THE PHILIPPINE REVIEW OF ECONOMICS
VOL. XLV NO. 1 JUNE 2008 P. 49-56

Determining the best evolution path for export industries using product spaces

Cyrus Paolo M. Buenafe
Hewlett-Packard Company

A deterministic method was developed in finding the best evolution path through which a country's particular export industry will evolve into another export industry. The problem was framed as a shortest-path problem particularly known in graph theory literature. A concept on a typical probability graph was defined and a shortest-path algorithm was formulated for this purpose. The data used by Hidalgo et al. [2007] were used and reconstructed using the probability graph concept to derive the maximum joint probability path of export industry evolution given an originating export industry and a target export industry.

JEL classification: A12, C63, O21
Keywords: product space, graph theory, shortest path, industry evolution

1. Introduction

A general theory of evolution of economies hinges on a reality that products and factors of production improve over time. Some products or skills that a country has fully mastered usually become their main exports, and this mastery is at the very heart of the theory of comparative advantage.

Hidalgo et al. [2007] developed a concept known as product space, a network of relatedness between products. It is claimed that countries can develop goods that are close to the ones they usually produce. By understanding how researchers and analysts form and understand product spaces, it is possible to understand how countries can evolve over time.

While Hidalgo et al. [2007] were able to present and structure the concept of a product space, knowing how a country should move from one industry to another remains unclear. The resulting product space was simply used to describe how various countries fared against each other, but it was not used to determine which industries need to be traversed to move from one industry to another with the highest rate of success.

2. Theoretical framework

2.1. Probability graphs and joint probability paths

We adjust and qualify the current concepts of graph theory in this particular problem. (See Grimaldi [2003] and Johnsonbaugh [2008], among many other references, for an excellent review on graph theory.) First of all, the type of graphs in scope in this analysis can be called probability graph. Such graphs have edges whose weights are real numbers between 0 and 1. It is possible to have a directed probability graph, although the present study assumes that product space has bidirectional edges. Operationally, just like any other graph, we represent a probability graph in an adjacency matrix, usually in a functional form $\gamma(x,y)$ equal to the weight of the edge connecting from x to y but $\gamma(x,y) \in [0,1]$. In bidirectional graphs, $\gamma(x,y) = \gamma(y,x)$.

The edges of a probability graph can be viewed as a discrete event, specifically the event of moving from one state to another. Also, probability theory defines the multiplicative principle: the probability of two disjointed events from happening is equal to the product of their individual probabilities. Hence, we define the joint probability path (JPP) as the set of edges linking from the source vertex to the target vertex in the probability graph and no vertex is repeated along the traversal. Formally, it can be defined in this manner:

$$JPP_{x,y} = \left\{ (n_0,n_1),(n_1,n_2),\cdots,(n_{i-1},n_i) \mid n_0 = x; n_i = y; n_q \neq n_r \forall 0 \leq q,r \leq i; \gamma(n_j,n_{j+1}) \neq 0 \forall 0 \leq j \leq i-1 \right\} \quad (1)$$

In the case of product spaces, an edge connotes a probability event of moving from one industry to another, consistent with what Hidalgo et al. [2007] specified. As such, moving from one industry to another can be viewed as the joint probability path between them. Further, their definition of proximity between industries, defined as $\phi_{x,y}$ in Hidalgo et al. [2007], is analogous to the concept of an adjacency matrix $\gamma(x,y)$.

2.2. The maximum joint probability path algorithm

One of the main optimization problems when faced with a probability graph would be finding the maximum joint probability path (MJPP) from one point to another. To solve this, we borrow the line of reasoning used by Dijkstra [1959] in formulating his shortest-path algorithm. The key differences between a typical shortest-path algorithm and the maximum joint probability-path algorithm are as follows:

- Different optimization goals: a shortest-path algorithm seeks for a minimum while the MJPP algorithm seeks for a maximum.
- Different accumulation operation: in a shortest-path algorithm, the path is a sum of the weights of the edges. In the MJPP algorithm, the path is a product of the weights of the edges, consistent with the multiplication principle in probability theory.

It is not accurate to completely use Dijkstra's algorithm, but certain principles that governed the mechanics of Dijkstra's algorithm can be adapted for the MJPP algorithm.

The algorithm to determine the maximum joint probability path from one point to another is as follows:

1. As with Dijkstra's algorithm, we use a "labeling" system in which each vertex S_k is initially labeled as $S_k:(0,-)$. With the source vertex S_0, we label it as $S_0:(1,0)$. The first element x in the label (x,y) signifies the maximum joint probability in reaching S_k, and the second element y signifies the previous vertex before reaching S_k. Let Q be the set of vertices in the probability graph.
2. We select S_j from Q such that the first element in its label is the maximum among those in Q. If more than one vertex satisfy this condition, we just choose any one of them at random. (If this is the first time this step is done, we set $S_j = S_0$.)
3. Look for all vertices that are neighbors of S_j (i.e., $\gamma(j,k)>0$ for any vertex S_k). Let *TEMP* be the product of the first element of S_j's label and $\gamma(j,k)$. We compare *TEMP* with *TEMP2* = first element of neighbor S_k's label. If *TEMP*< *TEMP2* , we replace the first element of neighbor S_k's label with *TEMP* and replace the second element of neighbor S_k's label with the value j.
4. S_j is removed from set Q and the process (nos. 2-4) is repeated until Q is empty.

The target vertex S_i will have the maximum joint probability stored as the first element of its own label. To determine the path, we backtrack from the target vertex S_i, going further back as specified by the second element of each vertex's label in the backtracking process. Eventually, we will reach the source vertex S_0, signifying the end of the backtracking process. Figure 1 below presents the Visual Basic code used to implement this maximum joint probability path for a given probability graph.

Figure 1. Visual basic implementation of MJPP

```
V(sour) = 0       'remove source vertex from priority queue
Chi(sour) = 1     'the probability of "evolving to itself" is always 1
nextver = sour
ctr = ctr - 1
While ctr > 0
    For i = 1 To 1252
        If (V(i) > 0) And (Prob(nextver, i) > 0) Then
            alternative = Chi(nextver) * Prob(nextver, i)
            If ((1 - Chi(i)) > (1 - alternative)) Then
                Chi(i) = alternative
                Prev(i) = nextver
            End If
        End If
    Next i
    strongest = 0
    For i = 1 To 1252
        If V(i) = 1 Then
            If strongest < Chi(i) Then
                strongest = Chi(i)
                nextver = i
            End If
        End If
    Next i
    V(nextver) = 0
    ctr = ctr - 1
Wend
```

3. Implementation results

From the original proximity data of Hidalgo et al. (2007) composed of 283,094 rows, we reconstructed the adjacency matrix $\gamma(x, y)$ for this probability graph. We came up with a 1,252 by 1,252 matrix, but there are only 774 rows/columns that are non-zero. The data were stored in a Microsoft Excel 2003 file, and a macro was created to program the MJPP algorithm. A typical run time to determine the maximum joint probability path of any two industries is under 10 seconds. Table 1 lists typical results of the maximum joint probability-path algorithm.

The appendix presents the complete Visual Basic code used for this purpose.

Table 1. The maximum joint probability paths of some pairs of industries

Source industry	*Target industry*	*Maximum joint probability*	*Industry evolution path*
"0011"	"8989"	0.230812	("0011") "ANIMALS OF THE BOVINE SPECIES, INCL. BUFFALOES, LIVE" --> ("7849") "OTHER PARTS & ACCESSORIES OF MOTOR VEHICLES" --> ("8989") "PARTS OF AND ACCESSORIES FOR MUSICAL INSTRUMENTS"
"7161"	"7761"	0.151209	("7161") "MOTORS & GENERATORS, DIRECT CURRENT" --> ("7368") "WORK HOLDERS, SELF-OPENING DIEHEADS & TOOL HOLDERS" --> ("7761") "TELEVISION PICTURE TUBES, CATHODE RAY"
"2231"	"7821"	0.074011	("2231") "COPRA" --> ("6254") "TYRES, PNEUM. NEW. OF A KIND USED ON MOTOR/BICYCLES" --> ("7821") "MOTOR VEHICLES FOR TRANSPORT OF GOODS/MATERIALS"
"3351"	"8812"	0.137309	("3351") "PETROLEUM JELLY AND MINERAL WAXES" --> ("7129") "PARTS OF THE POWER UNITS OF 712.6-" --> ("8812") "CINEMATOGRAPHIC CAMERAS, PROJECTORS, SOUND-REC, PAR"
"5111"	"8812"	0.168056	("5111") "ACYCLIC HYDROCARBONS" --> ("8812") "CINEMATOGRAPHIC CAMERAS, PROJECTORS, SOUND-REC, PAR"
"5111"	"9110"	0.020833	("5111") "ACYCLIC HYDROCARBONS" --> ("9110") "POSTAL PACKAGES NOT CLASSIFIED ACCORDING TO KIND"
"6121"	"6411"	0.281655	("6121") "ARTICLES OF LEATHER OR OF COMPOSITION LEATHER" --> ("2518") "CHEMICAL WOOD PULP, SULPHITE" --> ("6411") "NEWSPRINT"

4. Conclusion and recommendations

At first glance, it may seem strange to talk about economics to an audience highly oriented in information technology (IT) or to talk about IT to an audience highly oriented in economics. However, based on this work so far, it is possible to utilize computer science concepts to answer questions in the field of economics. This provides a different tool besides calculus, statistics, and econometrics in solving optimization problems. This presents a bold challenge to look into disciplines not "traditionally" handled by economics.

An extension of the work can be an exhaustive approach: looking for all of the maximum joint probability paths in the product space. From the results given henceforth, it is possible to determine the "important" industries based on the characteristics of the results.

The concept of joint probability paths can be used in other economic areas. For instance, to determine world foreign-exchange equilibrium, one can produce some sort of an arbitrage graph, where the edges are the exchange rates between two countries represented as vertices.

References

Dijkstra, E.W. [1959] "A note on two problems in connexion with graphs", *Numerische Mathematik*.

Grimaldi, R.P. [2003] *Discrete and combinatorial mathematics: an applied introduction*. Fifth Edition. Reading, MA: Addison-Wesley.

Hidalgo, C.A., B. Klinger, A.-L. Barabasi, and R. Hausmann [2007] "The product space conditions the development of nations", *Science* **317**: 482-487.

Johnsonbaugh, R. [2008] *Discrete mathematics*. Seventh Edition. Upper Saddle River, NJ: Prentice-Hall.

Appendix

The following code listing presents the entire program used to generate the maximum joint probability path, given a source industry and a target industry. The program is generally divided into three parts: creating the probability graph, computing for the maximum joint probability path (which has been highlighted in Figure 1), and displaying the sequence of industries based on the maximum joint probability path.

```
Dim Prob(1252, 1252), Chi(1252) As Double
Dim V(1252), Prev(1252) As Integer

Private Sub Shortest Path()
' there are 283094 rows
' there are 1249 products/industries (added 3 more = 1252 total)
' max 65536 rows in an Excel worksheet
' for last set, end at row 20950
' 20070917 - determined that there are only 774 valid products/
industries


starttime = Timer
For i = 2 To 17 Step 5
    For j = 1 To 65536
        px = Sheet5.Cells(j, i)
        py = Sheet5.Cells(j, i + 1)
        edgeweight = Sheet5.Cells(j, i + 2)
        Prob(px, py) = edgeweight
        Prob(py, px) = edgeweight
    Next j
Next i
For i = 22 To 22
    For j = 1 To 20950
        px = Sheet5.Cells(j, i)
        py = Sheet5.Cells(j, i + 1)
        edgeweight = Sheet5.Cells(j, i + 2)
        Prob(px, py) = edgeweight
        Prob(py, px) = edgeweight
    Next j
Next i

ctr = 0
For i = 1 To 1252
    V(i) = Sheet8.Cells(i, 3)
    ctr = ctr + V(i)
    Prev(i) = 0
    Chi(i) = 0
Next i

sourind = Sheet8.Cells(5, 6)
destind = Sheet8.Cells(6, 6)


For i = 1 To 1252
    If Sheet8.Cells(i, 1) = sourind Then sour = i
    If Sheet8.Cells(i, 1) = destind Then dest = i
Next i
'start of shortest-path algorithm
V(sour) = 0      'remove source vertex from priority queue
Chi(sour) = 1    'the probability of "evolving to itself" is always 1
nextver = sour
ctr = ctr - 1
While ctr > 0
```

```
  For i = 1 To 1252
        If (V(i) > 0) And (Prob(nextver, i) > 0) Then
            alternative = Chi(nextver) * Prob(nextver, i)
            If ((1 - Chi(i)) > (1 - alternative)) Then
                Chi(i) = alternative
                Prev(i) = nextver
            End If
        End If
    Next i
    strongest = 0
    For i = 1 To 1252
        If V(i) = 1 Then
            If strongest < Chi(i) Then
                strongest = Chi(i)
                nextver = i
            End If
        End If
    Next i
    V(nextver) = 0
    ctr = ctr - 1
Wend


nextver = dest
actualpath = "(" + Sheet8.Cells(dest, 1) + ") " + Sheet8.Cells(dest, 2)
While nextver <> sour
    actualpath = "(" + Sheet8.Cells(Prev(nextver), 1) + ") " +
Sheet8.Cells(Prev(nextver), 2) + " --> " + actualpath
    nextver = Prev(nextver)
Wend
Sheet8.Cells(7, 6) = Chi(dest)
Sheet8.Cells(8, 6) = actualpath
Sheet8.Cells(9, 6) = Timer - starttime

End Sub
```