An algebraic approach for the Shapley value computation in Multidimensional Integer Knapsack Games

Phuoc Hoang Le, Tri-Dung Nguyen, and Tolga Bektaş

Southampton Business School and Centre for Operational Research, Management Science and Information Systems (CORMSIS),
University of Southampton, Southampton, UK, SO17 1BJ
P.H.Le@soton.ac.uk, T.D.Nguyen@soton.ac.uk, T.Bektas@soton.ac.uk

Abstract. The combinatorial optimization game where the characteristic function is generated by solving a combinatorial optimization problem forms an important class in cooperative game theory. In this paper, we introduce a new subclass of combinatorial optimization games namely the multidimensional integer knapsack game (MIKG). Finding the Shapley value as the solution for the payoff allocation problem is challenging and time consuming for MIKG because of the integer program structure of the characteristic function. We propose to utilize an algebraic approach of Gröbner bases for solving the integer programs, and generate an efficient way to find the Shapley value in these cooperative games. Some computational experiments with the MIKG are investigated where the numerical results show the advantages of the method compared to CPLEX solver.

Keywords: Combinatorial optimization games, multidimensional knapsack problem, cooperative games, Shapley value, fairness.

1 Introduction

One of the main focus of cooperative game theory is the problem of payoff distribution (or cost allocation) among the participants in a joint venture. When the players collaborate in groups, they will share their resources and work together to optimize the outcome, hence a cooperative game model is linked to this underlying optimization problem. If the characteristic function of the cooperative games has supper-additive property, there are possibilities to create more value for each player than what they can work separately on their own. In these situations, we need to figure out ways to allocate the total cost or the revenue to the contributed players.

Let $N = \{1, 2, \dots, n\}$ be the set of players and 2^N be the set of all subsets S of N. A cooperative game with transferable utility is described by a pair (N, ν), where a characteristic function $\nu : 2^N \to \mathbb{R}$ assigns to each coalition $S \subset N$ of players a real value $\nu(S)$, representing the maximum value that the members of this group can achieve when they cooperate. An imputation vector $\boldsymbol{x} \in \mathbb{R}^n$ of the game (N, ν) is defined as the set of efficient and individually rational allocation of $\nu(N)$, i.e. $\sum\limits_{i \in N} x_i = \nu(N)$, and $x_i \geq \nu(i), \forall i \in S$.

In cooperative game theory, there are many solution concepts for allocating cost/payoff such as the core, the Shapley value and the nucleolus. However, the Shapley value is the only solution concept satisfying all four basic axioms of efficiency, symmetry, dummy and additivity (Shapley(1953)). For many games with a small number of players, the typical method to find the Shapley value is to employ the exact formula directly. In cases of the games with large number of the players, the exact method need to solve the value of the characteristic function 2^n times. Hence, when the coalition values problem is NP-hard, the computational time for the Shapley value is exponential large (for example, see, Deng & Papadimitriou (1994)).

In this paper, we will introduce a new class of cooperative games called Multidimensional Integer Knapsack Game. As a special class of combinatorial optimization games, it is also a type of homogeneous integer maximization games, which will be presented in the next chapter. Another version of the knapsack games called the Knapsack Budgeted Games has been investigated in Bhagat et al.(2014). Compared to our proposed games, this type of game is a binary knapsack game and the value of a coalition S of players is determined by a critical subset $T \subset S$ due to a budget constraint which restricts the size of T. We will later show the connection of our MIKG with the skill games in the literature and some practical applications of the MIKG model will be presented.

We will also proposed an algebraic approach of Gröbner bases to find the Shapley value of the MIKG. The algebraic Gröbner approach, recently known as a new approach to solve integer program (for examples, Conti & Traverso (1991) and Rekha Thomas(1995)). In the case of the number of players in MIKG is large than 20, the Monte Carlo sampling-based technique is applied to reduce the number of marginal contributions that we have to compute in order to find the Shapley value. Therefore, the main contributions of our work include:

- We introduce a new class of cooperative games named the Multidimensional Knapsack Games. The connection of MIKG with other games in the literature and some practical applications are given.
- The algebraic approach of Gröbner bases is proposed to apply in solving the integer optimization problem and then find the exact Shapley value of MIKG for small scale problem.
- For the class of MIKG with large number of players ($20 \leq n \leq 100$), we combined the algebraic approach with the sampling techniques to approximate Shapley value.
- Finally, the computational experiments show the effectiveness of the proposed methods compared to CPLEX solver.

The paper is organized as follows: In section 2 the preliminaries give some background about the combinatorial optimization games, the Shapley value and the Gröbner method to solve Integer Program. The section 3 formalizes the formulation of the MIKG, its applications and presents some properties of its

Shapley value. In section 4, we use an augmentation algorithm with the Gröbner basis to propose a framework to find the exact Shapley value for small scale games and approximate Shapley value for large scale games. Section 5 contains some computational experiment and shows the numerical results for different problem instances. In section 6, the conclusion and some further discussions are given.

2 Preliminaries

2.1 Combinatorial Optimization Games

In this section, we will review a class of cooperative games called combinatorial optimization games for which the game characteristic function arises from various combinatorial optimization problems. Formally, the value (or the cost) of each coalition S is defined as the objective outcome of a combinatorial optimization when the subset S of players decide the underlying combinatorial structure. In the direction of development, we now present some of general subclasses of combinatorial optimization games including the linear production games, packing games and integer optimization games. These generalized games will give us a broad sense of the connection between mathematical optimization and cooperative game theory.

The covering/packing game models are described in Deng et al.(1999) for some discrete optimization problems, where the class of cooperative games with the characteristic function value defined by an integer program:

$$\nu(S) := \max \ \{\boldsymbol{c}^T \boldsymbol{x} \ \mid \ \mathrm{A}\boldsymbol{x} \leq \boldsymbol{b}, \ \boldsymbol{x} \in \{0,1\}^q\},$$

where the matrix A is a $\{0,1\}$-matrix and $\boldsymbol{b}$. This subclass of the combinatorial optimization game includes many classic type of games such as the covering games, max-flow game, packing games and minimal colouring games.

More recent generalization of the combinatorial optimization games is defined as Integer Minimization game (IMG) in Caprara & Letchford(2010). The paper defined the total cost of each group coalition as the optimal value of an Integer Linear Program of the following form

$$c(S) := \min\{\boldsymbol{c}^T \boldsymbol{x} \ \ : \ \ \mathrm{A}\boldsymbol{x} \geq \mathrm{B}\,\boldsymbol{y}(S) + \boldsymbol{d}, \ \boldsymbol{x} \in \mathbb{Z}^q_+\},$$

where $\boldsymbol{y}(S)$ is the incidence vector of subset $S \subset N$, $\boldsymbol{d} \in \mathbb{Z}^r$ and A, B are integer matrices of dimension (r, q) and (r, n). Note that all combinatorial games can be formulated as an integer minimization game by introducing maximum one variable for each coalition. The authors also considered the facility location, travelling salesman and vehicle routing games, which have been formulated as homogeneous IMG, and are therefore supper-additive.

The similar version of the Integer Minimization Game, where the cooperative games are in the value form, is Integer Maximization Game (IMG). Using the same notation of $(\mathrm{A}, \mathrm{B}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{y}(S))$, the value function of a coalition S has the form:

$$\nu(S) := \max\{\boldsymbol{c}^T\boldsymbol{x} \;\;:\;\; \mathrm{A}\boldsymbol{x} \le \mathrm{B}\boldsymbol{y}(S) - \boldsymbol{d}, \boldsymbol{x} \in \mathbb{Z}_+^r\}.$$

The supper-additive property of the integer maximization games can be deduced when the condition of $\boldsymbol{d} \ge 0$ is satisfied, i.e., for any two disjoint coalitions S and S', we have $\nu(S \cup S') \ge \nu(S) + \nu(S')$. In this case, it is always beneficial for the players to work together in a grand coalition.

For the application survey of this research area, we refer the readers to the review paper of Borm et al.(2001). In the paper, the interplay between optimization and allocation is shown in various applications in the literature such as game on graphs, travelling salesman game, vehicle routing game, facility location game, and network design game among others. There are many application of this classes of cooperative games which related to our proposed MIKG.

2.2 Shapley value

The Shapley value is the most well-known solution concept based on the notion that the allocation for player i should be proportional to that player's power in the game, i.e how much value, player i creates. The Shapley value is the popular way to divide the total gains/costs to the players, assuming that they all collaborate. The formula of Shapley value of a player i is the weighted average of all marginal contribution of that player for all coalition $S \subseteq N \setminus \{i\}$.

Formally, The *Shapley value* of a cooperative game $G = (N, \nu)$ is the allocation of payoff where

$$\mathrm{Sh}_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} [\nu(S \cup \{i\}) - \nu(S)]. \tag{1}$$

Another way to formulate the Shapley value is mentioned as follow:

$$Sh_i = \frac{1}{n!} \sum_{\pi \in \Pi(N)} [\nu(\mathrm{Pre}^i(\pi) \cup \{i\}) - \nu(\mathrm{Pre}^i(\pi))]. \tag{2}$$

For the convenience of notation, let us denote $\nu^i(S) := \nu(S \cup \{i\}) - \nu(S))$ as the contribution value of player i to the coalition S. Due to four special properties of efficiency, symmetry, dummy and additivity, the Shapley value become very popular in both game theory literature and in the practice. We will then focus on the computation of the Shapley value in this paper.

2.3 Algebraic Gröbener method for Integer Programs

Consider an integer linear programming problem $ILP(\boldsymbol{b})$ in problem (6) with right hand side (rhs) vector $\boldsymbol{b} := \boldsymbol{b}(S)$ for short notation.

$$\begin{aligned} \mathrm{ILP}(\boldsymbol{b}) := \;\; \min \;\; & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \;\; & \mathrm{A}\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \in \mathbb{Z}_+^m, \end{aligned} \tag{3}$$

where $\mathrm{A} \in \mathbb{Z}^{r \times m}$, $\boldsymbol{b} \in \mathbb{Z}^{r}$, $\boldsymbol{c} \in \mathbb{R}^{m}$. The notation ILP($\boldsymbol{b}$) denotes the integer linear programming problem with right-hand-side fixed to $\boldsymbol{b}$. The notation ILP denotes the set of all the integer linear programming problems obtained by varying the right-hand-side vector $\boldsymbol{b}$, but fixing A and the cost function $\boldsymbol{c}$.

Several techniques for solving these problems exist (see, for example the book of Wolsey & Nemhauser) and in this part we investigate one such solution method. A natural approach to solving an integer program is to devise a method to search in the neighbourhood of a given solution for another solution with an improved cost value. If such a solution is found, we will update the current solution with the improved solution and repeat the search. In order to make this search well defined, we define the neighbours of a solution to be all the feasible solutions obtained by adding a finite set of vectors to the current solution. We call such a finite set of vectors a test set for the integer program if each non-optimal solution has at least one neighbour (obtained as above) with an improved cost value. If this test set exists, the program can be solved to optimality provided an initial solution is known. In this section, we establish the existence of a test set for an entire family of integer programs to which the given program belongs and present an algorithm for its construction. This gives an algorithm for solving integer programs. In a companion paper (Tayur et al. (1995)), the authors compute these test sets and the exact optimal solutions for a family of chance constrained integer programs that arose in a manufacturing setting. In this case, they were able to exploit the structure of these integer programs and certain features of the algorithm used to compute test sets, to considerably speed up the computation.

An important concept in polynomial ideal theory is that of a Gröbner basis (Buchberger(1985)) for a given polynomial ideal with respect to some fixed term order. Gröbner bases are special generators of polynomial ideals which can be computed using Buchberger's algorithm (Buchberger(1985)). The paper of Conti & Traverso (1991) is the first describe an algebraic method to solve integer programs of the above form, using ideas from Gröbner basis theory and commutative algebra. A geometric interpretation of the Conti-Traverso algorithm provides a unique minimal test set for integer programs of the previous form as $\boldsymbol{b}$ varies. Denote this family of programs by $IP(A, c)$. We call this test set the reduced Gröbner basis of $IP(A, c)$ due to its algebraic origins. A geometric version of the Buchberger algorithm follows from the above interpretation. Since Gröbner bases of polynomial ideals can be computed in practice, these test sets can be generated using a computer algebra package like 4ti2 (4ti2). In this paper, we will call this method for solving Integer Programming as the Algerbaic Gröbner (AG) method.

The computation of these Gröbner bases depends significantly on the Buchberger's algorithm. In the literature, there are a few methods to implement the Buchberger's algorithm (for example, see Hemmecke & Malkin(2009)). Among those, the Project and Lift algorithm is known as the best compared to others such as BLR algorithm and Saturation method . In general, as this method can be applied to solve system of equations, hence computing the Gröbner bases is

NP-hard problem. However there are some special cases depending on the structure of the matrix A, this process can be computed in polynomial time in input parameters (Loera et al.(2013)).

In the literature, there are a few application of the AG methods to linear (and non-linear) Integer Programs. Hemmecke et al. (2011) show a few linear integer program applications in practice such as the congestion-avoiding routing and the error-correcting codes problems. Two other non-linear examples are the minimization of the cost of scheduling jobs on parallel machines given restrictions on demands and capacity (Gago et al.(2013)) and the minimization of the cost in the series parallel redundancy allocation problem given a target reliability (Gago et al.(2015)). This algebraic approach is also implemented for the portfolio optimization problem by solving a dual search strategy using the Gröbner basis considered as test sets (Castro et al.(2011)). For some special classes such as the transportation problems and network flow problems, the Gröbner bases can be calculated in polynomial time w.r.t the input. More details about the theory of using algebraic and geometric method in discrete optimization can be found in the book of Loera et al.(2013).

3 Multidimensional Integer Knapsack Game (MIKG)

We now introduce the Multidimensional Integer Knapsack Game as a typical example of the homogeneous class of Integer Maximization Game. It can also be derived from the class of linear production games when we restrict the decision variables to be integer. There is a strong connection of this class of cooperative game with the skill games in the literature. Many practical versions of MIKG inspired from the applications of the MIKP such as the bin packing, cargo loading, cutting stock problems, processors allocation in huge distributed system, and capital budgeting.

A multi-dimensional knapsack game (N, ν) is a cooperative game with the set of players $N = \{1, \dots, n\}$ and each player i has a vector of resource capacities $\{b_i^k\}_{k=1}^r$. Each coalition of players S can choose a set of items in $J = \{1, \dots, q\}$ and there is no upper bound on the number of copies for each kind of item. Let denote $\boldsymbol{p} = \{p_j\}_{j=1}^q$, where p_j is the value of the item j, and the weight of that knapsack item j is given by a d-dimensional integer vector $\boldsymbol{w}_j = (w_{1j}, \dots, w_{rj})$. The players in the coalition aim to maximize the total value, however they have to satisfy all knapsack constraints. The MIKG characteristic function can be defined by the following Integer Linear Program:

$$\begin{aligned} \nu(S) := \max \ & \sum_{j=1}^{q} p_j z_j \\ \text{s.t.} \ & \sum_{j=1}^{q} w_{kj} z_j \le b^k(S); \quad k = 1, \dots, r \\ & z_j \in \mathbb{Z}_+ \qquad\qquad j = 1, \dots, q, \end{aligned} \tag{4}$$

From the definition of the MIKG, we can transform the characteristic function into standard form with new slack variable vector $\boldsymbol{y}$. Therefore the problem (4) can be written as:

$$\nu(S) := \max \{\boldsymbol{p}^T\boldsymbol{z} : \mathrm{W}\boldsymbol{z} + \mathrm{I}_r\boldsymbol{y} = \boldsymbol{b}(S),\, \boldsymbol{z} \in \mathbb{Z}_+^q, \boldsymbol{y} \in \mathbb{Z}_+^r\}, \tag{5}$$

or equivalently,

$$\nu(S) := \max \{\boldsymbol{c}^T\boldsymbol{x} : \mathrm{A}\boldsymbol{x} = \boldsymbol{b}(S),\, \boldsymbol{x} \in \mathbb{Z}_+^{q+r}\}, \tag{6}$$

where the matrix $\mathrm{A} = [\mathrm{W}|\mathrm{I}_r]$, the vectors $\boldsymbol{x} = [\boldsymbol{z}, \boldsymbol{y}]$ and $\boldsymbol{c} = [\boldsymbol{p}, 0_r]$ of size $m = q + r$. The 0_r is the zeros vector of length r and I_r is the identity matrix of size r. In the formulation, $\boldsymbol{b}(S) := \{\boldsymbol{b}^k(S)\}_{k=1}^r$ is the resource vector of the coalition S and the total capacity of resource k of the coalition S is defined by $\boldsymbol{b}^k(S) := \sum\limits_{i \in S} b_i^k$.

3.1 Applications of MIKG and related classes of cooperative games

The purpose of this section is to propose some applications for the multidimensional integer knapsack game (MDKG). The general multidimensional integer knapsack problem and its variants have been widely used to model many practical problems such as project selection, capital budgeting, combinatorial auctions , and inventory allocation in assemble-to-order systems, among others. We will now present some similar applications for the MIKG as follows:

- The integer production games, where the underlying optimization problem is similar to the Linear production games, is one typical example. When each player has a different resource vector and the outcomes of the production process must have integer value (i.e. there are integer products of each type to produce), our MIKG will be the generalized version of the LPG.
- In capital budgeting and project selection applications, if there are a set of investors with different type of capitals who want to collaborate for investment. The problem is a different version of the MIKG with the solutions are the binary vector.
- Allocation of databases and processors in a distributed data processing is another application of the MIKG when there are a list of players who want to collaborate for better outcome.

The MIKG have some similarities with the coalitional skill games (CSG) (Bachrach & Rosenschein (2008)). Both have different resources with which players are endowed with the input resources in MIKG and the agent skills in CSG. Similarly, there are outputs that coalitions of agents strive to achieve finished goods in MIKG, and completed tasks in CSGs. Finally, the market prices of finished goods in MIKG have the similar meaning with the weights of tasks in WTSG. The key difference between MIKG and CSG is that the input resources have no quantities in the CSG.

As the resources in CSGs are skills, they can be used as many times as required. In contrast, quantities stay as the important part of MIKG, where the amounts of the finished goods a coalition can manufacture depends on the amounts of input resources a coalition has, and the amount required to produce each finished good. Most of the previous research about coalitional skill games utilized the core solution concept instead of the Shapley value.

3.2 Analytical properties of the Shapley value of the MIKG

The multidimensional knapsack problem has attained a lot of attention in the literature. Although the classical knapsack problem can be solved in pseudo-polynomial time using dynamic programming, the multidimensional knapsack problem (MIKP) is NP-hard, especially when the dimension is getting larger (Bertsimas & Demir(2002), Puchinger et al.(2010)). Hence, solving 2^n times the multidimensional knapsack problem as the outcome of the MIKG characteristic function to find the Shapley value could be time consuming.

In the paper of Sturmfels & Thomas (1997), the authors investigate the AG method for solving Integer program when the cost vector of the objective function changed. Two cost functions c and c' are considered equivalent if they give the same optimal solutions for each $\boldsymbol{b}$. They construct a polytope $St(A)$ whose normal cones are the equivalence classes. Explicit inequality presentations of these cones are given by the reduced Gröbner bases associated with A. The union of the reduced Gröbner bases as c varies (called the universal Gröbner basis) consists precisely of the edge directions of St(A). They present the geometric algorithms for computing St(A) and the universal Gröbner basis.

We can extend this result for our problem of calculating Shapley value. For we change the cost functions c to c', which belong to the same Gröbner cone, the Shapley value of each player i will change proportionally w.r.t. these cost vectors.

$$\begin{aligned}\phi_i &= \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n-|S|-1)!}{n!} \left[\nu(S \cup \{i\}) - \nu(S)\right] \\ &= \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n-|S|-1)!}{n!} \left[\boldsymbol{c}^T\{\boldsymbol{x}(S \cup \{i\})\} - \boldsymbol{c}^T\{\boldsymbol{x}(S)\}\right] \\ &= \boldsymbol{c}^T \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n-|S|-1)!}{n!} \left[\boldsymbol{x}(S \cup \{i\}) - \boldsymbol{x}(S)\right] \\ &= \boldsymbol{c}^T \bar{\boldsymbol{x}}_i\end{aligned} \tag{7}$$

In the process of finding the Shapley value of ϕ_i we need to compute the value of vector $\bar{\boldsymbol{x}}_i$. Hence we could use this vector to find the Shapley value of player i for all the cases when c' belong to the same Gröbner cone of the cost function c.

4 Algebraic Gröbner Approach for Shapley value of MIKG

4.1 Framework for finding the Shapley value of IP games

In this section, we will show the way to incorporate the algebraic Gröbner approach to calculate the Shapley value. For finding the Shapley value, the algebraic approach will compute the Gröbner bases at the beginning as a fixed cost and then employ the augmentation algorithm to solve $\nu(S)$ with different resource vectors $\boldsymbol{b}(S)$. We notice that this approach will solve the IP problem substantially faster by by changing of the rhs in ILP problem (6) compared to other method when the Gröbner basis can be computed effectively. Afterwards, by repeating this process for 2^n coalition values, these variable costs of each IP evaluations will bring the total computational time of the Shapley value.

The Shapley value is based on the notion that the allocation for player i should be proportional to that player's contribution to the game, i.e., how much value, player i creates. The formula of the Shapley value of a player i is the weighted average of all marginal contribution $\nu(S \cup \{i\}) - \nu(S))$ of that player for all coalition $S \subseteq N \setminus \{i\}$, hence we can apply the AG approach as follows

Algorithm 1 : Shapley value - Algebraic Gröbner Exact Algorithm

```
Input: Problem instance $(A, B, c)$ with size $(r, q, n)$.
Output: Exact Shapley value for each player $i$.
Find the Gröbner basis of matrix $A$ and the term order $>_{\boldsymbol{c}}$.
Build the Integer Program model with the rhs $y(S)$ in problem (4).
for $\forall S \subset N$ do
  Compute the value of $\nu(S)$ by AG method with the Rhs $\boldsymbol{b}(S)$
end for
for i =1:n do
  Calculate the Shapley value $\boldsymbol{\phi}(i)$
end for
```

For the game with a small number of players, i.e. less than 20, we can find the real Shapley value using the exact formula. This process requires the evaluation of 2^n values of the coalition, each of which is an integer optimization problem. However, when the number of players increases we will apply the sampling-based technique to approximate the Shapley value with a limited number of samples.

4.2 Sampling technique for approximating the Shapley value

The sampling-based technique is applied and shown the effect when we approximate the Shapley value for large-scale MIKG. In these games, instead of evaluate all 2^n integer program, we can utilize the sampling-based technique to limit a fixed number of samples which we will solve the IP problem to find the coalition values. The number of samples taken to approximate the Shapley value depends on the level of exactness in required time. Moreover, we will combine the Shapley value approximation and the algebraic Gröbner approach as in the following algorithm:

Algorithm 2 : Shapley value - Algebraic Gröbner Sampling Algorithm

```
Input: Problem instance $(A, B, c)$ with size $(r, q, n)$.
Output: Approximated Shapley value for each player $i$.
First, calculate Gröbner basis of matrix $A$ and the term order $>_{\boldsymbol{c}}$.
Let $\{N_k\}_{k=0}^{n-1}$ be the sample size of stratum $k$;
Build the Integer Program model with the rhs $y(S)$ in problem (4).
for $k = 1 : n$ do
  for $j = 1 : N_k$ do
    Randomly generate a coalition $S$ of size $k$
    Solving the Integer Linear Program by AG method with the Rhs $\boldsymbol{b}(S)$.
  end for
end for
```

The algebraic Gröbner approach required us to find the Gröbner bases for our problems in the beginning as fixed cost. However, the step of augmentation algorithm is quicker than the typical ILP solver. If we have to solve the Integer Programming model many times for a specific problem model with changed rhs, this process with smaller variable costs will bring down the total calculation time in the long run.

4.3 Augmentation algorithm for integer program problem

This section describes the aumentation algorithm in the algebraic Gröbner (AG) approach to solve Integer Programming problems, after we have the Gröbner basis for the model using 4ti2 software. Augmentation algorithm is similar to the simplex method in the sense that the objective values improve through each iteration of the algorithms. Moreover, we can also apply the modified augmentation algorithm to find a feasible solution for the integer programming problem. We will show the augmentation algorithm (Loera et al.(2013)) in the details below:

Thus, the process of finding an optimal solution to ILP($\boldsymbol{b}$) via test sets can be decomposed into the following steps:

1. Find an initial feasible solution $\boldsymbol{x}_0$ of ILP($\boldsymbol{b}$).
2. Decide whether $\boldsymbol{x}_0$ is optimal, and
3. If $\boldsymbol{x}_0$ is not optimal, find a better feasible solution.

Often, several test set vectors could serve as a possible augmenting direction for a given non-optimal solution $\boldsymbol{x}_0$. Therefore, the question arises whether the augmenting vectors can be chosen in such a way that only a small number of augmentation steps are needed in order to reach an optimal solution.

The algorithm has an assumption that we can use the Gröbner basis to find out the initial feasible solution. This is a non-trivial problem in general, however,

Algorithm 3 : Augmentation Algorithm with the rhs vector $\boldsymbol{b}$

```
Input: $A, \boldsymbol{b}, \boldsymbol{c}$ where A is the matrix, $\boldsymbol{b}$ is rhs, and vector $\boldsymbol{c}$ give a comparison order,
A finite test set $G_{A,<_{\boldsymbol{c}}}$ of the ILP(6) (using 4ti2 software),
A feasible solution $\boldsymbol{x}_0$ to ILP($\boldsymbol{b}$) (solving a matrix decomposition problem).
Output: an optimum $\boldsymbol{x}^*$ of ILP($\boldsymbol{b}$)
Initialize: $check = 1,\ f_0 = 0$;
while $check = 1$ do
    Let $f_s = f_0$;
    for $\forall \boldsymbol{t} \in G_{A,<_{\boldsymbol{c}}}$ do
        Find $\lambda_{\boldsymbol{t}} = \operatorname{argmax}_{\lambda \in \mathbb{Z}} f(\boldsymbol{x}_0 + \lambda \boldsymbol{t})$ such that $\boldsymbol{x}_0 + \lambda \boldsymbol{t}$ feasible.
        Set $f_t = f(\boldsymbol{x}_0 + \boldsymbol{t}\lambda_{\boldsymbol{t}})$
    end for
    Compute $(\boldsymbol{t}_e, \lambda_e) = \operatorname{argmax}_{\boldsymbol{t} \in G_{A,<_{\boldsymbol{c}}}} f_{\boldsymbol{t}}$; and set $\boldsymbol{x}_0 = \boldsymbol{x}_0 + \boldsymbol{t}_e\lambda_e,\ f_e = f(\boldsymbol{x}_0)$
    if $f_e = f_s$ then
        $check = 0$;
    end if
end while
return $\boldsymbol{x}^* = \boldsymbol{x}_0$
```

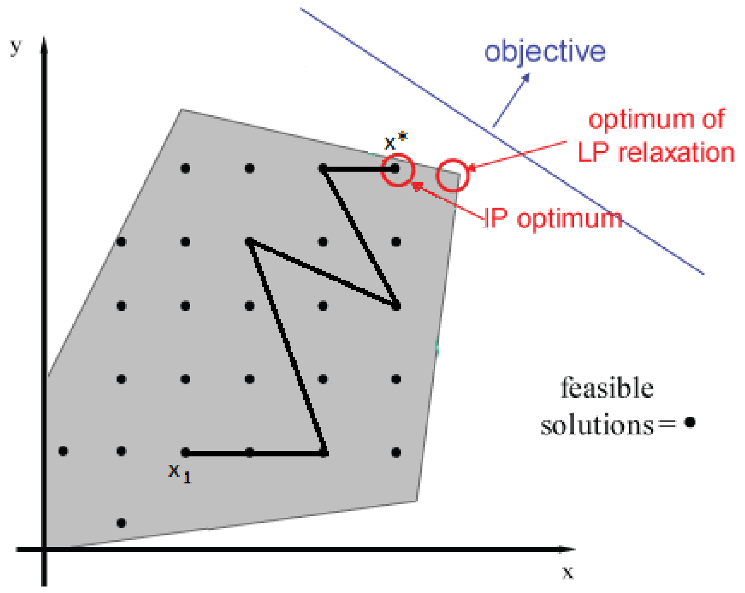

Fig. 1. Augmentation algorithm for an example of Integer Program

note that in the case of multidimensional knapsack games, the zeros vector is one feasible solution. The next step will replace $\boldsymbol{x}_0$ by $\boldsymbol{x}_0 + \lambda t$ and repeat until an feasible solution is attained. It is worth to notice that in the implementation of the augmentation algorithm, we have two decisions to make:

- The first decision is the way to find the suitable vector $t \in G_{\mathrm{A},<_{\mathbf{c}}}$, which we can utilize search algorithms such as Depth-first search (DFS) or Breath-first search (BFS). The effectiveness of the searching strategy will depend on the particular problem instance. However, we apply the BFS and check all the possible vector in the Gröbner basis to find the best solution in the algorithm 4.3.
- The second choice is how we can search the integer value of $\boldsymbol{\alpha}$ to minimize $\|(\boldsymbol{x}_0 + \lambda t)^{-}\|_1$ such that the point $\boldsymbol{x}_0 + \lambda t$ is feasible. In here, the notation $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$ if $\boldsymbol{x} = \{x_i\}_{i=1}^{n}$ and $(\boldsymbol{x})^{-} := min\{\boldsymbol{x}, 0\}$. For this step, we solve an unconstrained optimization problem with one integer variable $\lambda \in \mathbb{Z}$ by a simple heuristic method.

5 Computational Experiment

The algorithms have been implemented in Matlab and run on an Intel-core i5 PC with 2.6 GHz CPU and 4GB RAM. In the experiments, we utilize the program 4ti2 to compute the Gröbner bases. After we show the Gröbner basis computational time for some test instances, we will check our AG algorithm into those instances. We will test the algorithm 1 with the small-scale instance where the number of players less than 20, and the algorithm 2 otherwise.

First, we will shows the computational time and the numbers of vectors in the Gröbner bases for some ILP test instances. The data in the test problems has been randomly generated from uniform distribution for the integral matrices W of various sizes (ranging from 4×6 to 11×15) and the matrices' entries of magnitude from 0 to k, where $k = 100, 20, 10, 1$. The linear cost function has the coefficient $c_{ij} \in [0, 50]$, and the entries of integer matrix B are in the range from 0 to 200. For example, the test instance mat$r \times q -$ e$\{k\}$ shows the calculation time of the Gröbner basis for a knapsack problem with q objects in dimension r, such that the entries in the underlying matrix are generated from the set $\{0, 1, \ldots, k\}$. The column N_{GB} shows the number of vectors in the Gröbner basis and the T_{GB} present the computational time of these bases. The table 1 shows that the growth of the computational time and the size of the Gröbner bases increase quickly for larger instances using the 4ti2 software.

The table indicates the Gröbner bases can be calculated under 5 seconds for the size of matrix W less than 11×13. When the problem size get larger, the total time of solving an ILP problem is the sum of the fixed cost of computing the Gröbner bases and the variable cost of implementing the augmentation algorithm. Therefore, if we solve only the ILP once or a few times, it may seem not worth the effort to invest in the computation of Gröbner bases. However, in the Shapley value formula, we need to solve ILP for a large number of times with the fixed underlying structure excepts the varied rhs. We also notice that these

Table 1. The computational times and sizes of Gröbner bases for different MIK instances

$TestCases$	N_{GB}	T_{GB}	$TestCases$	N_{GB}	T_{GB}
mat4 $\times$ 6 $-$ $e100$	80	0	mat5 $\times$ 7 $-$ $e100$	84	0
mat4 $\times$ 6 $-$ $e20$	65	0	mat5 $\times$ 7 $-$ $e20$	80	0
mat4 $\times$ 6 $-$ $e10$	55	0	mat5 $\times$ 7 $-$ $e10$	53	0
mat4 $\times$ 6 $-$ $e1$	6	0	mat5 $\times$ 7 $-$ $e1$	8	0
mat4 $\times$ 8 $-$ $e100$	238	0.01	mat5 $\times$ 9 $-$ $e100$	422	0
mat4 $\times$ 8 $-$ $e20$	175	0	mat5 $\times$ 9 $-$ $e20$	357	0.01
mat4 $\times$ 8 $-$ $e10$	127	0	mat5 $\times$ 9 $-$ $e10$	259	0
mat4 $\times$ 8 $-$ $e1$	8	0	mat5 $\times$ 9 $-$ $e1$	12	0
mat6 $\times$ 8 $-$ $e100$	210	0.01	mat7 $\times$ 9 $-$ $e100$	319	0.01
mat6 $\times$ 8 $-$ $e20$	126	0	mat7 $\times$ 9 $-$ $e20$	246	0.01
mat6 $\times$ 8 $-$ $e10$	68	0	mat7 $\times$ 9 $-$ $e10$	155	0
mat6 $\times$ 8 $-$ $e1$	10	0	mat7 $\times$ 9 $-$ $e1$	9	0
mat6 $\times$ 10 $-$ $e100$	216	0.01	mat7 $\times$ 11 $-$ $e100$	440	0
mat6 $\times$ 10 $-$ $e20$	47	0.01	mat7 $\times$ 11 $-$ $e20$	362	0.01
mat6 $\times$ 10 $-$ $e10$	37	0	mat7 $\times$ 11 $-$ $e10$	317	0
mat6 $\times$ 10 $-$ $e1$	10	0	mat7 $\times$ 11 $-$ $e1$	13	0
mat8 $\times$ 10 $-$ $e100$	1840	0.19	mat9 $\times$ 11 $-$ $e100$	7219	2.38
mat8 $\times$ 10 $-$ $e20$	1405	0.14	mat9 $\times$ 11 $-$ $e20$	5085	1.36
mat8 $\times$ 10 $-$ $e10$	755	0.03	mat9 $\times$ 11 $-$ $e10$	3816	0.92
mat8 $\times$ 10 $-$ $e1$	16	0	mat9 $\times$ 11 $-$ $e1$	37	0
mat8 $\times$ 12 $-$ $e100$	3883	0.73	mat9 $\times$ 13 $-$ $e100$	3463	0.47
mat8 $\times$ 12 $-$ $e20$	2387	0.34	mat9 $\times$ 13 $-$ $e20$	2751	0.38
mat8 $\times$ 12 $-$ $e10$	1212	0.11	mat9 $\times$ 13 $-$ $e10$	1593	0.12
mat8 $\times$ 12 $-$ $e1$	15	0	mat9 $\times$ 13 $-$ $e1$	35	0
mat10 $\times$ 12 $-$ $e100$	5030	1.26	mat11 $\times$ 13 $-$ $e100$	*	*
mat10 $\times$ 12 $-$ $e20$	3893	0.86	mat11 $\times$ 13 $-$ $e20$	252951	6988.7
mat10 $\times$ 12 $-$ $e10$	2252	0.27	mat11 $\times$ 13 $-$ $e10$	179443	3537.4
mat10 $\times$ 12 $-$ $e1$	47	0	mat11 $\times$ 13 $-$ $e1$	156	0
mat10 $\times$ 14 $-$ $e100$	*	*	mat11 $\times$ 15 $-$ $e100$	*	*
mat10 $\times$ 14 $-$ $e20$	124648	1342.5	mat11 $\times$ 15 $-$ $e20$	*	*
mat10 $\times$ 14 $-$ $e10$	73820	453.8	mat11 $\times$ 15 $-$ $e10$	221660	3699
mat10 $\times$ 14 $-$ $e1$	107	0.01	mat11 $\times$ 15 $-$ $e1$	330	0.02

parameters are also fairly depended the magnitude of matrices' entries. Hence, for problem with smaller entries' values, it will be easier to apply our approach.

In next part, we will apply our proposed method to compute the exact Shapley values for small-scale problem instance when the number of player is less than 20. For larger cases with $n = 20, 40, 60, 80, 100$, the algebraic Gröbner algorithm 2 is compared with the integer programming solver CPLEX 12.6 where the performance measure is the computational time.

5.1 Illustrative example

The next example shows us how to utilize the 4ti2 software (4ti2), which is an open source package to calculate the Gröbner bases, to help solve a simple integer program.

Example 1. **(Linear Integer Problem)** Consider a multidimensional knapsack problem of the following form:

$$\max\ \boldsymbol{p}^T\boldsymbol{z} \quad \text{s.t.} \quad \mathrm{W}\boldsymbol{z} \leq \boldsymbol{b}, \quad \boldsymbol{z} \in \mathbb{Z}_+^8$$

where: $\boldsymbol{b} = [433, 432, 344, 619, 442, 508]^T$, $\boldsymbol{p} = [3, 9, 31, 28, 21, 8, 10, 7]^T$, and

$$\mathrm{W} = \begin{pmatrix} 82 & -28 & 96 & 80 & 68 & 71 & 70 & 77 \\ 91 & 55 & 49 & 96 & 76 & -24 & -32 & 80 \\ 13 & 96 & 81 & 66 & 75 & 28 & 96 & 19 \\ 92 & 97 & 15 & -4 & 40 & 5 & -16 & 49 \\ 64 & -16 & 43 & 85 & 66 & 10 & 44 & 45 \\ 10 & 98 & 92 & 94 & -18 & 83 & 39 & 65 \end{pmatrix}.$$

First, we transform the problem into the standard form with new slack variables $\boldsymbol{y} \in \mathbb{Z}_+^6$ and the new coefficients of $\mathrm{A} = [\mathrm{W}|I_6]$ and $\boldsymbol{c} = [\boldsymbol{p}|0_6]$. With new variable $\boldsymbol{z} = [\boldsymbol{x}, \boldsymbol{y}]$, we have the problem structure as:

$$\max\ \boldsymbol{c}^T\boldsymbol{x} \quad \text{s.t.} \quad \mathrm{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \in \mathbb{Z}_+^{14}$$

Using *4ti2* software, the computational time of this Gröbner basis is less than 0.03 seconds and it contains 1057 vectors of dimension 14.

$$G_{\mathrm{A},<_{\boldsymbol{c}}} = \left\{ \begin{array}{cccccccccccccc} -5 & 0 & 2 & -2 & 0 & 0 & 0 & 1 & 301 & 469 & 16 & 373 & 359 & -11 \\ -4 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & 235 & 331 & 18 & 300 & 253 & -23 \\ \vdots & & & & & & & & & & & & & \\ 5 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & -358 & -332 & -5 & -439 & -212 & -66 \\ 6 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -424 & -470 & -3 & -512 & -318 & -78 \end{array} \right\},$$

The straightforward initial feasible solution is

$$\boldsymbol{x}_0 = [0, 0, 0, 0, 0, 0, 0, 0, 433, 432, 344, 619, 442, 508].$$

Afterwards, we move along the vector

$$\boldsymbol{t}_1 = [0, 0, 0, -1, 0, 0, 0, 0, 80, 96, 66, -4, 85, 94]$$

with the step length $\lambda_1 = -4$ to get the next feasible solution :

$$\boldsymbol{x}_1 = [0, 0, 0, 4, 0, 0, 0, 0, 113, 48, 80, 635, 102, 132].$$

Repeating this process we have:

$$\boldsymbol{t}_2 = [0, 0, -1, 1, 0, 0, 0, 0, 16, -47, 15, 19, -42, -2] \text{ and } \lambda_2 = -4.$$

$$\boldsymbol{x}_2 = [0, 0, 4, 0, 0, 0, 0, 0, 49, 236, 20, 559, 270, 140],$$

$$\boldsymbol{t}_3 = [0, 0, 2, -2, 0, -1, 0, 0, 39, 70, -2, -33, 94, 87] \text{ and } \lambda_3 = -1.$$

Finally

$$\boldsymbol{x}^* = [0, 0, 2, 2, 0, 1, 0, 0, 10, 166, 22, 592, 176, 53].$$

The final solution of the optimization problem is $\boldsymbol{x}^* = [0, 0, 2, 2, 0, 1, 0, 0]$ with the optimal value is 126. Using the algebraic Gröbner method, this IP problem is solved in 0.019 second by the augmentation algorithm, however CPLEX need 0.1 seconds. In the next section, we will show how to apply this algebraic approach to reduce the total computation time of the Shapley value.

The knapsack problem has eight object types in six dimensions. Assume that we have 10 players in a simple games with different resource vectors, which entries are randomly generated in the range $[0, 300]$ as follows:

$$\mathrm{B} = \begin{pmatrix} 198 & 57 & 27 & 65 & 124 & 82 & 24 & 208 & 223 & 218 \\ 275 & 218 & 147 & 53 & 14 & 196 & 288 & 269 & 47 & 41 \\ 190 & 236 & 279 & 22 & 63 & 287 & 162 & 102 & 209 & 94 \\ 110 & 292 & 237 & 268 & 219 & 131 & 252 & 19 & 253 & 126 \\ 166 & 256 & 146 & 192 & 195 & 21 & 51 & 260 & 218 & 264 \\ 59 & 163 & 137 & 43 & 144 & 17 & 78 & 87 & 108 & 46 \end{pmatrix}.$$

The time of calculating exact algorithm using the CPLEX is 104.27(s), but using the Gröbner method, it took only 16.39(s). This small example show the potential of the algebraic Gröbner approach in solving Integer Program problems.

5.2 Exact Shapley value calculation for small scale MIKGs

Computations of the Shapley value is know as $\#P$-hard problems for many cooperative games. However, in the cases with the number of players $n <= 20$, we are able to find the exact Shapley value if there is enough time to calculate all the 2^n characteristic values). Table (2) describes the computational times of Algebraic Gröbner algorithm (1) in column T_{AG} and the time of CPLEX in column T_{CPLEX}, where the problem sizes (r, q) range from 4×6 to 10×12.

This algebraic approach required us to find the Gröbner bases for the matrix in standard form $\mathrm{A} = [\mathrm{W}|\mathrm{I}_r]$ at the beginning as fixed cost to invest in the long term process. We could see different test cases' computational times of the Gröbner bases in the table 1. However, the next step of using the augmentation algorithm is quicker than the typical ILP solver. As we need to solve the Integer Programming model many times for a specific problem instance with changed rhs, this process with low variable costs will bring down the total calculation time in the long run.

$S_{Problem}$	$T_{Groebner}$	$N_{Players}$	T_{Augmt}	T_{AG}	T_{CLPEX}
$mat4 \times 6 - e100$	0	5	0.06	0.06	0.29
		8	0.22	0.22	3.31
		11	8.07	8.07	62.7
		14	20.57	20.57	259.08
		17	185.17	185.17	1276.62
$mat4 \times 8 - e10$	0	5	0.05	0.06	1.35
		8	0.26	0.26	11.35
		11	2.47	2.47	23.26
		14	22.01	22.01	227.33
		17	193.14	193.14	1541.29
$mat6 \times 8 - e100$	0	5	0.06	0.06	0.42
		8	0.26	0.27	4.29
		11	13.19	13.20	192.75
		14	23.04	23.05	422.17
		17	202.40	202.41	2413.80
$mat6 \times 10 - e10$	0.01	5	0.09	0.1	0.43
		8	0.21	0.21	2.27
		11	1.95	1.95	17.76
		14	18.67	18.67	201.48
		17	172.05	172.05	1440.58
$mat8 \times 10 - e100$	0.19	5	0.26	0.45	0.51
		8	1.19	1.38	4.59
		11	15.13	15.32	154.47
		14	63.85	64.04	462.63
		17	533.26	533.45	2356.87
$mat8 \times 12 - e10$	0.05	5	0.46	0.51	.88
		8	0.80	0.85	4.59
		11	7.09	7.14	37.30
		14	54.63	54.68	253.64
		17	482.25	482.30	2003.28
$mat10 \times 12 - e100$	1.26	5	0.38	1.64	0.63
		8	2.53	3.79	5.77
		11	21.07	22.33	237.75
		14	182.94	184.2	636.16
		17	847.89	849.15	4721.02

Table 2. Comparison of the computational time to find the exact Shapley value

5.3 Algebraic Gröbner method to approximate the Shapley value for large-scale MIKGs

When the number of players is getting larger, the algebraic Gröbner sampling algorithm 2 will be implemented to compared with CPLEX solvers. We will fix the sampling budget (the number of samples is 100 for each Shapley value of each player) for sampling. We can apply the AG method to find the approximation Shapley value and compare with the exact Shapley value to see the potential errors. For some problem instances $(\mathrm{A}, \mathrm{B}, \boldsymbol{c})$, we consider the case when the number of players ranges from 20 to 100.

$S_{Problem}$	$T_{Groebner}$	$N_{Players}$	T_{Augmt}	T_{AG}	T_{CLPEX}
$mat5 \times 7 - e100$	0	20	1.15	1.15	37.16
		40	1.58	1.58	86.34
		60	2.47	2.47	162.77
		80	3.16	3.16	198.08
		100	4.66	4.66	211.39
$mat5 \times 9 - e100$	0	20	0.54	0.54	7.52
		40	1.23	1.23	14.40
		60	1.84	1.84	27.21
		80	2.75	2.75	48.42
		100	3.78	3.78	67.58
$mat7 \times 9 - e20$	0.01	20	2.53	2.54	36.55
		40	4.39	4.40	64.14
		60	7.23	7.24	98.82
		80	9.72	9.73	140.93
		100	13.01	13.02	209.10
$mat7 \times 11 - e20$	0.01	20	2.97	2.98	40.08
		40	6.19	6.20	62.16
		60	9.79	9.79	98.06
		80	13.35	13.35	127.54
		100	16.77	16.77	161.93
$mat9 \times 11 - e10$	0.92	20	49.01	49.93	30.95
		40	113.56	114.48	76.60
		60	188.97	189.89	97.10
		80	237.33	238.25	150.69
		100	293.93	294.85	162.21
$mat9 \times 13 - e10$	0.09	20	19.84	19.93	50.57
		40	31.33	31.42	105.32
		60	50.14	50.23	164.09
		80	65.20	65.29	185.46
		100	86.15	86.24	288.82
$mat11 \times 13 - e1$	0.02	20	4.32	4.34	59.08
		40	6.19	6.21	64.81
		60	7.72	7.74	90.97
		80	10.47	10.49	100.77
		100	13.55	13.57	134.95
$mat11 \times 15 - e1$	0	20	6.72	6.72	17.41
		40	9.67	9.67	32.58
		60	14.95	14.95	43.99
		80	19.68	19.68	57.76
		100	26.90	26.90	71.92

Table 3. Time comparison of algebraic Gröbner method and CPLEX solver for approximation method to calculate the Shapley value with 100 samples for each player

From the numerical result, we found that the algorithm was competitive to CPLEX, often faster than five times for this problem instances. From the table 3, we can see the advantage of the Gröbner basis compared to CPLEX solver. Below we will show the table of computational time for the AG methods compared to CPLEX.

We can also investigate the error of our sampling techniques. Consider the cases when there are only three type of players, we can evaluate the exact Shapley value for this type of MIKGs with large number of player (for example, $n = 100$). Then we use the algorithm 2 to compare with the exact algorithm to see the errors (RMSE and MAPE). The *mean absolute percentage error* (MAPE) is a measure between the approximate and the exact Shapley values, and which is calculated as: $\text{MAPE}(\boldsymbol{\phi}, \widehat{\boldsymbol{\phi}}) = 100.\frac{1}{n}\sum_{i=1}^{n}|\frac{\widehat{\phi}_i - \phi_i}{\phi_i}|$, and the *root mean squared error* (RMSE) is a quadratic scoring rule which measures the average magnitude of the error. The RMSE is calculated as: $\text{RMSE}(\boldsymbol{\phi}, \widehat{\boldsymbol{\phi}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{\phi}_i - \phi_i)^2}$, where $\boldsymbol{\phi}$ is the real Shapley value and $\widehat{\boldsymbol{\phi}}$ is the approximate Shapley value. We want to combine both sampling errors MAPE and RMSE because the MAPE depends on the proportional value of the output, and the RMSE relies on the total vector value $\nu(N)$.

The diagram 2, 3 shows the relationship between the timing budget and the MAPE sampling errors. We also see that the two methods of using the Algebraic Gröbner approach and the CPLEX solver will generate different errors with respect to the time.

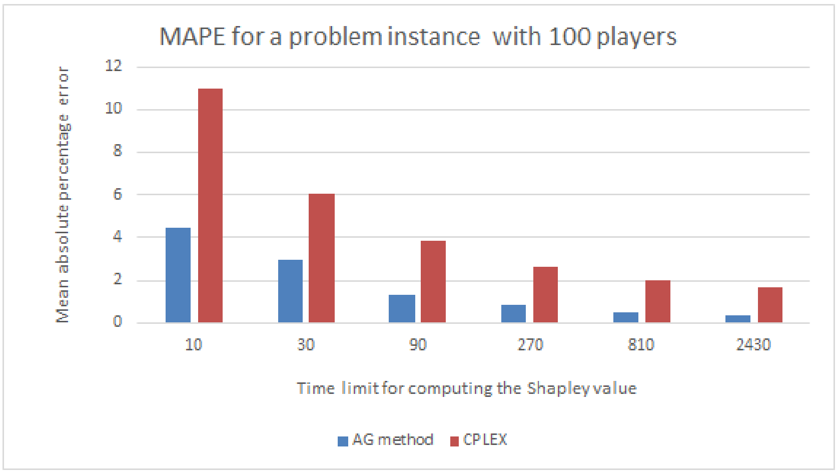

Fig. 2. MAPE Comparison of AG method and CPLEX on the test instance with 100 players

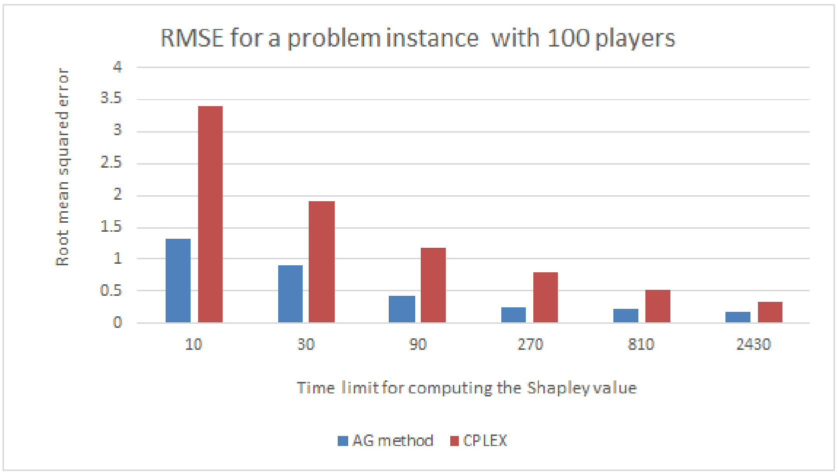

Fig. 3. RMSE Comparison of AG method and CPLEX on the test instance with 100 players

From the graph for most of our games with 100 players, we see that if we set the time limit is less than one hour, the sampling errors are quite small (i.e. the RMSE less than 2 for $\nu(N) = 2671$ and the MAPE is less than 1%). With the same time limit, the AG approach can solve the Shapley value with significantly smaller errors compared to the CPLEX solver for the tested instances.

6 Conclusion & Discussion

We have introduced the Multidimensional Integer Knapsack Games and its applications. The Shapley value is proposed to allocate the total value of the collaboration for the grand coalition of players. We proposed an algebraic Gröbner approach to find the Shapley values of the MIKGs. For the games with large number of players ($n \geq 20$), the sampling-based method is used to combine with the algebraic Gröbner approach for the approximation of the Shapley value. We also summary some results of the experiments and show the effects of our algorithm in different problem instances.

References

[Bachrach & Rosenschein (2008)]Bachrach, Y, Rosenschein, JS. Coalitional skill games, *AAMAS '08 Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems* , 2, 1023-1030 , 2008.

[Bertsimas & Demir(2002)]Dimitris Bertsimas and Ramazan Demir. An Approximate Dynamic Programming Approach to Multidimensional Knapsack Problems. *Management Science*, 55-565, 2002.

[Bhagat et al.(2014)]Smriti Bhagat, Anthony Kim, S. Muthukrishnan, Udi Weinsberg. The Shapley Value in Knapsack Budgeted Games. *Web and Internet Economics*, vol. 8877 of the series Lecture Notes in Computer Science, 106–119, 2014.

[Borm et al.(2001)]Peter Borm, Herbert Hamers, Ruud Hendrickx, Operations research games: A survey, *Top*, 9(2), 139–199,2001

[Buchberger(1985)]B. Buchberger. Grbner bases: An algorithmic method in polynomial ideal theory *N.K. Bose (Ed.), Multidimensional Systems Theory, D. Reidel Publishing Co.* (1985), pp. 184232

[Caprara & Letchford(2010)]A. Caprar and A. N. Letchford. New techniques for cost sharing in combinatorial optimization games, *Mathematical Programming , Ser. B.* 124: 93–118, 2010.

[Castro et al.(2011)]Castro, F., Gago, J., Hartillo, I., Puerto, J., Ucha, J.M.: An algebraic approach to integer portfolio problems. Eur. J. Oper. Res. 210(3), 647–659 (2011).

[Chalkiadakis et al.(2011)]G. Chalkiadakis, E. Elkind, and M. Wooldridge. *Computational Aspects of Cooperative Game Theory.* Morgan Claypool Publishers.

[Conti & Traverso (1991)]P. Conti and C. Traverso, Gröbner bases and integer programming, Proceeding AAECC-9 (New Orleans), Springer Verlag, LNCS 539, 130–139, 1991

[Cox et al.(1998)]D. Cox, J. Little and D. OShea, Using Algebraic Geometry, Springer-Verlag, New YorkBerlin-Heidelberg, 1998.

[Deng & Papadimitriou (1994)]Deng, X., Papadimitriou, C.H.: On the complexity of cooperative solution concepts. Mathematics of Operations Research. 19(2), 257–66 (1994)

[Deng et al.(1999)]Xiaotie Deng, Toshihide Ibaraki, Hiroshi Nagamochi. Algorithmic Aspects of the Core of Combinatorial Optimization Games. *Mathematics of Operations Research* 24(3):751–766, 1999.

[Dimitris(2000)]Dimitris Bertsimas, Georgia Perakis, Sridhar Tayur. A new algebraic geometry algorithm for Integer programming, Management Science, Vol 46, No.7 July 2000, 999–1008.

[Gago et al.(2013)]Gago V., Manuel J., Hartillo H., Maria I., Puerto A., Justo, U.E., Jose M. Exact cost minimization of a series-parallel reliable system with multiple component choices using an algebraic method. *Computers & Operations Research* 40(11). 2752–2759, 2013.

[Gago et al.(2015)]J. Gago-Vargas A I. Hartillo J. Puerto. J. M. Ucha, An improved test set approach to nonlinear integer problems with applications to engineering design. *Computational Optimization and Applications*, DOI 10.1007/s10589-015-9739-3

[Gebauer & Moeller(1988)]R. Gebauer and H. M. Moeller. On an installation of Buchberger's algorithm. *Journal of Symbolic Computation.* 6 (1988), 275–286.

[Granot (1986)]Granot, D. 1986. A generalized linear production model: A unified model. *Mathematical Programming.* 34 212–222.

[Graver(1975)]J.E. Graver. On the foundations of linear and integer programming, *Mathematical Programming*, 8, 207-226, 1975.

[Hemmecke (2002)]R. Hemmecke. On the Computation of Hilbert Bases of Cones. in: Mathematical Software, ICMS 2002, A. M. Cohen, X.-S. Gao, N. Takayama, eds., World Scientific, 2002.

[Hemmecke & Malkin(2009)]R. Hemmecke, P. N. Malkin. *Computing generating sets of lattice ideals and Markov bases of lattices.* Journal of Symbolic Computation, 44(10), 1463–1476, 2009.

[Hemmecke et al. (2011)]R. Hemmecke, S. Onn and R. Weismantel. A polynomial oracle-time algorithm for convex integer minimization. *Mathematical Programming, Ser.A*, 126:97-117, 2011.

[Hosten & Sturmfels(1995)]S. Hosten and B. Sturmfels. GRIN: An implementation of Gröbner bases for integer programming. *Integer programming and combinatorial optimisation.* E. Balas and J. Clausen, eds., LNCS 920, Springer-Verlag, 267–276, 1995.

[Loera et al.(2013)]Jesus A. De Loera, Raymond Hemmecke, and Matthias Koppe. *Algebraic and Geometric Ideas in the Theory of Discrete Optimization.* MOS-SIAM Series on Optimization, 2013.

[Onn (2010)]Shmuel Onn, *Nonlinear Discrete Optimization. An Algorithmic Theory* , Zurich Lectures in Advanced Mathematics, 2010.

[Owen (1975)]Owen, G. 1975. On the core of linear production games. *Mathematical Programming.* 9, 358–370.

[Puchinger et al.(2010)]Jakob Puchinger, Günther R. Raidl, Ulrich Pferschy. The Multidimensional Knapsack Problem: Structure and Algorithms. *INFORMS Journal on Computing.* 22(2):250-265, 2010.

[Pisinger (2005)]Pisinger, David. Where are the hard knapsack problems? *Computers and Operations Research.* 32(9), 2271-2284, 2005.

[Shapley(1953)]L.S. Shapley. A value for n-person games. *Contributions to the Theory of Games II*, 307-317. Princeton University Press, 1953.

[Sturmfels (1996)]Sturmfels, B.: *Gröbner Bases and Convex Polytopes*, vol. 8. American Mathematical Society, Providence (1996)

[Sturmfels (2003)]Sturmfels, B.: Algebraic Recipes for Integer Programming. *Proceedings of Symposia in Applied Mathematics: Trends in Optimization*, 61, (2004)

[Sturmfels & Thomas (1997)]Sturmfels, Bernd, Thomas, Rekha R.: Variation of cost functions in integer programming, *Mathematical Programming*, 77(2), 357–387, (1997)

[Tayur et al. (1995)]Tayur, S.R., Thomas, R.R., Natraj, N.R.: An algebraic-geometry algorithm for scheduling in presence of setups and correlated demands. *Mathematical Programming.* 69(3), 369–401 (1995)

[Rekha Thomas(1995)]R. Thomas. A Geometric Buchberger Algorithm for Integer Programming, *Mathematics of Operations Research*, 20(4), 864–884, 1995.

[Wolsey & Nemhauser]Integer and Combinatorial Optimization. L. A. Wolsey, G. L. Nemhauser. *John Wiley & Sons.*

[4ti2]4ti2 team, 4ti2 a software package for algebraic, geometric and combinatorial problems on linear spaces. http://www.4ti2.de/