Lemke and Howson Algorithm

Kunpeng Liu

February 3, 2013

1 Introduction

The Lemke-Howson algorithm is an effective method to find at least one Nash Equilibrium (NE) for a two-person bimatrix game. Here the conception of NE is extent to include not only pure NE but also mixed NE. The algorithm was first introduced in [1], and was interpreted geometrically in [2], which visualize the process of finding NE when both players' strategy sizes are small enough. The explanation here is based on the work in [2], and we also add certain our own understandings to make the statement even clearer.

2 The Lemke-Howson Algorithm

Consider a two-player game with payoff matrix as $U_i,\ i \in \{1, 2\}$ for player 1 and 2, respectively. We assume that all the elements in $U_i,\ \forall i \in \{1, 2\}$ are positive. The assumption is without loss of generality, since adding the same large number for every elements in $U_i,\ i \in \{1, 2\}$ clearly does not change the characteristic of the game, and one NE of the original game will still be a NE of the adjusted game.

Suppose player 1 has m strategies available, denoted as $S_1 = \{s_1, ..., s_m\}$, while play 2 has n strategies available, denoted as $S_2 = \{s_{m+1}, ..., s_{m+n}\}$. Therefore, A and B are both $m \times n$ matrix. Call a vector is a probability vector if it represents a probability distribution, i.e., all the elements are non-negative and the sum is 1.

Use p_i to represent the probability for player 1 to choose strategy s_i for $i \in \{1, ..., m\}$, while use q_j to represent the probability for player 2 to choose strategy s_j for $j \in \{m + 1, ..., m+n\}$. For a pair of probability vector (p, q), $p = \{p_1, ..., p_m\}^T$ and $q = \{q_{m+1}, ..., q_{m+n}\}^T$, it is a Nash Equilibrium (NE) if and only if for player 1 either $p_i = 0$, or $p_i > 0$ and s_i is the best reply to q, while for player 2 either $q_j = 0$, or $q_j > 0$ and s_j is the best reply to p. Suppose (p, q) is a NE, then:

$$\begin{aligned} U_1 \cdot q + r &= v_1 \cdot \mathbb{1} \\ U_2^T \cdot p + t &= v_2 \cdot \mathbb{1} \end{aligned} \tag{1}$$

where $r = \{r_1, ..., r_m\}^T$, $t = \{t_{m+1}, ..., t_{m+n}\}^T$ and $r \geqslant 0$, $t \geqslant 0$, while $\mathbb{1}$ indicates a column vector of 1's of appropriate dimension and $v_i,\ i \in \{1, 2\}$ is a scalar representing player i's payoff.

In (1), r satisfy the condition that $r_i \neq 0$ when and only when $p_i = 0$ for all $i \in \{1, .., m\}$ and t satisfy the condition that $t_i \neq 0$ when and only when $q_i = 0$ for all $i \in \{m+1, .., m+n\}$.

From (1), it is easy to get:

$$\begin{aligned} U_1 \cdot q' + r' &= \mathbb{1} \\ U_2^T \cdot p' + t' &= \mathbb{1} \end{aligned} \tag{2}$$

where $q' = q/v_1$, $r' = r/v_1$, $p' = p/v_2$ and $t' = t/v_2$.

Define the calculation of normalization for vectors as follows:

$$normal(x) = x / \sum_i x_i$$

Then, $normal(p') = p' / \sum_i p'_i = v_1 \cdot p' = p$. The same operation also stands for q', which brings that:

$$\begin{aligned} p &= normal(p') \\ q &= normal(q') \end{aligned} \tag{3}$$

Therefore, it is easy to get p and q as long as we can find p' and q'.

For (2), one obvious solution is $p' = 0$, $q' = 0$, $r' = \mathbb{1}$ and $t' = \mathbb{1}$. This solution, called *extraneous solution*, is a by-product of (2) and apparently does not fit our original problem. However, the extraneous solution can be used to find a practical solution, and the process will be explained in detail later.

To find p' and q', let us rewrite (2) as follows:

$$[U_2^T\ I] \begin{pmatrix} p' \\ t' \end{pmatrix} = \mathbb{1} \tag{4}$$

$$[I\ U_1] \begin{pmatrix} r' \\ q' \end{pmatrix} = \mathbb{1} \tag{5}$$

Again, in (4) and (5), r' satisfy the condition that $r'_i \neq 0$ when and only when $p'_i = 0$ for all $i \in \{1, .., m\}$ and s' satisfy the condition that $s'_i \neq 0$ when and only when $q'_i = 0$ for all $i \in \{m+1, .., m+n\}$, and these conditions can be formalized as:

$$\begin{aligned} p'_i \cdot r'_i &= 0,\ \forall i \in \{1, m\} \\ p'_i + r'_i &> 0,\ \forall i \in \{1, m\} \\ q'_j \cdot t'_j &= 0,\ \forall j \in \{m+1, m+n\} \\ q'_j + t'_j &> 0,\ \forall j \in \{m+1, m+n\} \end{aligned} \tag{6}$$

where, it is worth noting that the indexes of p and r are in the same range, and so are those of q and s.

(4) and (5) both contain $m+n$ variables, but (4) only has n equations while (5) only has m equations. In order to solve (4) and (5), it is necessary to eliminate m variables in (4) and n variables in (5). From (6), it is obvious that $m+n$ variables among the total $2m+2n$ variables have to be 0. Therefore, we need to pick m variables in (4) to be zero, and pick n variables in (5) to be zero. If we know which m variables in (4) are 0 and which n variables in (5) are 0, we can calculate the nonzero variables for (4) and (5) separately in the condition that the left equations constitute a non-singular matrix. Since the existence of NE is guaranteed by Nash's Theorem, the problem left for us is how to figure out which m variables in (4) and which n variables in (5)should be zero. From this aspect, the Lemke-Howson algorithm is an algorithm to find one NE through changing the zero settings little by little until the practical NE emerges.

The concept of *label* can be used to make the next analysis more clear. The labels for (4) and (5) are denoted as $L(P)$ and $L(Q)$, respectively, which are defined as follows:

$$L(P) = \{i : p'_i = 0, i \in \{1, ..., m\}\} \bigcup \{j : t'_j = 0, j \in \{1 = m+1, ..., m+n\}\} \tag{7}$$

$$L(Q) = \{i : r'_i = 0, i \in \{1, ..., m\}\} \bigcup \{j : q'_j = 0, j \in \{1 = m+1, ..., m+n\}\} \tag{8}$$

If $L(P) \bigcup L(Q) = \{1, 2, ..., m+n\}$, we will say that the pair of actions (p', q') is *completed labeled*. Apparently, the extraneous solution is completed labeled. When a completed labeled pair, other than extraneous solution, is found and the left two matrices are non-singular, we claim that the problem is solved. To solve the problem by enumeration, we need to first choose a start point, and the start point is completed labeled while the left matrices for the start point are non-singular. Then, for each step, we change the labels for $L(P)$ and $L(Q)$ in turns and guarantee that the left matrices are still non-singular, in other words, can be diagonalized, after the changes. If we get another completed labeled pairs, we find one pair of solutions for (4) and (5). In the process, the enumeration step is called *pivoting*.

To explain the pivoting, let us consider the following Table 1.

M	1	2	3	4	=
1	a_1	0	$\boxed{c_1}$	d_1	e_1
2	0	b_2	c_2	d_2	e_2

Table 1: A sample table M

Apparently, $L(M) = \{3, 4\}$ and the left matrix is non-singular. Next, we want to pivot matrix M on the element of $(1, 3)$, which is c_1. By multiplying c_2/c_1 for row 1 and subtracting the result from row 2 for matrix M, we can get Table 2.

M'	1	2	3	4	=
3	a_1	0	c_1	d_1	e_1
2	$-\frac{c_2}{c_1}a_1$	b_2	0	$d_2 - \frac{c_2}{c_1}d_1$	$e_2 - \frac{c_2}{c_1}e_1$

Table 2: A sample table M'

Now, $L(M') = \{1, 4\}$. Through pivoting matrix M on $(1, 3)$, we remove label 3, and add label 1. One requirement for pivoting is that every left matrix in each step should lead to a practical solution, which must be positive. In Table 2, this requires that $c_1 > 0$ and $c_1/e_1 > c_2/e_2$, which could lead that $e_2 - (c_2/c_1)e_1 > 0$.

The pivoting procedure can be summarized as follows:

In Algorithm 1, *cl* represent the complementary of the label, which is used to track the variables that can be calculated inside the matrix if the labeled variable are set to zero.

After understanding the algorithm for pivoting, the Lemke-Howson algorithm can be implemented as follows:

In Algorithm 2, the implementation of method *CalculateWithLabel*, which returns the value of p and q basing on the label status, is straightforward and ignored.

```
Data: pivot(M, k0, cl)
Result: M, k, cl
(m,n)=size(M);
k=k0;
max=0;
for i ← 1 to m do
    t = M_{i,k0}/M_{i,n};
    if t > max then
        ind=i;
    end
end
if max > 0 then
    swap(k, cl(ind));
    for i ← 1 to m do
        if i=ind then
            continue;
        end
        for j ← 1 to n do
            M_{i,j} = M_{i,j} − (M_{i,k0}/M_{ind,k0})M_{k,j};
        end
    end
end
return M, k;
```

Algorithm 1: Pivot matrix M for label $k0$

Data: LemkeHowson(U_1, U_2)
Result: p, q
(m,n)=size(U_1);
$P = [U_2^T, I, \mathbb{1}]$;
$Q = [I, U_1, \mathbb{1}]$;
LP=[1,...,m];
LQ=[m+1,...,m+n];
k=k0;
while *1* **do**
    Remove(LP,k);
    [P, k]=Pivot(P, k);
    Add(LP,k);
    if *k=k0* **then**
        break;
    end
    Remove(LQ,k);
    [Q, k]=Pivot(Q, k);
    Add(LQ,k);
    if *k=k0* **then**
        break;
    end
end
[p,q]=CalculateWithLabel(P,Q,LP,LQ);
return normal(p), normal(q);

Algorithm 2: The Lemke-Howson algorithm

3 An Example

As an example, let us consider the bimatrix game (U_1, U_2) where $U_1 = \begin{pmatrix} 4 & 12 & 8 & 6 \\ 16 & 8 & 12 & 8 \\ 10 & 8 & 10 & 9 \end{pmatrix}$ and $U_2 = \begin{pmatrix} 25 & 5 & 5 & 8 \\ 1 & 15 & 8 & 4 \\ 17 & 10 & 10 & 9 \end{pmatrix}$. Hence, we get matrix P and Q as in Table 3. Let us first pivot matrix P on label 1 as a start point. Please note that after each pivoting step, we artificially multiply one constant for the corresponding row to make all the elements shown as integer, which will not affect anything but merely for neat display purpose.

P	p_1	p_2	p_3	t_4	t_5	t_6	t_7	=
4	$\boxed{25}$	1	17	1	0	0	0	1
5	5	15	10	0	1	0	0	1
6	5	8	13	0	0	1	0	1
7	8	4	9	0	0	0	1	1

L(P)={1, 2, 3}

Q	r_1	r_2	r_3	q_4	q_5	q_6	q_7	=
1	1	0	0	4	12	8	6	1
2	0	1	0	$\boxed{16}$	8	12	8	1
3	0	0	1	10	8	10	9	1

L(Q)={4, 5, 6, 7}

Table 3: The initial matrix P and Q

The pivoting for matrix P brings Label 4, then we proceed the pivoting for matrix Q on Label 4, which brings Label 2. These procedures bring us Table 4.

P	p_1	p_2	p_3	t_4	t_5	t_6	t_7	=
1	25	1	17	1	0	0	0	1
5	0	$\boxed{74}$	33	−1	5	0	0	4
6	0	39	48	−1	0	5	0	4
7	0	92	89	−8	0	0	25	17

L(P)={2, 3, 4}

Q	r_1	r_2	r_3	q_4	q_5	q_6	q_7	=
1	4	−1	0	0	$\boxed{40}$	20	16	3
4	0	1	0	16	8	12	8	1
3	0	−5	8	9	24	20	32	3

L(Q)={2, 5, 6, 7}

Table 4: Matrix P and Q after the first round of pivoting

For matrix P in Table 4, due to the duplication of Label 2, we need to pivot matrix P on Label 2, which would bring us Label 5. Then, we pivot matrix Q on Label 5, which brings us Label 1. This round of pivoting generate Table 5.

After this round of pivoting, we get a completed label, which means a NE has been found. From Table 5, it is obvious that: $p' = [0.0378,\ 0.0541,\ 0]^T$ and $q' = [0.025,\ 0.075,\ 0,\ 0]^T$. Therefore, $p = normal(p') = [0.4118,\ 0.5882,\ 0]^T$ and $q = normal(q') = q' = [0.25,\ 0.75,\ 0,\ 0]^T$.

P	p_1	p_2	p_3	t_4	t_5	t_6	t_7	$=$
1	370	0	245	15	−1	0	0	14
2	0	74	33	−1	5	0	0	4
6	0	0	453	−7	−39	74	0	28
7	0	0	3550	−500	−460	0	1850	890

L(P)={3, 4, 5}

Q	r_1	r_2	r_3	q_4	q_5	q_6	q_7	$=$
5	4	−1	0	0	40	20	16	3
4	−32	48	0	640	0	320	192	16
3	−6	−11	20	0	0	20	56	3

L(Q)={1, 2, 6, 7}

Table 5: Matrix P and Q after the second round of pivoting

4 Some Extension

In Sect. 2, we assume that matrix U_1 and U_2 are utility matrices. However, the Lemke-Howson algorithm can also be used when U_1 and U_2 are cost matrices after certain simple adjustments. In this section, we shows how to use the Lemke-Howson algorithm to find one NE when the given matrices are cost matrices.

Following the same methodology of (1) in Sect. 2, it is easy to get (9) when U_1 and U_2 are cost matrices.

$$\begin{aligned} U_1 \cdot q - r &= v_1 \cdot \mathbb{1} \\ U_2^T \cdot p - t &= v_2 \cdot \mathbb{1} \end{aligned} \tag{9}$$

Let us use $\mathbb{M}$ to indicate a matrix, whose dimension decided by the context, and every elements in $\mathbb{M}$ is a relative large number M. Then $(\mathbb{M} - U_1) \cdot q + r = M \cdot \mathbb{1} - U_1 \cdot q + r = (M - v_1)\mathbb{1}$. In the same way, $(\mathbb{M} - U_2^T) \cdot p + t = (M - v_2)\mathbb{1}$. Suppose $U_1' = \mathbb{M} - U_1$, $U_2' = \mathbb{M} - U_2$, $v_1' = M - v_1$ and $v_2' = M - v_2$, then we can get (10).

$$\begin{aligned} U_1' \cdot q + r &= v_1' \cdot \mathbb{1} \\ U_2'^T \cdot p + t &= v_2' \cdot \mathbb{1} \end{aligned} \tag{10}$$

Apparently, as long as M is large enough, (10) can be solved with the Lemke-Howson algorithm, and the action pair $(p,\ q)$ is also the action pair for U_1 and U_2.

References

[1] C. Lemke and J. Howson Jr, "Equilibrium points of bimatrix games," *Journal of the Society for Industrial & Applied Mathematics*, vol. 12, no. 2, pp. 413–423, 1964.

[2] L. S. Shapley, "A note on the Lemke-Howson algorithm," in *Pivoting and Extension*, ser. Mathematical Programming Studies. Springer Berlin Heidelberg, 1974, vol. 1, pp. 175–189, 10.1007/BFb0121248. [Online]. Available: http://dx.doi.org/10.1007/BFb0121248