ISSN 2278 – 0211 (Online)

An Efficient Algorithm for Solving Nucleolus of Cooperative TU Games Using MATLAB

M. V. Durga Prasad
Professor, Institute of Rural Management Anand, Anand, Gujarat, India

Abstract:
The approach to determine the nucleolus of the cooperative game is to apply a sequence of linear programs. In this paper a new algorithm for solving Nucleolus for cooperative TU games is proposed. Some constraints are dropped at any iteration for solving a linear program as those dropped constraints will not contribute in getting unique solution. Also dropping constraints will not affect in the single point Nucleolus solution but the total number of linear programs to be solved will be less in the proposed method. It can be claimed that Nucleolus can be obtained in solving maximum (n-1) linear programs for a n person cooperative TU games. Numerical examples for different set of players are provided. Advantages of using MATLAB are discussed. The results clearly prove the effectiveness of the proposed method with the usage of MATLAB.

Keywords: *Cooperative transferable utility game, Linear programs, Nucleolus, Algorithm, Allocation*

1. Introduction

Transferable utility n-person co-operative games have been studied extensively in the recent past (see References [ii], [v], [x], [xii] etc). Core of the cooperative game was introduced by J. Von Neumann and Morgernstern, 1944 [xvii]. Other allocation (solution) concepts like Nucleolus, Nucleon, F-Nucleolus etc have been introduced and considered as relaxations to the concept of core. To get a unique allocation vector one should look into single point solution concepts. Nucleolus is one of the single point solution concepts.
In recent past several authors had applied cooperative TU games in various fields like solving power transmission loss allocation problems, solving public transport problems; cost-saving allocation problems in three level supply chain (see [xviii], [iv] & [viii] respectively) Nucleolus is the most commonly used concept in solving cooperative TU games.
To compute the nucleolus (due to Schmeidler [xii], we know that a sequence of linear programs is to be solved. Several authors have developed algorithms for obtaining solution concept like Nucleolus (see references [v], [ii] & [x]). The central question in finding nucleolus is how to detect all those coalitions at an iteration whose excess cannot be further improved without harming the position of worse off coalitions. The answer to this question can be well explained if one uses MATLAB software. In the proposed algorithm for obtaining nucleolus the concepts of linear algebra as well as dropping of constraints has been used. The idea of dropping constraints is common in the terminology of relationship between game theory and mathematical programming duality
(see [ix]). Potters et al [x] have used the concept of dropping constraints in terms of elementary rows. Our algorithm is based on an algorithmic scheme given in Faigle et al [v] and Dragan [ii]. In both algorithms it may uses more than (n-1) programs. However, in algorithmic scheme of Potters et al [x], it uses maximum of (n-1) linear programs. This occurs mainly due to the dropping of linearly dependent constraints at the end of every loop. In our algorithm we use one program less when compared to Potters et al [x].
The organization of the rest of the paper is as follows. Section 2 deals with the preliminaries and the statement of the problem. Algorithm for the proposed method in obtaining Nucleolus which includes the comparison of MATLAB usage with other softwares is presented in section 3. Section 4 consists of presenting numerical examples of 3, 4 & 5 player cooperative TU games with results

2. Preliminaries & Statement of the problem

Let (N, v) be the n-person transferable utility co-operative game in characteristic function form where N= {1, 2, ---, n} be a set of n players. These players may form coalitions $S \subseteq N$ in an arbitrary way. Each coalition S can achieve a value $v(S) \in R$. The value v (N) of the grand coalition N can thus be understood as the total profit arising from the co-operation of all players. An allocation is an n-dimensional vector with component sum equal to v(N).
The Nucleolus can be solved by a sequence of the following linear programs:
(LP_1) Max ε
Subject to

$$\sum_{i \in N} x_i = v(N)$$
$$\sum_{i \in S} x_i \geq v(S) + \varepsilon \qquad \forall S \notin \{\phi, N\}$$

The set of optimal solutions of (LP_1) is usually called least core of the game. In [v], the same set has been called as prenucleolus of the game. If (LP_1) has a unique optimal solution (x^*, ε_1) then x^* is the nucleolus of the game (N, v). Otherwise we go to second step program (LP_2) where

(LP_2) Max ε
Subject to

$$\sum_{i \in N} x_i = v(N)$$
$$\sum_{i \in S} x_i = v(S) + \varepsilon_1 \qquad \forall S \in \tau_1{}^*$$
$$\sum_{i \in S} x_i \geq v(S) + \varepsilon \qquad \forall S \notin \{\phi, N, \tau_1\}$$

*The set of active coalitions which are common to all optimal solutions at $\varepsilon = \varepsilon_1$. We have used common active coalitions instead of active coalitions used in [v].

Continuing this way, we obtain a sequence $\varepsilon_1 < \varepsilon_2 < \varepsilon_3$ ------$< \varepsilon_k$ until finally the optimal solution of (LP_k) is unique with an allocation vector x^*, the nucleolus of the game.

It is observed that all the common active coalitions at any iteration may become redundant to the system of common active coalitions in the previous iteration. In this case improvement towards unique solution has become void at the present iteration. The main aim in this paper is to overcome such situations. This kind of situation is explained in the numerical example of four player problem. The proposed algorithm presented in section 3 will help in overcoming such kind of situations. This algorithm is based on the following results:

Theorem1: Let Γ_1 be an optimal solution set of (LP_1) with ε^* as optimal value and Γ_j be an optimal solution set of any (LP_j) for j = 2 ,3, ---, k with optimal value $\hat{\varepsilon}_j$, then $(\hat{x}, \hat{\varepsilon}_j) \in \Gamma_j$ implies that $(\hat{x}, \varepsilon^*) \in \Gamma_1$.

Proof: We can write (LP_1) is equivalent to the following program:

$$\text{Max} \quad \underset{S \notin (\phi, N)}{Min} \{\sum_{i \in S} x_i - v(S)\}$$

Subject to

$$\sum_{i \in N} x_i = v(N)$$

Since $\hat{x}$ satisfies grand coalitional constraint and ε^* is the optimal value of the above program, it follows that $\sum_{i \in S} \hat{x}_i - v(S) \geq \varepsilon^* \ \forall S \notin \{\phi, N\}$

Hence it is proved.

Conjugate: Any coalition S^c is said to be a conjugate of a coalition S if S^c and S are disjoint and their union is a grand coalition.

Theorem1 will be helpful in establishing the following proposed algorithm

3. The Proposed Algorithm

In the linear program (LP_2) the active coalitions $S \notin \{\phi, N, \tau_1\}$ can be expressed as union of two or more disjoint coalitions $S \in \tau_1$. Such active coalitions will not contribute towards unique solution and we have to wait for the next iteration. For example, if S_1 = {1,2}and S_2= {3,4} $\in \tau_1$ then their coalitional constraints are $x_1 + x_2 = v(S_1) + \varepsilon_1$ & $x_3+x_4=v(S_2) + \varepsilon_1$. If their union becomes active in the next iteration, then its coalitional constraint will be $x_1+x_2+x_3+x_4 = v(S_1 U S_2) + \varepsilon_2$ which will be redundant and hence no contribution towards obtaining unique solution. Such coalitional constraints can be dropped in the respective linear program. In LP_2 the obvious dropped constraint is the grand coalitional constraint. The reasons of dropping can well be explained by the following Theorem2:

Theorem2: If S1& S2 are any two active disjoint coalitions in the first linear program (LP_1) then coalitional constraint corresponding to the coalition S1 U S2 can be dropped in the next and subsequent linear programs.

Proof: Let $(x^*, \varepsilon^*) \& (\hat{x}, \hat{\varepsilon}_j)$ are optimal to (LP_1) and (LP_j) respectively, where j=2,3, ----k. It follows from Theorem1 that $(\hat{x}, \varepsilon^*)$ satisfies the coalitional constraint corresponding to the coalition S1 U S2. i.e.

$$\sum_{i \in S1US2} \hat{x}_i \geq v(S1U\,S2) + \varepsilon^* \;\Rightarrow\; \sum_{i \in S1US2} \hat{x}_i - v(S1U\,S2) \geq \varepsilon^*$$

As $\sum_{i \in S1US2} \hat{x}_i - v(S1U\,S2)$ is always greater than or equal to ε^*, it can be equal to either ε^* or ε^*+α where α >0. In any case it is an equation with some $\varepsilon \geq \varepsilon^*$. This equation will be redundant and it is not helpful in getting unique allocation and hence it can be dropped.

Theorem 3: If S_1 is any active coalition in the current linear program and its conjugate S_1^c is not active then S_1^c can be dropped in the next and subsequent linear programs.

Proof: By the definition of conjugate, S_1^c can be written as linear combination of grand coalition N and S_1. Therefore, the coalition S_1^c can be dropped in the next and subsequent linear programs.

Corollary: If S_1 and S_2 are disjoint active coalitions in the current linear program and if $(S_1 \cup S_2)^c$ is not active then $(S_1 \cup S_2)^c$ can be dropped in the next and subsequent linear programs.
Note: Theorem2 and Theorm3 can be extended to more than two coalitions.
We now present the new algorithm for obtaining Nucleolus.

Algorithm
Step 1: Solve LP_1 as mentioned in the section 2. Identify the common active coalitions** and fixations of any variables. If the solution is unique then stop and claim the present solution as Nucleolus. Otherwise go to step 2
**MATLAB provides solution which is interior to the set of optimal solutions. This will help in identifying active common coalitions by just finding the binding constraints of the optimal solution provided by MATLAB. If we use other softwares like Microsoft solver etc, then it will give the optimal solution in one of the extreme points of the optimal solution set. In this case the common active coalitions can be identified by either searching a closer neighborhood interior point of the optimal solution set and find the active coalitions or search for other extreme point and find common active coalitions for the two extreme points. In a large number of players, it may be difficult to find the common active coalitions if we use softwares whose outputs are not similar to that of MATLAB. An optimal solution in the first linear program through MATLAB will be very close to the nucleolus as it gives an interior point of optimal solution set.
Step2: Solve
(LP_2) Max ε
Subject to

$$\sum_{i \in N} x_i = v(N)$$

$$\sum_{i \in S} x_i = v(S) + \varepsilon_1 \qquad \forall S \in \tau_1$$ (The set of common active coalitions at ε = ε_1)

$$\sum_{i \in S} x_i \geq v(S) + \varepsilon \qquad \forall S \notin \{\phi, N, \tau_1\} \cup \xi_1 \cup C$$

where $\xi_1 = \{S / S = S_i^1 U S_i^2 U ... U S_i^p, S_i^l \in \tau_1 \; \forall l = 1,2,..., p \;\&\; p \geq 2\}$

and $C = \{S / S \notin \tau_1, S = T^c, T \in (\tau_1 \cup \xi_1)\}$

Continuing this way, we obtain a sequence $\varepsilon_1 < \varepsilon_2 < \varepsilon_3$ ------$< \varepsilon_h$ until finally the optimal solution of (LP_h) is unique with an allocation vector x^*, the nucleolus of the game.
It is to be observed here that $\varepsilon_h = \varepsilon_k$ where h ≤ k. i.e. The optimal value in the final linear programs of the proposed algorithm and the existing algorithm are equal but the total number of linear programs used in the proposed algorithm is less than that in the existing algorithm. This algorithm can be applied to both balanced and unbalanced cooperative TU games.
Note: Earlier Reijneirse et al [xi] have proved that for an n-player cooperative TU game there exist a collection of maximum 2(n-1) coalitions that determine the nucleolus. Also sufficient conditions are established to have same maximal value even after dropping coalitions. More relaxed conditions are considered in Theorem 2 & 3.
We now present with the examples of three, four & five player problems in section 4.

4. Numerical Examples

Example 4.1
Consider three player problem with v (1) =54, v (2) =78, v (3) =54, v (1,2) =150, v (1,3) =108, v (2,3) =132, v (1,2,3) =210. Results are given in Table-1. We observe that the existing method and proposed method have same number of iterations. The active coalitions in the first linear program LP_1 are disjoint in this case.

Method	Player 1	Player 2	Player 3	Epsilon
LP I Iteration Existing method	64.7229	88.2771	57.0000	3.000
LP II Iteration Existing method	64.5000	88.5000	57.0000	10.500
LP I Iteration Proposed method	64.7229	88.2771	57.0000	3.000
LP II Iteration Proposed method	64.5000	88.5000	57.0000	10.500

Table 1

Example 4.2:
Consider a four-player problem with data given in the table-2 and results are given in the table-3.

Coalition	{1}	{2}	{3}	{4}	{1,2}	{1,3}	{1,4}	{2,3}	{2,4}
Value	54	78	54	30	150	108	86	132	110

Coalition	{3,4}	{1,2,3}	{1,2,4}	{1,3,4}	{2,3,4}	{1,2,3,4}
Value	88	210	182	145	170	255

Method	Player 1	Player 2	Player 3	Player 4	Epsilon
LP I Iteration Existing method	66.4814	90.5186	61.000	37.000	7.000
LP II Iteration	66.4644	90.5356	61.000	37.000	8.000
LPIII Iteration	66.4451	90.5549	61.000	37.000	10.00
LPIV Iteration Existing method	66.4207	90.5793	61.000	37.000	12.00
LPV Iteration Existing method	66.5	90.5	61.000	37.000	12.5
LPI Iteration Proposed method	66.4814	90.5186	61.000	37.000	7.000
LPII Iteration Proposed method	66.5	90.5	61.000	37.000	12.5

Table 2:4-player problem results

As we observe from the table-3 final iteration results of both existing method and proposed method are same but in the proposed method we used only two linear programs whereas five linear programs are used in the existing method. In the first linear program result the active coalitions are {1,2}, {3}, {4}. So $\xi_1 =$ ({1,2,3}, {1,2,4} and {3,4})
$C =$ ({3,4}, {1,2,4} and {1,2,3}). In this case all active coalitions are disjoint and hence the set ξ_1 coincides with C.

Example 4.3:
Let us consider a 4-player unbalanced problem with data given in table-4 and results are given in table-5.

Coalition	{1}	{2}	{3}	{4}	{1,2}	{1,3}	{1,4}	{2,3}	{2,4}
Value	0	0	0	0	18	0	2	0	2

Coalition	{3,4}	{1,2,3}	{1,2,4}	{1,3,4}	{2,3,4}	{1,2,3,4}
Value	4	21	19.5	8	9	22

Table 3:4-player unbalanced problem

Method	Player 1	Player 2	Player 3	Player 4	Epsilon
LP I Iteration Existing method	8.8769	9.2897	2.6667	1.1667	-0.1667
LP II Iteration	8.8919	9.2747	2.6667	1.1667	0.1667
LPIII Iteration	8.8835	9.2832	2.6667	1.1667	1.1667
LPIV Iteration Existing method	8.8768	9.2898	2.6667	1.1667	2.6667
LPV Iteration Existing method	8.5833	9.5833	2.6667	1.1667	4.4167
LPI Iteration Proposed method	8.8769	9.2897	2.6667	1.1667	-0.1667
LPII Iteration Proposed method	8.5833	9.5833	2.6667	1.1667	4.4167

Table 3: 4-player problem results

In the first iteration the active coalitions are ({3,4}, {1,2,3} and {1,2,4}. As these are not disjoint coalitions so $\xi_1 = \varphi$ and $C =$ ({3,4}, {1,2,4} and {1,2,3}). Potters et al dropped the same constraints as in the set ξ_1 but the termination of algorithm extends it to one more loop (program) with total 7 pivot steps. It is observed that MATLAB results of first linear program are closely related to nucleolus.

Example 4.4
Consider a 5-player problem with coalitional values is given in appendix Table-I and results of LP iteration existing method and proposed method are given in appendix table-II. In the first linear program it is observed that coalitions {1,5}, {2,3}, {2,4} and {3,4} are active. $\xi_1 =$ ({1,2,3,5}, {1,2,4,5} and {1,3,4,5}) and C= ({2,3,4}, {1,4,5}, {1,3,5}, {1,2,5}, {2}, {3}, {4}). We can observe from the set C that the players of 2,3 and 4 are fixed and no improvement later. More than dropping of constraints at later stage the information about fixation of players will be known from the set C.
Note: Computational time in calculating Nucleolus by the proposed method is much lower than that of determining nucleolus by existing sequence of linear programs used in [5] as the former method has to solve less number of linear programs than the latter one. MATLAB 7.1 is used to carry out the simulations. MATLAB subroutine program for a five player problem is given in appendix. In this program checking of uniqueness of solution as well as the linearly dependent coalitional constraints is done at the end of every linear program in the form of rank. Dropped constraints set which was obtained from checking the linear dependent coalitional constraints by MATLAB subroutine program is claimed to be equivalent to the set $\xi_1 \cup C$. If we enter the coalitional values as inputs for b and run the MATLAB subroutine program, then we get the end result.

5. Conclusions
To compute the Nucleolus the efficient algorithm has been introduced on the basis of linear algebraic equations with rank concept. Usage of MATLAB software had helped in answering the central question in finding the nucleolus.

6. References

i. Bastian Fromen: Reducing the number of linear programs needed for solving the nucleolus problem of n-person game theory, European Journal of Operational Research. 98,626-636 (1997)
ii. Dragon, I.: A procedure for finding the nucleolus of a cooperative n-person game, Zietschrift fur operations research 25, 119-131(1981)
iii. Durga Prasad, M.V., Saloman Danaraj, R.M.: F-Nucleolus of Cooperative Games: An approach for affair allocation, Vision 2020: The strategic role of operational research, Proceedings of the annual Operations Research Society of India conference held at IIM, Ahmedabad, India, 313-320 (2005)
iv. Durga Prasad, M.V.: Public transport and cooperative TU games: A case study of Asmara city, The ICFAI journal of Science & Technology II(no.1),70-79 (2006)
v. Faigle, U. Kern, W., Fekete, S.P., Hochstattler, W.: The Nucleon of Co-operative Games and an algorithm for matching games, Mathematical Programming 83,195-211 (1998)
vi. Faigle, U., Kern, W.: On some approximately balanced combinatorial cooperative Games. ZOR - Methods and Models of Operations Research 38, 141-152 (1993)
vii. Granot, D, Granot, F, Zhu, W.R.: Characterization sets for the nucleolus, International Journal of Game theory 27,359-374 (1998)
viii. Mingming Leng, Mahmut Parlar: Allocation of cost savings in a three level supply chain with demand information sharing: A cooperative game approach, Operations Research 57: 200-213(2009)
ix. Mond, B, Chandra, S., Durga Prasad, M.V.: Constrained games and Symmetric duality, OPSEARCH,3,1-10 (1987)

x. Potters, J.A.M., Reijnierse, J.H, Ansing, M.: Computing the nucleolus by solving a prolonged simplex algorithm, Mathematics of Operations Research 21,757-768 (1996)

xi. Reijnierse, J.H, Potters, J.A.M: The *B*- Nucleolus of TU-Games, Games and Economic Behaviour 24, 77-96 (1998)

xii. Schmeidler, D.: The nucleolus of a characteristic function game. SIAM Journal of Applied Mathematics 17, 1163-1170 (1969)

xiii. Shapley, L.S., Shubik, M.: Quasi-cores in a monetary economy with non-convex Preferences. Econometrica 34, 805-827 (1966)

xiv. Shubik, M.: Game theory models and methods in political economy. In: Handbook of Mathematical Economics, vol. 1 (K.J Arrow et al. eds.), North-Holland, New York (1981)

xv. Shubik, M.: Cooperative Game solutions Australian, Indian, and U.S. opinions, Journal of Conflict Resolution 30, 63-76(1986)

xvi. T. Solymosi, T., Raghavan, T.E.S.: An algorithm for finding the nucleolus of Assignment games. International Journal of Game Theory 23, 119-143 (1994)

xvii. Neumann, J. Von, Morgenstern, D.: Theory of Games and Economic Behaviour, Princeton Univ. Press, Princeton, NJ (1944)

xviii. Zhu Han and Vincent Poor, H.: Coalition games with cooperative transmission: A cure for the curse of boundary nodes in selfish packet-forwarding wireless Networks, IEEE transactions on communications 57, 203-213. (2009).

Appendix

Coalition	Value	Coalition	Value
{1}	15	{1,2,3}	77
{2}	25	{1,2,4}	75
{3}	20	{1,2,5}	86
{4}	18	{1,3,4}	73
{5}	30	{1,3,5}	80
{1,2}	49	{1,4,5}	77
{1,3}	47	{2,3,4}	79
{1,4}	32	{2,3,5}	92
{1,5}	56	{2,4,5}	86
{2,3}	58	{3,4,5}	84
{2,4}	55	{1,2,3,4}	100
{2,5}	66	{1,2,3,5}	112
{3,4}	52	{1,2,4,5}	110
{3,5}	63	{1,3,4,5}	107
{4,5}	50	{2,3,4,5}	115
{1,2,3,4,5}		150	

Table-I: 5-player problem

	LPI iteration Exis.method	LPII iteration Exis.method	LPIII iteration Exis.Method	LPI iteration Prop.method	LPII iteration Prop.method
Player1	22.3066	22.3971	22.3	22.3066	22.3
Player2	32.8000	32.8000	32.8	32.8000	32.8
Player3	29.8000	29.8000	29.8	29.8000	29.8
Player4	26.8000	26.8000	26.8	26.8000	26.8
Player5	38.2934	38.2029	38.3	38.2934	38.3
Epsilon	4.6	4.8	5.1	4.6	5.1

Table-II: Results

MATLAB program for 5-player problem

```
clear;
clc;
tic;
global n m
n=5;
m=2^n-1;
A=coalition(n);
e1=-ones(m-1,1);
A=[A e1];
b=[15 25 20 18 30 49 47 32 56 58 55 66 52 63 50 77 75 86 73 80  77 79  92  86 84 100 112 110 107 115 ]';
A1=A;
beq=150;
Aeq=[ones(1,n) 0];
f=[zeros(1,n) -1];
 X =linprog(f,-A,-b,Aeq,beq);
 k=A*X-b;
 kk=1;
R=rank(Aeq);
mm=1
while R<n
XX(:,mm)=X;
D(:,mm)=k;
 [A,b,Aeq,beq,k,kk,R]=main(A,b,Aeq,beq,k,kk,m,n,X);
X =linprog(f,-A,-b,Aeq,beq);
k=A*X-b;
mm=mm+1;
end
toc;
XX
D
```