Summary Report

Tobenna Peter, Igwe
http://www.github.com/ptigwe/lh-vector

August 20, 2012

1 Introduction

Through out the course of this project, the computation of all equilibria reachable from the artificial equilibrium with the Lemke-Howson algorithm using covering vectors has been implemented. The project was built upon Berhnard von Stengel's implementation of Lemke's algorithm in Linear Complementarity Problems (LCP) from `http://www.github.com/stengel/ecta2002`. This report provides a brief summary of how system that was implemented, and some of the observations which were made.

2 Implementation

Minor changes were made to the underlying implementation of Lemke's algorithm which were mainly the way it represented rational numbers. The functionality of computing rational numbers having the numerator and denominator integers of large numbers was added using the GNUMP library as well as the multiple precision (`mp.h`) files provided by Berhnard.

The algorithm used to navigate the bi-partite graph of equilibria, is similar to the Breadth First Search(BFS) algorithm. Starting from the artificial equilibrium, for each missing label k in $1 \cdots m+n$ in a given $m \times n$ game, it computes the equilibrium by computing the value of the covering vector d which represents the missing label and following a set of pivoting rules. All computed equilibrium are stored as nodes in the bi-partite graph, and the labels are used as the edges linking the two equilibria (i.e the equilibrium started from and the computed equilibrium). For every equilibrium e found, the algorithm then tries to compute the equilibria which can be found from e. When restarting from e, for all missing labels k' in $1 \cdots m+n$, if there is an an edge from e to an already computed equilibrium using k', then restarting from e using the missing label k' is skipped to avoid backward computation. Otherwise, the covering vector d' which represents k' is computed and inserted into the tableau, z_0 is pivoted, and Lemke's algorithm finds a new equilibrium, e' which is added to the graph and linked with e using label k'. Note that all

equilibrium in the first set of equilibrium are restarted from before any equilibrium in the second set, and so on.

The algorithm terminates when there is a given set of equilibrium such that no new equilibrium can be computed from it (i.e. when restarting from such an equilibrium, for all missing labels k', there exists a link from the equilibrium to an already computed equilibrium).

3 Testing and Observations

The test cases used can be found in the folder `test`. These include identity matrix games and dual-cyclic polytope games. It also includes testing rational and decimal input system. A bash script which runs all the tests is available at `test/testall`, and accepts one argument, which is the size of the identity matrix game to be tested. Some other test cases were done using best response diagrams and can be found in the final report (`http://www.csc.liv.ac.uk/~cs0tpi/GSOC/final.pdf`).

While testing some of the games, it was noticed that in some cases the path length ended up being had 4 extra pivots instead of 3 extra pivots to the LH-path. This was mainly noticed with $\Gamma(d \times 2d)$ dual-cyclic polytope games, for missing labels whose path lengths were odd numbers.

4 Input and Output

4.1 Input

The program receives as its input a 2-player bi-matrix game in the format

$$\begin{aligned} m &= \quad ? \\ n &= \quad ? \\ A &= \\ & \quad \ddots \\ B &= \\ & \quad \ddots \end{aligned}$$

where A and B are the payoff matrices for player 1 and 2 respectively. For example, for a 2×2 identity matrix game, the input would be:

$$
\begin{array}{ll}
m = & 2 \\
n = & 2 \\
A = & \\
1 & 0 \\
0 & 1 \\
B = & \\
1 & 0 \\
0 & 1
\end{array}
$$

4.2 Output

The output provided can be split into two parts, the LH-path and the bi-partite graph.

As the program is computing the path from a given equilibrium, it first prints out the tableau representing the current equilibrium which it is starting from, followed by the set of pivot steps, and finally the tableau representing the current equilibrium. If the program is run with the verbose flag (`-v`), it prints out the tableau after every pivot step.

Following the information of the LH-path, is the information of the bipartite graph. This is of the following format:

$$
\begin{array}{rc}
\text{Equilibria discovered:} & x \\
\text{y-ve index:} & \\
\text{Eq[0]:} & \cdots \\
\text{Leads to:} & \cdots k\text{->}i \cdots \\
\vdots & \\
\text{Eq[y-1]:} & \cdots \\
\text{Leads to:} & \cdots k\text{->}i \cdots \\
 & \\
\text{z-ve index:} & \\
\text{Eq[0]:} & \cdots \\
\text{Leads to:} & \cdots k\text{->}i \cdots \\
\vdots & \\
\text{Eq[z-1]:} & \cdots \\
\text{Leads to:} & \cdots k\text{->}i \cdots
\end{array}
$$

where the program found x equilibria excluding the artificial equilibrium, of which y have a negative index, and z have a positive index. For each equilibrium, the strategy is printed out in form of fractions, containing both player 1 and 2's strategies. It also shows how the equilibria are linked together, and for every value of the missing label k, it shows the current equilibrium is linked to some other equilibrium with location i in the opposite list. For example, if we had 2->1, and the current equilibrium is `Eq[2]`, and it is negatively indexed, then the missing label 2 would lead us to the equilibrium `Eq[1]`.