., computing the coalition values and fair payoffs to farmers. Since the number of agents participating to a call for a group is not large, merely 20-25 agents,

computing their values can be done relatively quickly provided that information about distances between each pair of farmers is available. The main concern of farmers, and field officers, lies on how much they can save, and, more importantly, they want to know the results quickly. This situation occurs regularly in certain times of a year, particularly in rainy season, in which a lot of farmers would need resources for their farms almost at the same time. Consequentially, there are a lot of computation tasks being requested from computer servers concurrently.

Firstly, we examine how fair the payoffs are, depending on the number of permutations that can be covered. We have shown that our algorithm computes for each agent its marginal contribution in all coalitions, possible prefix coalitions and possible suffix coalitions. For the sake of clarity, we investigate the exact figures for the case of 20 agents. For coalitions with cardinality 1, the numbers of coalitions, prefixes, and suffixes are 20, 1, and 121,645,100,408,832,000, respectively. The number of contributions of all agents in cardinality 1 is 2,432,902,008,176,640,000. For cardinality 2, the number of contributions of all agents is also 2,432,902,008,176,640,000. While cardinality increases up to 20, the numbers of coalitions, prefix and suffix change accordingly. However, the number of contributions of all agents is always 2,432,902,008,176,640,000. When we add up all these figures, the total contribution is $48{,}658{,}040{,}163{,}532{,}800{,}000 = 20! \times 20 = 48{,}658{,}040{,}163{,}532{,}800{,}000$.

Secondly, we examine how quickly our algorithm performs in practice. Due to the NP-Hard complexity of computing Shapley value, most research papers commonly approximate the value in order to achieve the results in timely fashion. Among a number of related papers, there are a handful of papers that can be compared with ours. The first one is Maleki et al. [8]. Unfortunately, it merely covers 15 agents. The second one is Corder and Decker [7], which covers 20 agents. However, both papers assume certain structures in the data that allows for speedy approximation. On the other hand, our algorithm does not assume such structures. It takes, as input, generic coalitions that do not have any structure. Although this is not really a fair comparison, it is the most appropriate one we can achieve. In addition, our algorithm works in a relative tougher environment. Therefore, if our algorithm's performance is close to that of the other two, that should be enough to prove that our algorithm has satisfactory results. Afterall, this algorithm will be used in real world environment and we do not expect perfect result as long as it is reasonably good and acceptable. In most cases, the size of coalitions is around 10. It may go up to 20 but hardly go up to 25 because the accumulated cost will decrease the coalition value.

We experimented on a computer equipped with 2.6GHz 6-Core Intel Core i7, 16GB 2400MHz DDR4 and 1GB HDD. We compare our execution time results with that of [7] and [8] in two categories: the average time (in millisecond: MS) and its order of magnitude (Log-10). As shown in Table 2, the execution times of our algorithm for 5-20 agents are very small. Our algorithm is as fast as the other two for 5 and 6 agents. Our algorithm and Maleki et al. [8] are equally fast but both are faster than Corder and Decker [7] for 7 to 10 agents. For 11-15 agents, our algorithm is faster than the other two. For 16-18 agents, our algorithm is as [7]. For 19-20 agents, our algorithm is slower than [7]. However, our algorithm average time is still well below a second and these two cases are rare. Note that [8] only offers results up to 15 agents. Also, the classical approach is the exhaustive search, according to Shapley definition [2]. In cases where the number of agents is 20 or lower, the execution time is reasonable to farmers, coupled with executions on other parts of servers and network communication. Therefore, our algorithm is applied to real world usage.

We also investigate when the numbers of agents are 21, 22, 23, 24 and 25, it is found that the average times are 1.41×10^3, 3.06×10^3, 6.47×10^3, 1.49×10^4 and 2.72×10^4 milliseconds, respectively. We examine further on how the algorithm behaves in cases

TABLE 2. Execution time of our approach – Our algorithm guarantees exact results in polynomial time in theory, and finishes the calculation very quickly in practice

Number of agents	Execution time							
	Classical approach		**Our approach**		Corder and Decker [7]		Maleki et al. [8]	
	Average (MS)	Log-10	Average (MS)	Log-10	Average (MS)	Log-10	Average (MS)	Log-10
5	1.00×10^2	2	0.10×10^0	0	7.00×10^0	0	1.00×10^0	0
6	1.59×10^2	2	0.10×10^0	0	8.00×10^0	0	1.00×10^0	0
7	1.71×10^2	2	0.10×10^0	0	1.20×10^1	1	1.00×10^0	0
8	1.80×10^2	2	0.10×10^0	0	1.00×10^1	1	1.00×10^0	0
9	3.35×10^2	2	0.10×10^0	0	1.30×10^1	1	1.00×10^0	0
10	2.53×10^3	3	0.10×10^0	0	1.50×10^1	1	1.70×10^1	0
11	2.89×10^4	4	0.50×10^0	0	2.10×10^1	1	2.20×10^1	1
12	3.39×10^5	5	1.00×10^0	0	1.90×10^1	1	1.80×10^1	2
13	1.20×10^7	7	4.00×10^0	0	2.30×10^1	1	1.00×10^3	3
14	1.68×10^8	8	5.00×10^0	0	2.40×10^1	1	2.00×10^4	4
15	2.51×10^9	9	8.50×10^0	0	2.50×10^1	1	2.50×10^4	4
16	4.02×10^{11}	10	1.50×10^1	1	4.50×10^1	1	–	–
17	6.84×10^{11}	11	2.93×10^1	1	4.70×10^1	1	–	–
18	1.23×10^{13}	13	5.67×10^1	1	4.60×10^1	1	–	–
19	2.34×10^{14}	14	1.31×10^2	2	4.80×10^1	1	–	–
20	4.68×10^{15}	15	2.52×10^2	2	6.00×10^1	1	–	–

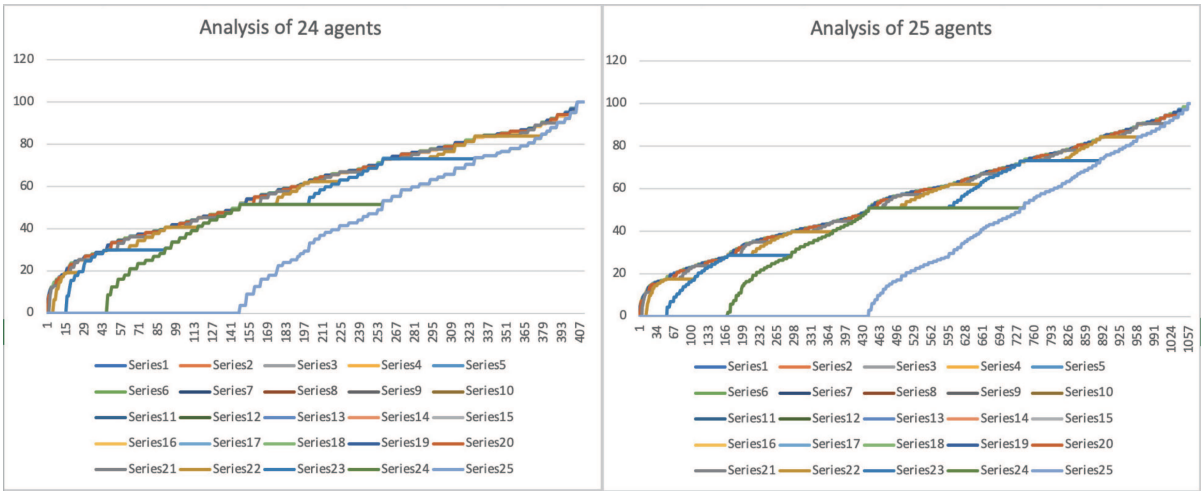

FIGURE 2. (color online) Progresses of contribution values of 24 (left) and 25 (right) agents

of 24 and 25 agents where the execution time goes beyond the criterion of one minute. Instead of following our ranking approach, the algorithm proceeds by following certain order, such as lexicographic. We find that going through the progress of execution of 24 and 25 agents has a certain pattern. Figure 2 shows that there are three agents, whose progresses are similarly slow on both cases. Their progresses are rapidly close to that of other agents when the execution goes pass the last quater of the whole time. This is not really a bad side. It simply tells us that these patterns do not occur in our approach.

We realize that there are limitations of our algorithm. For real-time process, our algorithm can deliver results in timely fashion for 20 agents. For batch process, the highest number of agents we can be assured that our algorithm works pragmatically is 25.

7. **Conclusion.** As part of the "Personalized Data" project developed and deployed at DOAE, Thailand, a mobile application has been developed to help farmers buy in groups for more discounts, and then distribute the benefit fairly. This paper presents algorithms for coalition formation of farmers, used in the project. The main contributions are two folds. Firstly, we propose algorithms that help farmers to form optimal coalition structure of farmers, based on real world data. Secondly, we propose an algorithm to calculate fair distribution of discount rapidly to farmers. The results show that the calculation can be done rapidly and suitable for using in real world agricultural domain. To our best knowledge, there is no other system that serves individual farmers with this advanced technique. In addition, other related parties can also benefit from this system. In the future, theoretical work can be carried out to mathematically prove the efficiency of the algorithm. Furthermore, the relation between different solution concepts should also be explored.

Acknowledgment. This research was supported by The Department of Agricultural Extension (DOAE), Ministry of Agriculture, Thailand. We are grateful for the cooperation and support of DOAE personnel.

REFERENCES

[1] J. P. Kahan and A. Rapoport, *Theories of Coalition Formation*, Psychology Press, New York, 1984.
[2] L. S. Shapley, *Notes on the n-Person Game – II: The Value of an n-Person Game*, RAND Corporation, Santa Monica, Calif., 1951.
[3] D. Gillies, *Some Theorems on n-Person Games*, Master Thesis, Princeton University, 1953.
[4] T. Sandholm, K. Larson, M. Andersson, O. Shehory and F. Tohme, Anytime coalition structure generation with worst case guarantees, *Proc. of the 15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pp.46-53, 1998.
[5] M. Davis and M. Maschler, The kernel of a cooperative game, *Naval Research Logistics Quarterly*, vol.12, no.3, pp.223-259, 1965.
[6] F. Lupia, G. Greco and F. Scarcello, Structural tractability of Shapley and Banzhaf values in allocation games, *Proc. of the 24th International Conference on Artificial Intelligence*, pp.547-553, 2015.
[7] K. Corder and K. Decker, Shapley value approximation with Divisive clustering, *Proc. of International Conference on Machine Learning Applications*, pp.234-239, 2019.
[8] S. Maleki, T. Rahwan, S. Ghosh, A. Malibari, D. Alghazzawi, A. Rogers, H. Beigy and N. Jennings, The Shapley value for a fair division of group discounts for coordinating cooling loads, *PLoS ONE*, pp.1-28, 2020.
[9] F. Lupia, A. Mendicelli, A. Ribichini, F. Scarcello and M. Schaerf, Computing the Shapley value in allocation problems: Approximations and bounds, with an application to the Italian VQR research assessment program, *Journal of Experimental & Theoretical Artificial Intelligence*, Taylor & Francis, no.4, pp.505-524, 2018.
[10] D. C. Landinez-Lamadrid, D. G. Ramirez, D. Neira, K. Armando, P. Negrete and J. Cómbita, Shapley value: Its algorithms and application to supply chains, *INGE CUC*, vol.13, no.1, pp.61-69, 2017.
[11] J. Fu, Z. Luo, P. Chen and X. Shen, Cooperative game of income allocation of shell-egg industry supply chain, *Proc. of the 2019 International Conference on Electronical, Mechanical and Materials Engineering (ICE2ME2019)*, 2019.
[12] X. Y. Ren, Q. Feng, S. Wang and X. Wen, Profit distribution of agricultural supply chain based on Shapley value, *Advance Journal of Food Science and Technology*, vol.7, pp.479-483, 2015.
[13] B. Rozemberczki and R. Sarkar, The Shapley value of classifiers in ensemble games, *Proc. of the 30th ACM International Conference on Information & Knowledge Management*, Association for Computing Machinery, New York, NY, USA, pp.1558-1567, 2021.
[14] J. Li, K. Kuang, L. Li, L. Chen, S. Zhang, J. Shao and J. Xiao, Instance-wise or class-wise? A tale of neighbor Shapley for concept-based explanation, *Proc. of the 29th ACM International Conference on Multimedia*, pp.3664-3672, 2021.
[15] J. Li, K. Kuang, B. Wang, F. Liu, L. Chen, F. Wu and J. Xiao, Shapley counterfactual credits for multi-agent reinforcement learning, *Proc. of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining*, 2021.

[16] S. Touati, M. S. Radjef and L. Sais, A Bayesian Monte Carlo method for computing the Shapley value: Application to weighted voting and bin packing games, *Computers & Operations Research*, vol.125, 2021.

[17] R. Okhrati and A. Lipani, A multilinear sampling algorithm to estimate Shapley values, *Proc. of the 25th International Conference on Pattern Recognition (ICPR)*, Virtual Event/Milan, Italy, pp.7992-7999, 2020.

[18] R. Patel, M. Garnelo, I. Gemp, C. Dyer and Y. Bachrach, Game-theoretic vocabulary selection via the Shapley value and Banzhaf index, *Proc. of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp.2789-2798, 2021.