Midterm Evaluation Report

Tobenna Peter, Igwe

http://www.github.com/ptigwe/lh-vector/

10 July, 2012

1 Introduction

The aim of this report is to show the progress which has been made towards computing the Lemke-Howson path using a covering vector of Linear Complementarity Problems. Emphasis would be made on some of the problems and observations whilst undertaking the given project so far. This system was based on the implementation of Bernhard von Stengel's Lemke's algorithm on LCP. Most of the observations which have been made are based on the initial implementation of the Lemke's algorithm, the implementation of the Lemke-Howson algorithm uses some of the observations which were found to help initialise the pivoting system for the Lemke-Howson algorithm. Explanation of the testing procedure and further development would also be discussed.

1.1 LCP conversion

The conversion of a given $m \times n$ game (A,B) with the missing label k to a Linear Complementarity Problem with matrix M, vector q and covering vector d is as explained in Berhnard's documentation of the Lemke's algorithm.

$$q = \begin{pmatrix} -1 \\ -1 \\ \mathbb{0} \\ \mathbf{0} \end{pmatrix}, M = \begin{pmatrix} & & \mathbb{1}^\top & \\ & & & \mathbf{1}^\top \\ -\mathbb{1} & & & -A \\ & -\mathbf{1} & -B^\top & \end{pmatrix}, d = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \tag{1}$$

such that $d[k+2] = 0$. The value of $-A$ is computed by subtracting the ceiling of the maximum element in A from all elements in A, and finding the absolute value of each element. The similar computation is done for B.

$$-A = |A - m(A)|, \quad m(A) = \begin{cases} \lceil max(A) + 1 \rceil & \text{if } max(A) \text{ is an integer} \\ \lceil max(A) \rceil & \text{otherwise} \end{cases}$$

2 Observations

2.1 Lexicographic Degeneracy Resolution

Due to the lexicographic degeneracy resolution which was implemented in the function `int lexminvar(int enter, int *z0leave)` for the Lemke's algorithm, it was observed that for any missing label chosen for a game, the label $w2$ was always chosen as the first label to leave the basis. This is because after the LCP has been initialised, the degeneracy resolution first checks for the basic variables where the column of $z0$ is positive, thus eliminating the missing label. It then checks the minimum ratio of the $z0$ column with the RHS column, and eliminates all but the two payoff variables. After which $w2$ is chosen.

The rest of the path for missing labels that are player 2's strategies ($k > m$) give the correct path using the second method shown below, while missing labels that are player 1's strategies are not always correct. If $k \leq m$ (player 1's strategy) then after $w2$ leaves, the next label to leave is always $w(m+n+2)$, followed by $w1$, $w(k+2), \cdots, w(b+2), \cdots, z0$.

In some instances, especially when $m+n$ is not the best response to k (for player 1's strategies only), the path usually ends up having extra steps to rectify the problem since the wrong label was chosen, for example in $d \times d$ dual cyclic polytopes, the extra steps happened for only the odd strategies of player 1. Excluding the extra three pivot steps, in 4×4 dual cyclic polytopes, there were 6 extra steps, in 6×6 there were 10 extra steps, and in 8×8 there were 14 extra steps.

2.2 Lemke-Howson path

For a given $m \times n$ game, if for some missing label k it has a Lemke-Howson path of $a, b, \cdots, l, \cdots, i, j, k$ that is of length x, where label l is the vth label that is picked, it was observed that there were two different ways of setting up the initial pivots which would be equivalent to the LH path. The length of both representations of any LH path would have 3 extra steps due to pivoting the payoff variables ($w1$ and $w2$) as well as the $z0$ variable.

2.2.1 Method 1

This method involves pivoting one of the two payoff variables first ($w1$ or $w2$), followed by the best response to label k i.e. $w(a+2)$, after which the second payoff variable leaves the basis, and then the missing label follows ($w(k+2)$) before the rest of the path. For the rest of the path, if a label y is picked up, then the equivalent variable which leaves the basis would be $w(y+2)$, if $w(y+2)$ is not in the basis then $z(y+2)$ would be the leaving variable. The equivalent path for the above LH path is given below:

1. $w2$ if $k \leq m$ otherwise $w1$

2. $w(a+2)$

3. $w1$ if $k \leq m$ otherwise $w2$

4. $w(k+2)$

5. $w(b+2)$

$\vdots$

$v+2$. $w(l+2)$ if $w(l+2)$ is a basic variable otherwise $z(i+2)$

$\vdots$

$x+1$. $w(i+2)$ if $w(i+2)$ is a basic variable otherwise $z(i+2)$

$x+2$. $w(j+2)$ if $w(j+2)$ is a basic variable otherwise $z(j+2)$

$x+3$. $z0$

2.2.2 Method 2

This method is slightly different from the previous method, as the conditions for pivoting the payoff variables are changed. Firstly, one of the payoff variables leaves the basis, followed by the missing label ($w(k+2)$), then the second payoff variable, and the best response to the missing label ($w(a+2)$) and the rest of the path. The equivalent path for the above LH path is given below:

1. $w1$ if $k \leq m$ otherwise $w2$
2. $w(k+2)$
3. $w2$ if $k \leq m$ otherwise $w1$
4. $w(a+2)$
5. $w(b+2)$

$\vdots$

$v+2$. $w(l+2)$ if $w(l+2)$ is a basic variable otherwise $z(i+2)$

$\vdots$

$x+1$. $w(i+2)$ if $w(i+2)$ is a basic variable otherwise $z(i+2)$

$x+2$. $w(j+2)$ if $w(j+2)$ is a basic variable otherwise $z(j+2)$

$x+3$. $z0$

2.3 Exception

So far, from all the testing which has been done, all the LH paths for games work with the above initialization, except for the $\Gamma(d, 2d)$ dual cyclic polytope games where the path length was odd. In such cases, the computed path length always had an extra pivot (i.e. the path length is +4 instead of +3), just before $z0$ leaves the basis right at the very end, $z(k+2)$ leaves the basis. This is because the missing label is picked in the same tableau from which it was dropped, implying that $w(k+2)$ is in the basis which explains why $z(k+2)$ left the basis, another alternative which has been tested is to skip pivoting $w(k+2)$ when using method 1, which works and gives a path length which is +2 to the actual LH path, but this only works for paths of odd lengths.

3 Implementation

3.1 Problem Solutions

The problems which were specified in 2.1 were fixed by adding two initialization methods. Each one of these initialization methods pivots a maximum of 4 times, before the rest of the algorithm can be completed using the Lemke's algorithm. The initialization works by computing what the best response to the missing label is, and pending on the method for converting to an equivalent path which has been selected, performs the first 3 pivots if method 1 is selected, or the first 4 pivots if the second method was selected.

3.2 Versions

The software has been written in such a way as to allow for compilation using the GNUMP library for multiple precision integers, or a built-in definition of multiple precision integers (in `mp.h`) if the GNUMP library is not available on the system its being compiled on.

If the GNUMP library is available on the system, then the program should be compiled using the make target `inglh` (`make inglh`) for the program to make use of the GNUMP library. Otherwise the program should be compiled with the make target `inlh` (`make inlh`). These two make targets end up producing the same executable file `inlh` and they both have the same functionality.

4 Testing

Some bash scripts have been written to aid with the testing of the program, which can be found in the `test` folder. Amongst the tests cases considered are those for identity matrices, dual cyclic polytopes, and some extra games.

For testing the identity matrix games (`test/identity`), two programs were written, one (`genidentity`) which generates an *nxn* bimatrix identity game and sets the missing label to be 1, and the latter program (`genidenpath`) which generates the LH path for the *nxn* game, given a label k. The bash script takes in the value of n as its first argument. It then iterates for the value of k the missing label between the values of 1 and $2 \times n$, and for each label, the script uses the above mentioned programs to generate both the game input and the path is converted to one of the equivalant paths using one of the above methods. The LH path is then computed, and verified with the generated path.

For testing $\Gamma(d,d)$ dual cyclic polytopes games (`test/dualcyclic/dxd`) , a similar approach to that of the identity matrices was followed, except that the games for 2×2, 4×4, 6×6 and 8×8, have already been generated, the program which generates the LH path for the game is `lh-label`, and the script automatically checks the paths for all four games, and all the possible missing label for each game.

The dual cyclic polytopes $\Gamma(d,2d)$ (`test/dualcyclic/dx2d`) were tested by computing the paths and cross-checking that the path length match up with the pre-entered path lengths, which are specified in the path file for the specific game, and the tests take into consideration that the computed path length should be +3 longer than the actual path length, which is why some of the paths which are tested and have 4 extra pivots than expected fail the test as explained .

5 Future Development

So far, computing the LH path of a bimatrix game from the artificial equilibrium with the use of covering vectors has been completed, tested and documented., errors and solutions have also been explained in 2.3.

The next half of the project would involve restarting from an already known equilibrium point. While computing the first equilibrium e with the missing label k, the sequence of variables being pivoted would be stored. To restart from the equilibrium e with the missing label k', the new covering vector for label d would be used in the initial tableau (representing the artificial equilibrium), and the same sequence of pivots used to generate equilibrium e would be used to get the tableau to have the same set of basic variables as the tableau representing e. The covering vector d' is the gotten from the current tableau and inserted into the tablaeu for e, $z0$ then enters and the tableau is pivoted until $z0$ leaves and the new tableau when $z0$ leaves would represent the new equilibrium e'. This would be also tested agaisnst starting from the newly discovered equilibrium e' with a different missing label k''.

Once restarting from a known equilibrium has been completed, it would then be used to compute all reachable equilibrium by the LH algorithm. This would be acheieved by first computing the equilibrium points starting from the artificial equilibrium, and all the missing labels i (such that $1 \leq i \leq m+n$) to give equilibrium e_i. When restarting from an equilibrium e_i, all labels would be considered, except labels i and j for any label j if e_i equals e_j, because both labels would return back to the previous known equilibrium.

The equilibrium would be computed in a breadth first search manner, i.e. all labels would be counsidered in the artificial equilibrium first (level 1), then in all the equlibrium discovered from the artificial equilibrium (level 2), the equilibria dicovered from equilibria in level 2 and so on until for each equilibrium in a new level, the set of labels that can be considered is empty, i.e. no new equilibrium can be discovered.

This would be tested against similar kinds games as mentioned in the previous sections, as well as games with unique equilibrium and games with multiple equilibria. The LH path gotten from restarting the equilibrium would also be tested.

Doxygen comments and documentation would also be added to the code upon the completion of the project, and would be made available in the online repository. The Doxygen configuration file would also be made available along with the project files.