mbm

by Tedy Setiadi

Submission date: 25-Aug-2020 07:27PM (UTC+0700)
Submission ID: 1373856355
File name: MBM.pdf (563.57K)
Word count: 2398
Character count: 10312

The solution of the Maximum Weighted Matching problem (MWM) using Primal Dual Algorithm

Tedy Setiadi [1]

[1] Informatic Engineering University of Ahmad Dahlan Yogyakarta
[1]Email : tedyasni@gmail.com

Abstract. The issue of matching problem is relatively simple when determining the maximum cardinality matching (MSM) and occurs in two disjoint sets (bipartite graph). But it would be a complex problem when the MWM that occurs in the general graph (not necessarily bipartite).

In this research, has developed software to help resolve the issue of MBM on a general graph using the primal-dual algotitma of combinatorial optimization problems ..

***Keywords**: MBM, General Graph, Combinatorics, primal dual algorithm*

1 Introduction

In a graph G = (V,E) the number of arcs that meet at node i is called the degree of node i . The issue of matching is the selection of a subset arc based on the node degree limit. A matching M $\subseteq$ E is a subset bow to the nature of each node in the subset graph G (M) = (V,M) are connected by no more than one arc. The simplest case is a 1-matching (or so-called matching only) . Every graph G has a matching M = 0. Generalization of 1- matching is a b - matching where node i associated with no more than bi arc , where bi a positive integer [1].
Classic application of the matching problem is the installation of loose objects from two sets [2]. Suppose there are four employees are p1 , p2 , p3 and p4 to fill six positions j1 , j2 , j3 , j4 , j5 , and j6 . When p1 qualified employees to fill positions j2 or j5 . P2 employees can fill positions j2 or j5 . P3 employee can fill the positions j1 , j2 , j3 , j4 , or J6, p4 employees to fill positions j2 or j5. The problem that arises is it possible to assign all employees at any positions that meet the qualifications , if not then how the maximum number of positions that can be filled by employees who are given . The problem of matching shape known as assignment (assignment) , and determine the maximum number of cardinality matching problem .
Examples of other forms of matching is given by [3] that the theory of marriage (Marriage Theorem). There are n people n grooms and brides who are getting married. We want to set n more desirable marriage and the marriage took place only for men and women who have known each other . The question is whether it is possible to occur n the marriage . This issue is known to form a complete matching (perfect matching) .
A matching is said to have weight (weighted matching) if the arc has weight . [4] gives examples of applications in the form of weighted matching problems postman (postman problem). Given a graph G with weights on the arcs , the postman problem is to find a minimum weight set of arcs that are added to G to produce multigraph eulier contains a circuit (i.e a path (walk) contains any arc

MG covered exactly once). Euler circuit on MG translate into minimum weight in G where each arc visited at least once , resulting in the generation of minimum weight from being sent to the postman .
The issue of matching can be viewed as a combinatorial optimization problem One of the most widely used tools in combinatorics is the completion of linear programming problems (linear programming) . The issue of matching in the field of linear programming models belonging to programa integer .
Formulas 0-1 integer programming of weight b - matching is

$$\begin{aligned} maks\ \ & cx \\ & Ax \le b \\ & x \in B^n \end{aligned} \qquad (1)$$

where A is the node-arc matrix connected graph, | E | = n, and xe = 1 if there is matching.
In the MWM problem, then the form of integer programming are (

$$(WM) \qquad \begin{aligned} & maks \sum_{e\in E} c_e x_e \\ & \sum_{e\in\delta(v)} x_e \le 1 \text{ for every } v \in \mathrm{E} \\ & x \in B^n \end{aligned} \qquad (2)$$

2 Library studies

Given primal-dual algorithms for linear programming

$$\begin{aligned} & maks \sum_{e\in E} c_e x_e \\ & \sum_{e\in\delta(v)} x_e \le 1 \quad \text{for every } v \in \mathrm{V} \\ & \sum_{e\in E(U)} \mathrm{x}_e \le \left[\frac{|U|}{2}\right] \quad \text{for every odd set } U \subseteq V \\ & x \in R_+^n \end{aligned} \qquad (2)$$

and prove that the solution is to blend any objective function vector c, which is the solution to the maximum weight matching. Assumed ce> 0 for e $\in$ E, if ce $\le$ 0 result no optimal solution with xe = 0.
Given matching M, xe = for e $\in$ M, and xe = 0 for the other, then

$$c_e' = \sum_{v:e\in\delta(v)} \pi_v + \sum_{\text{odd sets U : e}\in\text{(U)}} y_u - c_e$$

Complementary slackness conditions for linear program is :

1.1. $\mathrm{c}_e' x_e = 0 \text{ for } \mathrm{e} \in \mathrm{E}\ (\mathrm{c}_e' = 0 \text{ or } \mathrm{e} \notin \mathrm{M})$

1.2. $\left(\| U \,|/\, 2\,| - \sum_{e\in E(U)} x_e\right) y_u = 0, \text{for every odd set } U\ (y_u = 0 \text{ or } M \cap E(U) = \| U \,|/\, 2 |)$

Primal-dual algorithm keeping primal-dual feasibility and also the condition of 1.1. and 1.2., the optimal solution is reached when 1.3. fulfilled..
Initialization of an integrated solution feasible primal and dual that satisfies 1.1. and 1.2. is given by:

$$\begin{aligned} & x_e = 0, \text{for } e \in E \\ & y_u = 0, \text{for every odd set } U \\ & \pi_v = \tfrac{1}{2} \max_{e\in E} c_e, \text{for } v \in V \end{aligned} \qquad (1.3)$$

For $c'_{e'} = 0$ every $e' \in E$ then $c_{e'} = \max_{e \in E} c_e$

Algorithm of Maximum Weighted Matching

1. Initialization : Start with the primal dual solution given by (1.3) . Suppose E ' = { e $\in$ E : c'e = 0) , G ' = (V , E ') , $\tilde{G}'$= G ' , $\tilde{M}$ = M = ϕ and $\tilde{F}'$ = ϕ
2. Step 1 : Continue to construct alternating forest . If the path augmentation is found then to step 2 . If not then to step 3 .
3. Step 2 (Augmentation) : Update of the primal solution M and expand all psedonode B (U) with yu = 0 . Update the base of the rest of the blossoms . subgraph with restrictions reduced the matching equation , if π_v = 0 for all vertices are open , the primal and dual solution is optimal applicable . If not , specify $\tilde{F}$ = ϕ and to step 1 .
4. Step 3 (Dual Change) : Apply a dual change given by (3.5) and (3.6) below . If $\pi_v = 0$ for all vertices open , primal and dual solutions are already optimal.Jika not apply , renew and expand all psedonode B (U) with yu = 0 . If e (u,v) was added where u and v are both even and is at a different tree from , then identifying the path augmentation and to step 2 . If not , keep intact , and returned the first step .

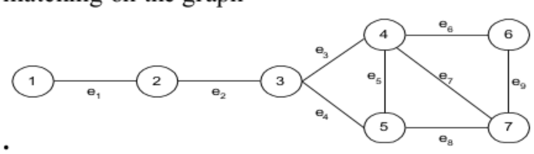
Theorem: Weighted Matching algorithm find the integral optimal solution for (1.2) and also the optimal solution dual to (1.3). Complexity is O (m2 n).

Proof: an integral primal solution is maintained because any solution matching. When the algorithm stops, the primal and dual solutions are both feasible and satisfy complementary slackness.

Working between successive dual change is O (n). By proposition 3.4. The maximum number of changes between a dual augmentation is O (m), and the number of augmentation is O (m). Finally, after the change of the dual p, that π, y and c 'associated with denominator 2k for round k, $0 \le k \le p$. Therefore, the number of calculations remained within the limits of polynomials.

Case in point:

Given the following weighted graph, then will we find the maximum weight matching on the graph



.

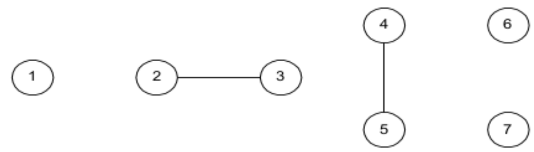
c = (c_{e1}, c_{e2},....., c_{e9}) = (8 9 8 7 9 4 5 2 1)

1. initialization

π_v = 4.5 for every v $\in$ V

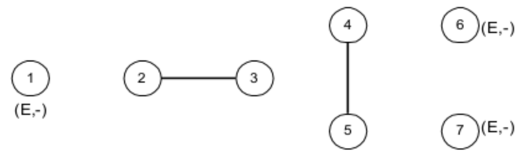
y_u = 0 for every u

c' = (1 0 1 2 0 5 4 7 8)



equality constrained subgraph

2. Equality constrained subgraph and labelling M = $\{e_2, e_5\}$



3. Dual Change

   $\delta_1 = \min(\pi_1, \pi_6, \pi_7) = 4.5$ $\quad\delta_2 = \infty$ $\quad\delta_3 = 1/2c'_{e9} = 4$

   $\delta_4 = \min(c'_{e1}, c'_{e6}, c'_{e7}, c'_{e8}) = 1$ $\quad\delta = \delta_4 = 1$

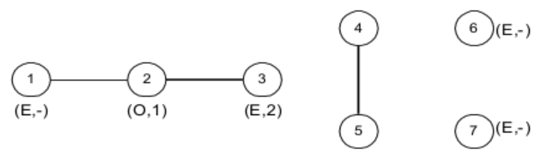
   $\pi = (3.5 \quad 4.5 \quad 4.5 \quad 4.5 \quad 4.5 \quad 3.5 \quad 3.5)$

   $y_u = 0$ for every U

   $c' = (0 \quad 0 \quad 1 \quad 2 \quad 0 \quad 4 \quad 3 \quad 6 \quad 6)$

   e_1 add to subgraph with Equality constrained

4. Subgraph with Equality constrained and labelling



4. Dual Change

   $\delta_1 = 3.5$ $\quad\delta_2 = \infty$ $\quad\delta_3 = 3$

   $\delta_4 = \min(1 \ 2 \ 4 \ 3 \ 6) = c'_{e3} = 1$ $\quad\delta = \delta_4 = 1$

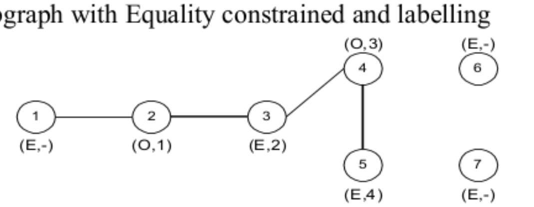
   $\pi = (2.5 \quad 5.5 \quad 3.5 \quad 4.5 \quad 4.5 \quad 2.5 \quad 2.5)$

   $y_u = 0$ for every U

   $c' = (0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 3 \quad 2 \quad 5 \quad 4)$

   e_3 add to subgraph with Equality constrained

6. Subgraph with Equality constrained and labelling



5. Dual Change

   $\delta_1 = 2.5$ $\quad\delta_2 = \infty$ $\quad\delta_3 = ½ \min(1 \quad 5 \quad 4) = 1/2$

   $\delta_4 = \infty$ $\quad\delta = \delta_3 = 1/2$

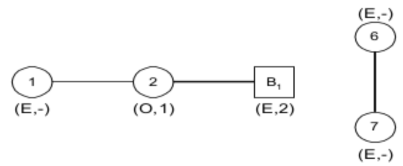
   $\pi = (2 \quad 6 \quad 3 \quad 5 \quad 4 \quad 2 \quad 2)$

   $y_u = 0$ for every U

   $c' = (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 3 \quad 2 \quad 4 \quad 3)$

e_4 add to subgraph with Equality constrained

8. Subtract subgraph with Equality constrained and labelling

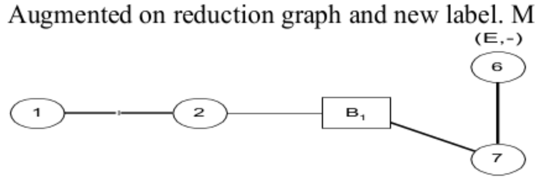


U = { 3,4,5}
B_1 = B(U)
b(U) = 3

9. Dual Change

$\delta_1 = 2$ $\delta_2 = \infty$ $\delta_3 = \frac{1}{2}\min\{c'_{ei}\} = 1$, i = 6,7,8,9
$\delta_4 = \infty$ $\delta=\delta_3 = 1$
$\pi = (1 \quad 7 \quad 2 \quad 4 \quad 3 \quad 1 \quad 1)$
$y_u = 2$ for every U = { 3,4,5}, $y_u = 0$ for others
$c' = (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 2 \quad 1)$
e_7 add to subgraph with Equality constrained

10. Augmented on reduction graph and new label. M = { e_1, e_2, e_3}



$b(U_1) = 4$

6. Dual Change

$\delta_1 = \pi_6 = 1$ $\delta_2 = \infty$ $\delta_4 = \infty$
$\delta_4 = \min\{c_{e6}, c_{e9}\} = 1$
$\pi = (1 \quad 7 \quad 2 \quad 4 \quad 3 \quad 0 \quad 1)$
$y_u = 2$ for U_1 = { 3,4,5}, $y_u = 0$ for others

12. Optimal Solution

Primal : $x_{ei} = 1$ for i = 1,4,7, and $x_{ei} = 0$ for others i
Dual : $\pi = (1 \quad 7 \quad 2 \quad 4 \quad 3 \quad 0 \quad 1)$
$y_{u1} = 2$ for U_1 = { 3,4,5}, $y_u = 0$ for others

3 Design of Sofware MWM

3.1 Design of Data structure

1. Input graph presented in the form of a weighted graph with a adjency matrix.

```
Const NMax  = 100;
Type Matrix = array[1..NMax,1..NMax] of real;
```

2. Node is represented by the type of structure and array. Attribute node consists of weights, the status is even, odd or not labeled, ismatched attribute indicates

whether the node is open or not, and connect indicate which nodes are connected.

```
   Type
         tNode = record
               phi : real;
               evenodd : char;
               ismatched : boolean;
               connect : integer;
         end;
```

3. Arc represented the type of structures and array. Attributes consist arc weights, the first node and a second node connected two such arcs, ismatched indicate whether or not a matching bow.

```
   Type
         tEdge = record
               first,second : integer;
               ismatched : boolean;
         end;
```

3.2 Design of Algorithm

```
Procedure Initialization
{ determine the initial value of node weights, weight bow on MBM }
Declaration
    i, a, b : integer
    w : real
algorithm
    w ← -999
    for i ← 1 to edgecount do
        if weight[i].edge>w
             then w ← weight[i].edge
    for i←1to nodecount do node[i].phi←w/2
    for i ←1 to edgecount do
        if  weight[i].edge=0 then
          a←edge[i].first,b← edge[i].second
          edge[i].ismatched←true,
          node[a].ismatched←true,
          node[b].ismatched ← true

Procedure CreateLink (node : integer)
{ create a path in a graph MBM recursively }
Declaration
    i, next: integer
algorithm
    for i ← 1 to edgecount do
        if  weight[i].edge = 0 then
             if e[dge[i].first=node then
                   next ← edge[i].second
          elseif e[dge[i].second=node then
                   next ← edge[i].first
          if next >= o then
             if nodes[next].status='x' then
               if nodes[node].status='E'
```

```
              then
                 nodes[node].status←'O'
             else
                 nodes[node].status ←'E'
           nodes[next].connect ←node
           CreateLink(next)

Function Is Blossom (a,b: integer) : boolean
{true if at least find one blossom }
Declaration
    i, next: integer
algorithm
    IsBlossom ← false
    for i ← 1 to edgecount do
        if  nodes[a].status='E' and
        nodes[b].status='E' and
        edges[i].weight=0 then
          Isblossom  ← true

Procedure DualChange
{ determine the value of the dual node and arc weights update SKP
}
Declaration
    i: integer
     d1,d2,d3,d4,dual : real
algorithm
    d1,d2,d3,d4 ← infinite
    for i ← 1 to nodecount do
        if  nodes[i].status= 'O' then
          d1← nodes[i].phi
        if IsBlossom(i)  then
               d2← edge[i].second
    for i ← 1 to edgecount do
        if nodes[edge[i].first].status ='O'
        and odes[edge[i].second].status
          ='O' then
             d3 ← edges[i].weight/2
          if nodes[edge[i].first].status
          ='E'and nodes[edge[i].second].status ='X' then
             d4 ← edges[i].weight
        dual ← min (d1,d2,d3,d4)
    for i ← 1 to nodecount do
        if  nodes[i].status= 'E' then
          nodes[i].phi← nodes[i].phi - dual
        elseif nodes[i].status= 'O' then
          nodes[i].phi← nodes[i].phi + dual
    for i ← 1 to edgecount do
        if nodes[edges[i].first].status=
         'E' and    nodes[edges[i].first].status='X'then
      edges[i].weight←dges[i].weight - dual
        if nodes[edges[i].first].status=
         'X' and
         nodes[edges[i].first].status='E'
```

```
then
edges[i].weight←edges[i].weight-
  dual
if nodes[edges[i].first].status=
'E' and
   nodes[edges[i].first].status='E' then
     edges[i].weight←edges[i].weight-2 * dual
```

4 Conclusion

The solution of the problem MBM with primal dual algorithm Edmond deliver the optimum solution when no node is open or all open vertices weighted zero. In addition, when finding sirkuti odd (blossom) requires special care, which need to be depreciated once again which will be described with the augmentation. It would be quite complicated and complex problem, which is encountered when a relatively large blossom.

5 Reference

[1] Chegireddy C.R. dan Hamacher H.W : "Algorithms for Finding K-Best Perfect Matchings", Discrete Applied Mathematics 18, North-Holland, 1987

[2] Deo Narsingh. : "Graph Theory : with applications to Engineering and Computer Science", Prentice Hall., New Delhi, 1995.

[3] Lovasz L. dan plummer M.D. : "Matching Theory", Annals of Discrete Mathematics, North-Holland, 1986.

[4] Nemhauser G. & Wolsey L.: "Integer and Combinatorial Optimization", John Wiley & Sons, Inc. 1988.

mbm

ORIGINALITY REPORT

5%	3%	5%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Kenjiro Takazawa. "A weighted even factor algorithm", Mathematical Programming, 10/2008 Publication	1%
2	"Frontiers in Algorithmics", Springer Science and Business Media LLC, 2009 Publication	1%
3	David F. Crouse. "On implementing 2D rectangular assignment algorithms", IEEE Transactions on Aerospace and Electronic Systems, 2016 Publication	1%
4	www.ijmttjournal.org Internet Source	1%
5	mafiadoc.com Internet Source	<1%
6	www.springerprofessional.de Internet Source	<1%
7	Ellis L. Johnson. "On Four Problems in Graph Theory", SIAM Journal on Algebraic and	<1%

Discrete Methods, 1987
Publication

8	link.springer.com Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		