Proceedings of the Twenty-Eighth RAMP Symposium
Niigata University, Niigata, October 13–14, 2016

ネットワークフローのモデルで論文査読者割当の問題を考える

Paper review assignment problem under the minimum cost flow network model

Xuefeng LIU[1*], Yuan LIU[2†], and Zhouwang Yang[2‡]

Abstract For the paper review assignment problem, i.e., to assign papers to proper reviewers, most of the existing algorithms regard it as an integer programming problem, which is however a class of NP-hard problem. In this paper, the paper review assignment problem is considered under the minimum cost flow network model. The running time of network simplex algorithm to solve the review assignment is no worse than $O(nm(n+m)^2 \log(n+m))$ (n:paper number, m:reviewer number). Thus the problem can be solved in strongly polynomial time. The efficiency of our proposed algorithm is demonstrated by computation results.

1. Introduction

In this paper, we are considering a practical problem of paper review assignment for academic conferences. The scenario of such a problem can be described as follows. Papers are submitted to a conference and there are reviewers to review these papers. To find proper reviewers from the reviewer database for each paper, the conference chair will first ask the reviewers to do a bidding for the papers, i.e., each reviewer selects the papers that he wants to review, the ones he may agree to review and the ones he does not want to review. By also taking into account other factors, for example, the interest conflict between reviewers and papers, the conference chair decides how to assign papers to reviewers.

For the paper review assignment problem, the existing algorithms usually deal with it in an integer programming model with properly selected objective function. However, since the integer programming problems is a NP-hard class. The computing time will dramatically increase as the scale of the problem increases. In this paper, we consider the paper review assignment problem in the min-cost flow (MCF) network model, which is a model problem in the field of operations research. The theoretical analysis shows that a MCF network problem can be solved in strongly polynomial runtime; see the survey in, e.g., [4]. Besides theoret-

1 Graduate School of Science and Technology, Niigata University, 8050 Ikarashi 2 no-cho, Nishi-ku, Niigata City, Niigata, 950-2181, JAPAN
2 School of Mathematical Sciences, University of Science and Technology of China, No.96, Jinzhai Road, Hefei, Anhui, 230026, China
∗ E-mail address: xfliu@math.sc.niigata-u.ac.jp
† E-mail address: liuyzz@mail.ustc.edu.cn
‡ E-mail address: yangzw@ustc.edu.cn

ical aspects, there have been efficient implementations of the MCF algorithms. One of the popular algorithms is the *network simplex* algorithm, for which one of the estimations for the worst-case runtime is $O(N^2 M \log(NC))$, where N, M are numbers of nodes and edges of a network, and C is the maximum cost of any edges. For MCF problem corresponding to paper review assignment problem with n papers and m reviewers, we have $N = n + m + 2$, $M \leq nm + n + m$, and C is chosen to be a constant of small value. This implies that, the paper review assignment problem requires at most strongly polynomial runtime and it will be more efficient to apply the MCF model than the integer programming one. Such a theoretical prediction is in fact verified by our simulation results. For example, for 800 papers and 640 reviewers, our simulation results show that the running time is reduced from 7.0 seconds by integer programming algorithm to only 0.25 seconds by network simplex algorithm.

The construction of this paper is as follows. In §2, we introduce the problem of paper review assignment and the exiting algorithms. In §3, we show the min-cost flow network model and explain how to formulate the paper review assignment problem as a min-cost flow network model. In §4, the efficiency of our proposed methods is demonstrated by computation results.

2. Problem of paper review assignment

The reviewer assignment problem can be described as follows: Given m reviewers and n papers to review, assign the papers to the reviewers under the conditions that

- For each paper, there should be q reviewers.
- For each reviewer, there should be at most p papers to review.

When solving the review assignment problem, the bidding results, presented by matrix $B_{n,m}$, should be taken into account.

- For paper i, reviewer j has a decision as "Want", "Maybe", "Don't want". Let us take
  - $B_{ij} = 2$ in case of "Want";
  - $B_{ij} = 1$ in case of "Maybe";
  - $B_{ij} = 0$ in case of "Don't want".
- Reviewer cannot review a paper if there exists conflict of interest, for example, the reviewer is also the author of the paper. In this case, take $B_{ij} = -1$.

Denote the final assignment result by 0-1 matrix $S_{n\times m}$. The case $S_{ij} = 1$ means paper i is assigned to reviewer j.

2.1. Existing approaches for review assignment problem

A natural idea to solve the assignment problem is to set up the problem as below.

$$\text{Propblem A:} \quad \max \sum_{i=1}^{n} \sum_{j=1}^{m} S_{ij} c_{ij} \tag{1}$$

subject to

$$S_{ij} \in \{0,1\}, \quad \sum_j S_{ij} = q, \quad \sum_i S_{ij} \leq p\,. \tag{2}$$

Here $c_{n\times m}$ is the weight matrix that evaluates the relation between reviewers and papers. For paper i and reviewer j, we can, for example, take

- $c_{ij} = 2$ in case of "Want",
- $c_{ij} = 1$ in case of "Maybe",
- $c_{ij} = 0$ in case of "Don't want",
- $c_{ij} = -\infty$ in case of interest conflict.

To solve the problem A, a simple try is to apply the dynamic programming algorithm, which is be regarded as a universal algorithm but the computing time will dramatically increase for larger m and n. For a simple model such that each reviewer can review any paper, there are total $\binom{pm}{nq}$ combinations for the papers and the reviewers.

The model (1) can also be classified by integer linear programming problem, which is however a class of NP-hard problem. As we know, the integer linear programming problems are in general very difficult to solve. In §4, a comparison between integer programming algorithm from MATLAB Optimization Toolbox and our proposed method is performed.

In [9], the author reformulate Problem A as follows,

$$\text{Problem A}^*\text{:} \quad \text{Maximize } a^T b, \text{ subject to } Kb \leq d \tag{3}$$

where b is the $0-1$ vector denoting the relations between reviewers and papers, K is a $0-1$ matrix describing the the constraint conditions. In the worst case, there are mn relations and the constraint number is $(m+n)+2mn$, noticing that Problem A* is just a reformulation of (2). By taking advantage of the so called "totally unimodular" property of K, it is proved that there is one optimal solution as binary solution.

Problem A* can be solved by applying the simplex method or the interior point method, which is usually remarkably efficient in practice. For the problem with proper scale of papers and reviewers (for example, $n = 1100$, $m \approx 500$), the problem can be solved in several seconds[9]. However, as noted in [6], for the worst-case, for example, the case of the Klee-Minty cube, the complexity of simplex method is exponential time on the number of variables and constraint number.

In this paper, we consider a simlar model as Problem A as follows.

$$\text{Propblem B:} \qquad \min \sum_{i=1}^{n}\sum_{j=1}^{m} S_{ij}c_{ij} \tag{4}$$

subject to

$$S_{ij} \in \{0,1\}, \quad \sum_j S_{ij} = q, \quad \sum_i S_{ij} \leq p\,. \tag{5}$$

The cost function is decided upon the bidding results. For the bidding result between reviewer

j and paper i, take

- $c_{ij} = 0$ in case of "Want";
- $c_{ij} = CostRef(> 0)$ in case of "Maybe";
- $c_{ij} = CostScale(> CostRef)$ in case of "Don't want";
- $c_{ij} = \infty$ in case of interest conflict.

In next section, instead of using integer programming algorithm, we will solve Problem B by regarding it as a minimum cost flow network problem.

3. Minimum cost flow network model

Let us introduce the standard minimum cost flow problem. Let $G = (V, E)$ be a directed graph with associated integral cost $c_{ij}(\geq 0)$ and capacity $u_{ij}(\in N^+)$ for each $(i, j) \in E$. Associate each node in V an integer number b which indicates the supply (or demand) of the node if $b(i) > 0$ (or $b(i) < 0$). The minimum cost flow problem is stated as follows

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{6}$$

subject to

$$x_{ij} \in N^+ \cup \{0\}, \quad 0 \leq x_{ij} \leq u_{ij}, \quad \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ij} = b(i)\,. \tag{7}$$

A graph showing the minimum cost flow network is given in Figure 1.

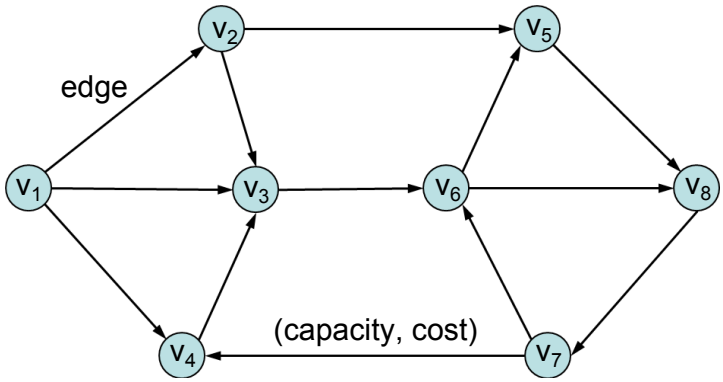

Figure 1 Minimum cost flow network

For the above model problem, there have been many literatures in the history. From theoretical analysis, it is shown that the min-cost flow network problem can be solved in *strongly polynomial runtime*. Here, the *strongly polynomial runtime* means the runtime is a polynomial of $|V|$ and $|E|$ and is independent on the magnitude of cost and capacity. For the implementation of MCF algorithms, one of the estimations shows that the worst runtime of network simplex algorithm is $O(N^2 M \log(NC))$, where $N = |V|$, $M = |E|$ and C is the maximum cost of any edges [7, 8]. For a comprehensive study of the theories, algorithms, and applications of network flows, refer to the book of Ahuja, Magnanti, and Orlin [4].

Remark 3.1. Algorithm Runtime Terminology *Take the number of inputs to our algo-*

rithm to be n integers in the range $0, 1, ..., U$. *Assume that we are working in the Word-RAM model, where standard arithmetic operations can be computed in constant time. The runtime of algorithms is classified into the types as follows.*

1. *An algorithm is strongly polynomial if its runtime is* $poly(n)$.
2. *An algorithm is weakly polynomial if its runtime is* $poly(n, \log U)$.
3. *An algorithm is pseudopolynomial if its runtime is* $poly(n, U)$.

3.1. Review assignment problem v.s. min-cost flow problem

We describe how to formulate Problem B to a min-cost flow problem.

Nodes and edges Define the paper set $V_p := \{p_1, \cdots, p_n\}$ and the reviewer set $V_r := \{r_1, \cdots, r_m\}$. Introduce a source node n_{in} and the target node n_{out}. Let $V = V_r \cup V_p \cup \{n_{in}, n_{out}\}$. The edge set E can be divided into 3 groups:

1) *Source edges*: all edges from the source node n_{in} to each node of V_p.
2) *Target edges*: all edges from each node of V_r to the target node n_{out}.
3) *Paper-reviewer edges*: the edge (p_i, e_j) from $p_i \in V_p$ to $r_j \in V_r$ if $B_{ij} \geq 0$.

In Figure 2, we show the graph for the network of paper review problem.

Remark 3.2. *Notice that if a reviewer* r_j *has interest conflict with a paper* p_i, *then there is no edge between* p_i *and* r_j.

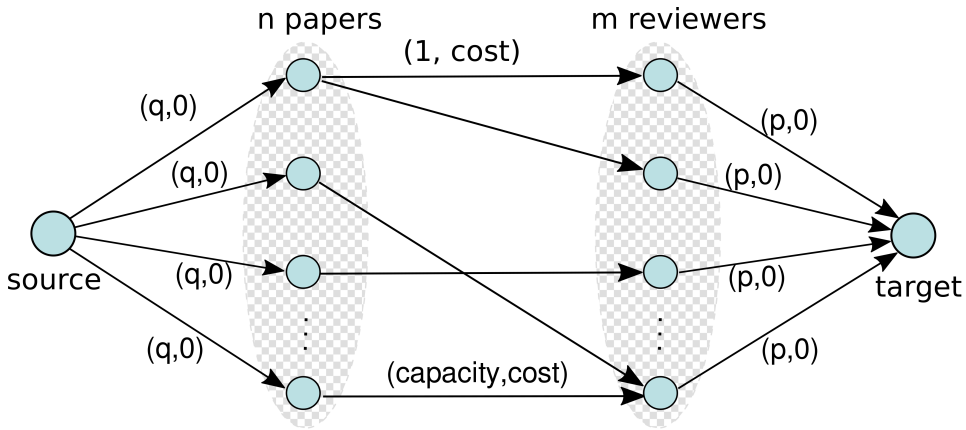

Figure 2 Minimum cost flow network for paper review

Supply/demand and capacity The supply/demand function b is defined by

$$\begin{cases} b(n_{in}) = -nq; \\ b(n_{out}) = nq; \\ b(v) = 0, \quad \forall v \in V_r \cup V_p. \end{cases} \tag{8}$$

The capacity u for edges of E is defined by

1) $u = q$ for each edge from n_{in} to V_p;
2) $u = p$ for each edge from node of V_r to n_{out};

3) $u = 1$ for each edge from node of V_p to node of V_r;

Remark 3.3. *For a solution of the min-cost flow problem, the flow on Paper-reviewer edges will be taken* 0 *or* 1*, the later of which means that one paper is assigned to a reviewer. The above setting of supply/demand and capacity makes sure that for each paper there should be* q *reviewers and for each reviewer, the number of assigned paper will not be larger than* p*.*

Cost of flow Define cost c for each edge of E. For edge (p_i, e_j) of *Paper-reviewer edges* from $p_i (\in V_p)$ to $r_j (\in V_r)$,

$$\begin{cases} c_{ij} := 0 & \text{if } B_{ij} = 2; \\ c_{ij} := CostRef \ (> 0) & \text{if } B_{ij} = 1; \\ c_{ij} := CostScale \ (CostScale > CostRef) & \text{if } B_{ij} = 0. \end{cases} \tag{9}$$

Here, $CostRef$ and $CostScale$ are both positive integers. For other edge of *Source edges* and *Target edges*, take $c = 0$.

Remark 3.4. *The rationality of the above selection of edge set and the associated cost can be understood as follows. If a reviewer "Want" to review a paper, the cost for such a job(= a route in a flow network) is regarded as* 0*, thus such a route will have the highest priority. For the bidding result as "Maybe", a reference cost as* $c = 1$ *is assigned. If a reviewer "Don't want" to review a paper, we can adjust the value of* $CostScale$ *to respect reviewer's will. A larger value of* $CostScale$ *will drop the possibility that a reviewer is assigned a paper that he don't want to review.*

Remark 3.5. *One should pay more efforts to the cost setting for "Don't want" bidding result in practical review assigning system. For a conference with many papers, the default bidding result is often taken as "Don't want" to save the time of reviewers. Thus, it is not too bad to assign a paper to a reviewer who "Don't" want to review. As a response in the flow network model, the value of* $CostScale$ *can be given a relatively small value.*

Another scheme for defining the cost of "Don't want" is to consider the research field matching for paper and reviewers. Define a category list (i.e., research field list) $G = \{1, 2, \cdots, \ell\}$*. The research field information for a reviewer can be presented by a* $0-1$ *matrix* R *with dimension* $m \times \ell$*, the* (j,k)*th element of which is denoted by* R_{jk}*.* $R_{j,k} = 1$ *means that reviewer* r_j *belongs to field* k*. Similarly, we define the* $n \times \ell$ *matrix* P *for the category information of papers. Now, we can define matching score, for example, by*

$$MatchScore_{ij} := \sum_{k=1}^{\ell} P_{ik} R_{jk} / \max\{\sum_{k=1}^{\ell} P_{ik}, \sum_{k=1}^{\ell} R_{jk}\} \quad (0 \le MatchScore_{ij} \le 1)$$

For a high-score matching, one should take smaller value of cost, for example,

$$c_{ij} := CostScale \cdot (2 - MatchScore_{ij}) \quad \text{if } B_{ij} = 0 \, .$$

Flow solution For a solution of minimum cost flow problem, the flow x on an edge $(p_i, r_j) \in E$ will be 0 or 1. Such a result can be saved into the 0-1 review assignment matrix S, i.e., $S_{i,j} = x$.

Noticing the theoretical analysis for the runtime of MCF algorithm, the MCF problem considered here for paper review assignment problem can be solved in $O(nm(n+m)^2 \log(n+m))$ time.

3.2. Other related operation models

The paper review assignment problem is related to the *Assignment Problem* and the *Generalized Assignment Problem*. Here we show the difference between each other.

Assignment Problem (AP) The assignment problem is described as follows. Given two equally sized sets N_1 and N_2 (i.e., $|N_1| = |N_2|$), a collection of pairs $A \subset N_1 \times N_2$ representing possible assignments, and a cost c_{ij} associated with each element $(i, j) \in A$. The corresponding min-cost flow network is $G = (N_1 \cup N_2, A)$ with $b(i) = 1$ for $i \in N_1$, $b(j) = -1$ for $j \in N_2$ and the capacity being 1 for all edges. The application of min-cost flow network to *assignment problem* is also mentioned in Chapter 1 of [4]. However, since each reviewer can have multiple review tasks, the assignment problem along with its algorithm cannot be applied to the paper review assignment problem here.

Generalized Assignment Problem (GAP) The GAP is also a standard problem in operations research; see, e.g., [5]. Given n items and m knapsacks with

$$p_{ij} = \text{profit of item } j \text{ if assigned to knapsack } i\,, \tag{10}$$

$$w_{ij} = \text{weight of item } j \text{ if assigned to knapsack } i\,, \tag{11}$$

$$c_i = \text{capacity of knapsack } i\,, \tag{12}$$

$$\tag{13}$$

assign the items to knapsacks so as to minimize or maximize

$$\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}x_{ij}$$

subject to

$$\sum_{i=1}^{m} w_{ij}x_{ij} \le c_j, \quad \sum_{j=1}^{n} x_{ij} = 1, \quad x_{ij} = 0 \text{ or } 1, \qquad \forall i = 1, \cdots n,\ \forall j = 1, \cdots m\,.$$

It is not difficult to transform the paper review assignment problem into the above GAP by the following operations.

1) take $w_{ij} = 1$;
2) regard the nq reviewing tasks as nq items and the m reviewers as m knapsacks;
3) for each paper, take only one p_{ij} to be postivie finite value for the q review tasks (items) and others to be negative value or even $-\infty$, in case it is a maxmization problem.

The operation 3) is to make sure that a paper is not assigned to the same reviewer more than one time.

The GAP with $w_{ij} = 1$ can also be tranformed to a min-cost flow network. However, GAP itself is an NP-hard problem. Thus it cannot be expected that the algorithm for GAP gives better efficiency than the one for MCF.

From the above analysis we can see that the reviewer assignment problem discussed here is between the AP and GAP and to select a proper model for such a problem is very important.

4. Optimization library and simulation results

The optimization problems have been well explored in the history along with the development of computing technologies. In these days, there are many optimization libraries provided in various programming languages. In our computing experiments, to compare the efficiency of different models, we adopt three libraries as listed in Table 1. Notice that for different libraries, the computing environment is little different from each other.

Table 1 Libraries and computing environment

Library	Language	Environment(OS, Memory, CPU)
Optimization Toolbox $^{\text{TM}}$ of MATLAB	MATLAB	Windows 7, 16GB, 2.80GHz*4
LEMON[2]	C++	Ubuntu 16.04, 16GB, 2.80GHz*4
Google Optimization Tools (or-tools)[1]	Python	Ubuntu 16.04, 16GB, 2.80GHz*4

Below, we give a short introduction to LEMON and or-tools.

1) LEMON stands for *Library for Efficient Modeling and Optimization in Networks*. It is an open C++ template library providing efficient algorithms for combinatorial optimization tasks connected mainly with graphs and networks. It provides easy-to-use and highly efficient implementations of graph algorithms and related data structures, which help solving complex real-life optimization problems.
2) The or-tools is one of Google's software suite for combinatorial optimization. As an open source and free software, the suite contains constraint programming solver, a simple and unified interface to several linear programming and mixed integer programming solvers, Knapsack algorithms and Graph algorithms.

In the comparison of the efficiency of optimization models used in paper review assignment problem, the testing data of different scales, i.e., the bidding result matrices, is created randomly by using the MATLAB code in Appendix A.

To give an estimation of the assignment results from different viewpoints, we pay attention to how the "Want" bidding results are dealt with in the min-cost flow network model. For an assignment result, define two estimation scores by

$$\begin{aligned}\text{ScoreP} &= \sum_{i=1}^{n} (\min(q, \#\{\text{reviewer who wants to review paper } i\}) \\ &\quad -\#\{\text{reviewer assigned to paper } i \text{ who wants to review the paper }\}) \quad (14)\end{aligned}$$

$$\text{ScoreR} = \sum_{j=1}^{m} (\min(p, \#\{\text{paper that reviewer } j \text{ wants to review}\})$$

$$-\#\{\text{paper assigned to reviewer } j \text{ that is wanted by reviewer } j\}) \qquad (15)$$

Here, $\#\{\cdot\}$ means the number of the elements of a given set.

In Table 2, the computing time (unit: second) and the estimation scores ($ScoreP$, $ScoreR$) of three libraries are displayed. For LEMON library, the network simplex algorithm is adopted. The parameters in the simulations are takes as follows.

$$q = 3, \quad p = 5, \quad CostScale = 1, \quad CostScale = 2\,.$$

Table 2 Comparison of models and libraries (unit of time: second)

(n,m)	Integer programming (MATLAB) (time, ScoreP, ScoreR)	Min-cost flow (LEMON) (time, ScoreP, ScoreR)	Min-cost flow (or-tools) (time, ScoreP, ScoreR)
(100, 80)	(0.032, 0, 0)	(0.002, 0, 0)	(0.003, 0, 0)
(200, 160)	(0.101, 1, 20)	(0.009, 2, 21)	(0.010, 2, 21)
(400, 320)	(0.462, 2, 200)	(0.051, 2, 200)	(0.057, 2, 200)
(800, 640)	(6.993, 3, 118)	(0.164, 3, 118)	(0.246, 3, 118)
(1600, 1240)	(43.08, 1, 917)	(0.670, 1, 917)	(1.070, 1, 917)

From the simulation results we can see that the min-cost flow network model gives significant improvement of running time, compared with the integer programming model. For the two min-cost flow network libraries, LEMON and or-tools give almost the same performance, while the minor difference may be up to the programming language selected for each library. Also, notice for each of the scores $ScoreP$ and $ScoreR$, the values are not all the time same to each other for three libraries; for example, see the case $(n, m) = (200, 160)$.

To confirm the effect of parameter $CostScale$, we perform simulation with different values of $CostScale$ and display the results in Table 3. The data is taken from a conference hosted at SmartChair[3]. The parameters for this computation are $n = 123, m = 86$, $CostRef = 10$.

Table 3 Effect of parameter $CostScale$ (or-tools)

CostScale	15	20	25	30
ScoreP	31	34	37	39
ScoreR	5	8	11	13

From the computation result we can see that a larger $CostScale$ gives worse $ScoreR$ and $ScoreP$ (i.e., larger values), which means the model has put more weight on "Don't want" than "Want". In practical applications, by adjudging the value of $CostScale$, we can give a proper respect to "Don't want" upon whether "Don't want" is a default bidding result or not.

5. Summary

The research in optimization theories has provided us powerful tools to solve the practical problems. However, the selection of models will give quite different performance even for the

same problem. In this paper, we novelly formulate the paper review assignment problem as a min-cost flow network problem and the computing time is greatly cut, compared with the existing algorithms based on integer programming model.

For a paper review assignment problem with scale as about $n = 800$, less than 0.3 second is needed. This makes it possible to give real response to the adjustment of input and parameters. That is, one can dynamically set the matching between certain papers and reviewers upon his experience and then check the corresponding result of from automatic assignment.

References

[1] Google Optimzation Tools. `https://developers.google.com/optimization/`. Accessed: 2016-09-01.

[2] LEMON, Library for Efficient Modeling and Optimization in Networks. `http://lemon.cs.elte.hu/trac/lemon`. Accessed: 2016-09-01.

[3] SmartChair conference system. `http://www.smartchair.org`. Accessed: 2016-09-01.

[4] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows: theory, algorithms, and applications. 1993.

[5] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[6] GJ Minty and V Klee. How good is the simplex algorithm. *Inequalities*, 3:159–175, 1972.

[7] James B Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Math. Program.*, 78(2):109–129, 1997.

[8] Robert E Tarjan. Dynamic trees as search trees via euler tours, applied to the network simplex algorithm. *Math. Program.*, 78(2):169–177, 1997.

[9] Camillo J Taylor. On the optimal assignment of conference papers to reviewers. *Technical Report MSCIS-08-30, University of Pennsylvania*, 2008.

A. A MATLAB code to create random bidding results

The code in Table 4 gives a random bidding result matrix. For each reviewer, the bidding result to papers are decided by the following possibilities.

"Want": 0.3%, "Maybe": 1.7%; "Don't want": 97.5%; "Intrest conflict": 0.5% .

Table 4 A MATLAB code to create random bidding results

```
n= 800; m= 640; B=zeros(n,m);
for k = 1:m
  bidding_var = rand(n,1);
  idx_2 = find( bidding_var>=0.997) ;
  idx_1 = intersect( find( bidding_var>=0.98) , find( bidding_var<0.997));
  idx_0 = intersect( find( bidding_var>=0.005) , find( bidding_var<0.98));
  idx_negtive = find( bidding_var<0.005);
  B( idx_2 , k ) = 2;
  B( idx_1 , k ) = 1;
  B( idx_negtive , k ) = -1;
end
save('B.mat','B');
B=int32(B);
save('-ascii', 'B.txt','B');
```

B. Sample code for solving the paper review assignment problem

In Table 5, we provide a Python code for solving the paper review assignment problem. Such a code is using the min-cost flow network algorithm of Google Optimization Tools (https://developers.google.com/optimization/). To show a complete code, a sample bidding matrix B is also given in the code.

Below is the running result from the sample code.

Total 3 papers , 6 reviewers.
Minimum cost : 6
Paper 1 : reviewer [2 3 6]
Paper 2 : reviewer [1 2 5]
Paper 3 : reviewer [3 5 6]

Table 5 The code using Google Optimization Tools for paper review assignment problem

```
from ortools.graph import pywrapgraph
import numpy as np
#A sample bidding result
B=np.array([ [ 0, 1, 2, -1, 1, 0],
             [ 1, 2, 0,  0, 1, 0],
             [ 0, 1, 2,  1, 2, 1]])
(n,m) = B.shape # n: number of papers; m: number of reviewers
print("Total %d papers, %d reviewers."%(n,m))
q = 3; #The number of reviewer needed for each paper
p = 2; #The maximum number of papers for each reviewer

def node_idx_of_paper(i): return 1+i   #Paper nodes: 1~n
def node_idx_of_reviewer(i): return 1+n+i   #Reviewer nodes: (n+1)~(n+1+m)
def node_idx_of_source(): return 0; #Source node: the 0th node
def node_idx_of_target():return 1+m+n; #Target node: the last node

start_nodes = []; end_nodes = []; capacities = []; unit_costs = []
# Source edges
for i in range(0,n): #The edge from source to each paper
  start_nodes.append(node_idx_of_source())
  end_nodes.append(node_idx_of_paper(i))
  capacities.append(q);  unit_costs.append(0)
# Target edges
for j in range(0,m): #The edge from source to each paper
  start_nodes.append(node_idx_of_reviewer(j))
  end_nodes.append(node_idx_of_target())
  capacities.append(p);  unit_costs.append(0)
# Paper-reviewer edges
for i in range(0,n):
  for j in range(0,m):
    if B[i,j] >= 0:
      start_nodes.append(node_idx_of_paper(i))
      end_nodes.append(node_idx_of_reviewer(j))
      capacities.append(1);  unit_costs.append( (2 - B[i,j]) )
# Define an array of supplies at each node
supplies = np.zeros(2+m+n, dtype=int)
supplies[node_idx_of_source()] = n*q; supplies[node_idx_of_target()]= -n*q
# Instantiate a SimpleMinCostFlow solver
min_cost_flow = pywrapgraph.SimpleMinCostFlow()
Solution = np.zeros((n,3), dtype=int) #Assignment result
# Add each arc
for i in range(0, len(start_nodes)):
   min_cost_flow.AddArcWithCapacityAndUnitCost(start_nodes[i], end_nodes[i], capacities
    [i], unit_costs[i])
# Add node supplies
for i in range(0, len(supplies)): min_cost_flow.SetNodeSupply(i, supplies[i])
# Find the minimum cost flow
if min_cost_flow.Solve() == min_cost_flow.OPTIMAL:
  print 'Minimum cost: %d'%min_cost_flow.OptimalCost()
  for i in range(min_cost_flow.NumArcs()):
    if min_cost_flow.Flow(i) > 0:
      p_idx=min_cost_flow.Tail(i)-1; r_idx=min_cost_flow.Head(i)-1-n
      if p_idx >= 0 and r_idx < m: Solution[p_idx,sum(Solution[p_idx,:]>0)]=r_idx+1
  for i in range(0,n): print("Paper %d: reviewer %s"%(i+1, Solution[i]) )
else: print('No solution found')
```