**Hungarian Algorithm**

By – Ishan Kaushik

CSCI – 174

FALL -2014

**Introduction –**

The **Hungarian method** is a [combinatorial optimization](http://en.wikipedia.org/wiki/Combinatorial_optimization) [algorithm](http://en.wikipedia.org/wiki/Algorithm) that solves the [assignment problem](http://en.wikipedia.org/wiki/Assignment_problem) in [polynomial time](http://en.wikipedia.org/wiki/Polynomial_time) and which anticipated later [primal-dual methods](http://en.wikipedia.org/w/index.php?title=Primal-dual_method&action=edit&redlink=1). It was developed and published by [Harold Kuhn](http://en.wikipedia.org/wiki/Harold_Kuhn) in 1955, who gave the name "Hungarian method" because the algorithm was largely based on the earlier works of two Hungarian mathematicians: [Dénes Kőnig](http://en.wikipedia.org/wiki/D%C3%A9nes_K%C5%91nig) and [Jenő Egerváry](http://en.wikipedia.org/wiki/Jen%C5%91_Egerv%C3%A1ry).

[James Munkres](http://en.wikipedia.org/wiki/James_Munkres) reviewed the algorithm in 1957 and observed that it is [(strongly) polynomial](http://en.wikipedia.org/wiki/Time_complexity#Strongly_and_weakly_polynomial_time). Since then the algorithm has been known also as the **Kuhn–Munkres algorithm**or **Munkres assignment algorithm**. The [time complexity](http://en.wikipedia.org/wiki/Computational_complexity_theory#Time_and_space_complexity) of the original algorithm was O(n^4), however [Edmonds](http://en.wikipedia.org/wiki/Jack_Edmonds) and [Karp](http://en.wikipedia.org/wiki/Richard_Karp), and independently Tomizawa noticed that it can be modified to achieve an O(n^3) running time. [Ford](http://en.wikipedia.org/wiki/L._R._Ford,_Jr.) and [Fulkerson](http://en.wikipedia.org/wiki/D._R._Fulkerson) extended the method to general transportation problems. In 2006, it was discovered that [Carl Gustav Jacobi](http://en.wikipedia.org/wiki/Carl_Gustav_Jacobi)had solved the assignment problem in the 19th century, and the solution had been published posthumously in 1890 in Latin.

**Matrix interpretation**

Given n workers and tasks, and an n×n matrix containing the cost of assigning each worker to a task, find the cost minimizing assignment.

First the problem is written in the form of a matrix as given below

\begin{bmatrix}
a1 & a2 & a3 & a4\\
b1 & b2 & b3 & b4\\
c1 & c2 & c3 & c4\\
d1 & d2 & d3 & d4\end{bmatrix}

where a, b, c and d are the workers who have to perform tasks 1, 2, 3 and 4. a1, a2, a3, a4 denote the penalties incurred when worker "a" does task 1, 2, 3, 4 respectively. The same holds true for the other symbols as well. The matrix is square, so each worker can perform only one task.

**Step 1**

Then we perform row operations on the matrix. To do this, the lowest of all *ai* (i belonging to 1-4) is taken and is subtracted from each element in that row. This will lead to at least one zero in that row (We get multiple zeros when there are two equal elements which also happen to be the lowest in that row). This procedure is repeated for all rows. We now have a matrix with at least one zero per row. Now we try to assign tasks to agents such that each agent is doing only one task and the penalty incurred in each case is zero. This is illustrated below.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | a2' | a3' | a4' |
| b1' | b2' | b3' | 0 |
| c1' | 0 | c3' | c4' |
| d1' | d2' | 0 | d4' |

The zeros that are indicated as 0' are the assigned tasks.

**Step 2**

Sometimes it may turn out that the matrix at this stage cannot be used for assigning, as is the case in for the matrix below.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | a2' | a3' | a4' |
| b1' | b2' | b3' | 0 |
| 0 | c2' | c3' | c4' |
| d1' | 0 | d3' | d4' |

In the above case, no assignment can be made. Note that task 1 is done efficiently by both agent a and c. Both can't be assigned the same task. Also note that no one does task 3 efficiently. To overcome this, we repeat the above procedure for all columns (i.e. the minimum element in each column is subtracted from all the elements in that column) and then check if an assignment is possible.

In most situations this will give the result, but if it is still not possible then we need to keep going.

**Step 3**

All zeros in the matrix must be covered by marking as few rows and/or columns as possible. The following procedure is one way to accomplish this:

First, assign as many tasks as possible.

* Row 1 has one zero, so it is assigned. The 0 in row 3 is crossed out because it is in the same column.
* Row 2 has one zero, so it is assigned.
* Row 3's only zero has been crossed out, so nothing is unassigned.
* Row 4 has two uncrossed zeros. Either one can be assigned (both are optimum), and the other zero would be crossed out.

Alternatively, the 0 in row 3 may be assigned, causing the 0 in row 1 to be crossed instead.

|  |  |  |  |
| --- | --- | --- | --- |
| 0' | a2' | a3' | a4' |
| b1' | b2' | b3' | 0' |
| 0 | c2' | c3' | c4' |
| d1' | 0' | 0 | d4' |

Now to the drawing part.

* Mark all rows having no assignments (row 3).
* Mark all (unmarked) columns having zeros in newly marked row(s) (column 1).
* Mark all rows having assignments in newly marked columns (row 1).
* Repeat for all non-assigned rows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| × |  |  |  |  |
| 0' | a2' | a3' | a4' | × |
| b1' | b2' | b3' | 0' |  |
| 0 | c2' | c3' | c4' | × |
| d1' | 0' | 0 | d4' |  |

Now draw lines through all marked columns and **unmarked** rows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| × |  |  |  |  |
| 0' | a2' | a3' | a4' | × |
| b1' | b2' | b3' | 0' |  |
| 0 | c2' | c3' | c4' | × |
| d1' | 0' | 0 | d4' |  |

The aforementioned detailed description is just one way to draw the minimum number of lines to cover all the 0s. Other methods work as well.

**Step 4**

From the elements that are left, find the lowest value. Subtract this from every unmarked element and add it to every element covered by two lines.

Repeat steps 3–4 until an assignment is possible; this is when the minimum number of lines used to cover all the 0s is equal to the max(number of people, number of assignments), assuming dummy variables (usually the max cost) are used to fill in when the number of people is greater than the number of assignments.

Basically you find the second minimum cost among the remaining choices. The procedure is repeated until you are able to distinguish among the workers in terms of least cost.

References –

<http://en.wikipedia.org/wiki/Hungarian_algorithm>

<http://www.wikihow.com/Use-the-Hungarian-Algorithm>

<https://www.youtube.com/watch?v=BUGIhEecipE>