University of Crete
Computer Science Department

Addressing the conference review assignment problem via reduction to integer linear programming.

Ioanna Zikou

Master's Thesis

July 2007

UNIVERSITY OF CRETE
COMPUTER SCIENCE DEPARTMENT

Addressing the conference review assignment problem via reduction to integer linear programming.

Assignment submitted by Ioanna G. Zikou
in partial fulfillment of the requirements for
MASTER OF SCIENCE DEGREE

Author:

Zikou Ioanna, Computer Science Department

Supervisory Committee:

Dr. Plexousakis Dimitris, Professor,
Chairman of Computer Science Department, Supervisor

Dr. Nikolau Christos, Professor, Member

Dr. Georgakopoulos Georgios, Assistant Professor, Member

Approved by:

Dr. Trahanias Panagiotis, Professor,
Chairman of Postgraduate Studies Committee

HERAKLION, JULY 2007

Addressing the conference review assignment problem via reduction to integer linear programming.

Ioanna Zikou
Master's Thesis
Computer Science Department, University of Crete

Abstract

In conferences, many papers are submitted that have to be examined by committee members. It is of high importance that justice rules be applied through the examination, such us fair distribution of the work load to the examiners and equivalent evaluation of the papers.

Up to now, the assignment is done manually and pseudo randomly without ensuring that all committee members will undertake the evaluation of about the same number of papers, neither that they will evaluate those papers closest to their fields of interest.

The purpose of this thesis is to accomplish this assignment through an automated procedure by reducing the problem to an integer linear programming one. This procedure takes data concerning both the members and the papers. Those about members are the topics of the conference that they are interested in, the minimum and the maximum number of papers they may examine and their special interest in some papers (bids). Concerning each paper, the data are the minimum and the maximum number of members that must evaluate it and the topics of the conference it comes under. The result of the procedure is the best possible assignment of the papers to the members, given the current data. Moreover, it detects cases where the problem is not feasible and informs the user about them with appropriate messages. The user

may run again the procedure, after altering suitably the data of the problem, to get a feasible solution.

Supervisor: Dimitris Plexousakis,
Professor

Επίλυση του προβλήματος της ανάθεσης των εργασιών που υποβάλλονται στα συνέδρια στους εξεταστές με αναγωγή σε ακέραιο γραμμικό προγραμματισμό.

Ιωάννα Ζήκου
Μεταπτυχιακή εργασία
Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Περίληψη

Στα συνέδρια υποβάλλονται πολλές εργασίες οι οποίες πρέπει να εξεταστούν από τα μέλη της επιτροπής. Είναι πολύ σημαντική η εφαρμογή κανόνων δικαιοσύνης κατά την εξέταση, όπως ισοκατανομή του φόρτου εργασίας για τα μέλη και ισότιμη αξιολόγηση για τις εργασίες.

Μέχρι τώρα, η ανάθεση των εργασιών στα μέλη γίνεται χειρωνακτικά και ψευδοτυχαία χωρίς να διασφαλίζεται ότι όλα τα μέλη θα αναλάβουν την αξιολόγηση του ίδιου περίπου πλήθους εργασιών, ούτε ότι οι εργασίες που θα αξιολογήσουν είναι αυτές που ταιριάζουν περισσότερο στα ιδιαίτερα ενδιαφέροντά τους.

Στόχος της παρούσας εργασίας είναι να υλοποιήσει την ανάθεση μέσω μιας αυτοματοποιημένης διαδικασίας ανάγοντας το πρόβλημα σε ακέραιο γραμμικό προγραμματισμό. Τα δεδομένα που λαμβάνει η διαδικασία για κάθε μέλος είναι τα θέματα του συνεδρίου που το ενδιαφέρουν, το ελάχιστο και μέγιστο πλήθος εργασιών που πρέπει να αξιολογήσει και τα ιδιαίτερα ενδιαφέροντά του για κάποιες εργασίες, και για κάθε εργασία το ελάχιστο και μέγιστο πλήθος εξεταστών που πρέπει να την αξιολογήσουν καθώς και τις θεματικές περιοχές του συνεδρίου στις οποίες ανήκει. Ως αποτέλεσμα επιστρέφει την βέλτιστη δυνατή ανάθεση των εργασιών στα μέλη για τα εκάστοτε δεδομένα. Επίσης, εντοπίζει περιπτώσεις που το πρόβλημα δεν είναι εφικτό και ενημερώνει τον χρήστη με κατάλληλα μηνύματα. Εν συνεχεία, ο χρήστης μπορεί να

ξαναχρησιμοποιήσει τη διαδικασία έχοντας κατάλληλα τροποποιήσει τα δεδομένα του προβλήματος, ώστε να προκύψει εφικτή λύση.

Επόπτης Καθηγητής: Δημήτρης Πλεξουσάκης,
Καθηγητής

Στη γιαγιά μου, Αδαμαντία...

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή και επόπτη μου κ. Δημήτρη Πλεξουσάκη για την πολύτιμη βοήθειά του καθόλη την διάρκεια των μεταπτυχιακών μου σπουδών και για την άψογη συνεργασία μας καρπός της οποίας αποτελεί και η παρούσα εργασία.

Ιδιαίτερες ευχαριστίες αξίζουν και στον κ. Γεώργιο Γεωργακόπουλο οι παρατηρήσεις και οι συμβουλές του οποίου υπήρξαν πολύτιμες για την περάτωση της παρούσας εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τον κ. Χρήστο Νικολάου για την προθυμία του να συμμετάσχει στην εξεταστική επιτροπή της μεταπτυχιακής μου εργασίας.

Θα ήθελα ακόμη να ευχαριστήσω την κα. Ρένα Καλαϊτζάκη που είναι πάντοτε χαμογελαστή και σε δύσκολες στιγμές μας δίνει κουράγιο και μας μεταδίδει την θετική της ενέργεια.

Τέλος, θα ήθελα να ευχαριστήσω τον Γιάννη, τους γονείς μου, Σωτηρία και Γιώργο, τους φίλους μου Δημήτρη, Γιώργο και Σίσσυ καθώς και τον φίλο Γιάννη Θ. για την υποστήριξή τους τα τελευταία δυόμισι χρόνια.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In conferences, papers are submitted to be examined by committee members. It is of high importance that justice rules be applied through the examination, such us fair distribution of the work load to the examiners and equivalent evaluation of the papers.

Up to now, the assignment is done manually and pseudo randomly without ensuring that all committee members will undertake the valuation of about the same number of papers, neither that they will evaluate those papers closest to their fields of interest.

However, an automated procedure for assigning the papers to the committee members is necessary. The purpose of this thesis is to cover this need ensuring that fair rules will be used. The problem is reduced to an integer linear programming one, which, in general, is NP-Complete. Due to theorems it is proved that it is a special case of integer programming and always has an integral feasible basic solution that can be found in polynomial time.

Information being the data of the problem are: the number of committee members and submitted papers, the topics that a member is interested in, the bids of each member for some papers, the topic(s) to which a paper belongs and the constraints of: a) the minimum and the maximum papers that each member has to examine and b) the minimum and the maximum members that have to examine each paper. If there is a special reason, these constraints may vary from member to member and from paper to paper. The results show that the solution is the best possible assignment of the

papers to the members given the current data. Moreover, special cases that will lead to an infeasible problem are detected and the user is informed by appropriate messages. Such cases are that of a member being interested in only one topic with no submitted papers or of a paper submitted in a topic that is not within the interest of any member, and, finally, the case where the constraints are not well defined and have to be changed. The user may run the procedure again, after altering suitably the data of the problem, to get a feasible solution.

The remainder of this thesis is organized as follows: Chapter 2 elaborates the simplex method as it applies to linear programming problems in both standard and general form. Chapter 3 presents the reduction of our problem to an integer programming one. In chapter 4, implementation and testing of the automated procedure of assigning the papers to the committee members are described. Chapter 5 presents the results of the assigning procedure. Finally, chapter 6 summarizes the contribution of this thesis and identifies issues for further research.

Chapter 2

The Simplex Method

In the previous chapter, it was mentioned that our problem is a special case of integer programming and always has an integral feasible basic solution that can be found in polynomial time. This is accomplished by solving it as a simple linear programming (LP) problem using the Simplex Method.

Some methods solving ILPs are Lenstra's method, Dynamic programming, Gomory's cutting plane method, Lagrangean relaxation method and Gomory's method of corner polyedra [2]. Here, the simplex method will be presented as it applies to linear programming problems in standard and general form [1].

2.1 The Linear Programming Problem

In LP problems, there are variables whose values are to be decided in some optimal fashion. These variables are referred to as *decision variables*. They are usually written as

$$x_j , j = 1, 2, \ldots , n.$$

In LP, the objective is always to maximize or to minimize some linear function of these decision variables

$$\zeta = c_1x_1 + c_2x_2 + \cdots + c_nx_n.$$

This function is called the *objective function*. It often seems that real-world problems are most naturally formulated as minimizations (since real-world planners always seem to be pessimists), but when discussing mathematics it is usually nicer to work with maximization problems. Of course, converting one to the other is trivial both from the

modeller's viewpoint (either minimize cost or maximize profit) and from the analyst's viewpoint (either maximize ζ or minimize - ζ).

In addition to the objective function, there are constraints. Some of these constraints are simple, while others are more involved. But in all cases they consist of either an equality or an inequality associated with some linear combination of the decision variables:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \left\{ \begin{matrix} \leq \\ = \\ \geq \end{matrix} \right\} b.$$

It is easy to convert constraints from one form to another. For example, an inequality constraint

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b$$

can be converted to an equality constraint by adding a nonnegative variable, w, which is called a *slack variable*:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n + w = b,\ w \geq 0.$$

On the other hand, an equality constraint

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

can be converted to inequality form by introducing two inequality constraints:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b$$
$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq b.$$

Hence, in some sense, there is no a priori preference for how one poses the constraints (as long as they are linear, of course). However from a mathematical point of view, there is a preferred presentation. It is to pose the inequalities as less-thans and to stipulate that all the decision variables be nonnegative. Hence, the LP problem can be formulated as follows:

$$
\begin{aligned}
\text{maximize } & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\text{subject to } & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \le b_1 \\
& a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \le b_2 \\
& \vdots \\
& a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n \le b_n \\
& x_1, x_2, \ldots x_n \ge 0.
\end{aligned}
$$

Linear programs formulated in this way are referred as linear programs in *standard form*. A proposal of specific values for the decision variables is called a *solution*. A solution (x_1, x_2, . . . x_n) is called *feasible* if it satisfies all of the constraints. It is called *optimal* if in addition it attains the desired maximum.

Some problems are just simply infeasible, as the following example illustrates:

$$
\begin{aligned}
\text{maximize} \quad & 8x_1 + 3x_2 \\
\text{subject to} \quad & x_1 + 2x_2 \le 4 \\
& -2x_1 - 4x_2 \le -12 \\
& x_1, x_2 \ge 0.
\end{aligned}
$$

Indeed, the second constraint implies that $x_1 + 2x_2 \ge 6$, which contradicts the first constraint. If a problem has no feasible solution, then the problem itself is called *infeasible*.

At the other extreme from infeasible problems, one finds unbounded problems. A problem is *unbounded* if it has feasible solutions with arbitrarily large objective values. For example, consider

$$
\begin{aligned}
\text{maximize} \quad & 3x_1 - 5x_2 \\
\text{subject to} \quad & -3x_1 + 2x_2 \le -4 \\
& -2x_1 - 4x_2 \le -3 \\
& x_1, x_2 \ge 0.
\end{aligned}
$$

Here, x_2 could be set to zero and x_1 be arbitrarily large. As long as x_1 is greater

than 2 the solution will be feasible, and as it becomes large the objective function does too. Hence, the problem is unbounded.

2.2 The Dual problem

Associated with every linear program is another called its dual. The dual of this dual linear program is the original linear program (which is then referred to as the primal linear program). Hence, linear programs come in primal/dual pairs. It turns out that every feasible solution for one of these two linear programs gives a bound on the optimal objective function value for the other. These ideas are important and form a subject called duality theory.

Given an LP problem in standard form,

$$\text{maximize } \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to } \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = 1, 2, \ldots, m \qquad (2.1)$$

$$x_j \ge 0 \quad j = 1, 2, \ldots, n,$$

the associated *dual linear program* is given by

$$\text{minimize } \sum_{i=1}^{m} b_i y_i$$

$$\text{subject to } \sum_{i=1}^{m} y_i a_{ij} \ge c_j \qquad j = 1, 2, \ldots, n$$

$$y_i \ge 0 \qquad i = 1, 2, \ldots, m.$$

(2.1) is called the *primal problem*. It will be shown that taking the dual of the dual returns us to the primal. To see this, the dual problem must be written in standard form. That is, changing the minimization into maximization and the first set of greater-than-or-equal-to constraints into less-than-or-equal-to. Of course, these changes should not alter the problem. To change a minimization into maximization, it is noted that to minimize something it is equivalent to maximize its negative and then negate the answer:

$$\min \sum_{i=1}^{m} b_i y_i = - \max (- \sum_{i=1}^{m} b_i y_i)$$

A multiplication by minus one changes the direction of the inequalities. The resulting equivalent representation of the dual problem in standard form then is

$$\text{- maximize } \sum_{i=1}^{m} (-b_i) y_i$$

$$\text{subject to } \sum_{i=1}^{m} (-a_{ij}) y_i \leq (- c_j) \; j = 1, 2, \ldots, n$$

$$y_i \geq 0 \quad i = 1, 2, \ldots, m.$$

Its dual is:

$$\text{- minimize } \sum_{j=1}^{n} (-c_j) x_j$$

$$\text{subject to } \sum_{j=1}^{n} (-a_{ij}) x_j \geq (- b_i) \; i = 1, 2, \ldots, m$$

$$x_j \geq 0 \quad j = 1, 2, \ldots, n,$$

which is clearly equivalent to the primal problem as formulated in (2.1).

2.2.1 Complementary Slackness

Sometimes it is necessary to recover an optimal dual solution when only an optimal primal solution is known. The following theorem, known as the *Complementary Slackness Theorem*, can help in this regard.

Theorem 2.1. Suppose that x = (x1, x2, . . . , xn) is primal feasible and that y = (y1, y2, . . . , ym) is dual feasible. Let (w1,w2, . . . ,wm) denote the corresponding primal slack variables, and let (z1, z2, . . . , zn) denote the corresponding dual slack variables. Then x and y are optimal for their respective problems if and only if

$$x_j z_j = 0, \quad \text{for } j = 1, 2, \ldots, n,$$
$$w_i y_i = 0, \quad \text{for } i = 1, 2, \ldots, m.$$

2.3 The Simplex Method for problems in standard form

In this section the simplex method is presented as it applies to linear programming problems in standard form.

2.3.1 The Primal Simplex Method

Consider the general linear programming problem presented in standard form:

$$\text{maximize } \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to } \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = 1, 2, \ldots, m$$

$$x_j \ge 0 \quad j = 1, 2, \ldots, n.$$

The first task is to introduce slack variables and a name for the objective function value:

$$\zeta = \sum_{j=1}^{n} c_j x_j$$

$$w_i = b_i - \sum_{j=1}^{n} a_{ij} x_j \quad i = 1, 2, \ldots, m \qquad (2.2)$$

As the simplex method proceeds, the slack variables become intertwined with the original variables, and the whole collection is treated the same. Therefore, it is at times convenient to have a notation in which the slack variables are more or less indistinguishable from the original variables. So they are simply added to the end of the list of x-variables:

$$(x_1, \ldots, x_n, w_1, \ldots, w_m) = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+m}).$$

That is, letting $x_{n+i} = w_i$. With this notation, (2.2) can be rewritten as

$$\zeta = \sum_{j=1}^{n} c_j x_j$$

$$x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j \quad i = 1, 2, \ldots, m.$$

Systems of equations like this are called dictionaries. This is the starting dictionary. With the exception of ζ, the variables that appear on the left (the dependent variables) are called *basic variables*. Those on the right (the independent variables) are called *nonbasic variables*. Any solution obtained by setting the nonbasic variables to zero is called *basic feasible solution*.

The simplex is an iterative process and begins with an initial feasible solution that satisfies equations and nonnegativities in (2.2). In each step a new solution is tried to be reached, which is better in the sense that it has a larger objective function value.

Each dictionary has m basic variables and n nonbasic variables. Let B denote the collection of indices from {1, 2, ..., n + m} corresponding to the basic variables, and let N denote the indices corresponding to the nonbasic variables. Initially, there are N = {1, 2, ..., n} and B = {n + 1, n + 2, ... , n + m}, but this of course changes after the first iteration. Down the road, the current dictionary will look like this:

$$\zeta = \tilde{\zeta} + \sum_{j \in N} \tilde{c}_j x_j$$

$$x_i = \tilde{b}_i - \sum_{j \in N} \tilde{a}_{ij} x_j \quad i \in B \qquad (2.3)$$

Note that there are bars over the coefficients to indicate that they change as the algorithm progresses. Within each iteration of the simplex method, exactly one variable goes from nonbasic to basic and exactly one variable goes from basic to nonbasic.

The variable that goes from nonbasic to basic is called the *entering variable*. It is chosen with the aim of increasing ζ; that is, one whose coefficient is positive: *pick* k *from* $\{j \in N : \tilde{c}_j > 0\}$. Note that if this set is empty, then the current solution is optimal. If the set consists of more than one element (as is normally the case), then there is a choice of which element to pick. There are several possible selection criteria. Usually an index k having the largest coefficient is picked (which again could leave a choice). The variable that goes from basic to nonbasic is called the *leaving variable*. It is chosen to preserve nonnegativity of the current basic variables. Once it has been decided that x_k will be the entering variable, its value will be increased from zero to a positive value. This increase will change the values of the basic variables:

$$x_i = \tilde{b}_i - \tilde{a}_{ik}x_k, \qquad i \in B.$$

It is necessary to ensure that each of these variables remains nonnegative. Hence, it is required that

$$\tilde{b}_i - \tilde{a}_{ik}x_k \geq 0, \qquad i \in B. \qquad (2.4)$$

Of these expressions, the only ones that can go negative as x_k increases are those for which $\tilde{a}_{ik}$ is positive; the rest remain fixed or increase. Hence, the attention can be restricted to those i's for which $\tilde{a}_{ik}$ is positive. And for such an i, the value of x_k at which the expression becomes zero is

$$x_k = \tilde{b}_i / \tilde{a}_{ik}.$$

Since none of these should become negative, x_k must be raised only to the smallest of all of these values:

$$x_k = \min_{i \in B : \tilde{a}_{ik} > 0} (\tilde{b}_i / \tilde{a}_{ik}).$$

Therefore, with a certain number of latitude remaining, the rule for selecting the leaving variable is *pick* i *from* {$i \in B : \tilde{a}_{ik} > 0$ *and* $\tilde{b}_i / \tilde{a}_{ik}$ *is minimal*}.

The rule just given for selecting a leaving variable describes exactly the process used in practice. That is, looking only at those variables for which $\tilde{a}_{ik}$ is positive and among those selecting one with the smallest value of the ratio $\tilde{b}_i / \tilde{a}_{ik}$. There is, however, another, entirely equivalent, way to write this rule. To derive this alternate expression the convention that 0/0 = 0 is used, so rewriting inequalities (2.4) gives

$$1/x_k \geq \tilde{b}_i / \tilde{a}_{ik}, \; i \in B$$

Since it is wished to take the largest possible increase in x_k, it is:

$$x_k = (\max_{i \in B} \max \tilde{a}_{ik} / \tilde{b}_i)^{-1}, \; i \in B.$$

Hence, the rule for selecting the leaving variable is as follows: *pick* i *from* {$i \in B$: $\tilde{a}_{ik} / \tilde{b}_i$ *is maximal*}. The main difference between these two ways of writing the rule is

that in one the ratio of $\tilde{a}_{ik}/\tilde{b}_i$ is minimized whereas in the other the reciprocal ratio is maximized. Of course, in the minimize formulation one must take care with the sign of the $\tilde{a}_{ik}$'s.

When these types of ratios have to be encountered, they will be written in the maximize form since that is shorter to write, acknowledging that it is often more convenient, in practice, to do it the other way.
Once the leaving-basic and entering-nonbasic variables have been selected, the move from the current dictionary to the new dictionary involves appropriate row operations to achieve the interchange. The entering variable should not appear on the right of the new basis equations. This step from one dictionary to the next is called a *pivot*. As mentioned above, there is often more than one choice for the entering and the leaving variables. Particular rules that make the choice unambiguous are called *pivot rules*.

2.3.2 Initialization

In the previous section, the simplex method was presented. However, only problems with nonnegative right-hand sides were considered. This ensured that the initial dictionary was feasible. In this section, it is examined what one needs to do when this is not the case.

Given a linear programming problem

$$\text{maximize } \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to } \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = 1, 2, \ldots, m$$

$$x_j \ge 0 \quad j = 1, 2, \ldots, n,$$

a dictionary is introduced

$$\zeta = \sum_{j=1}^{n} c_j x_j$$

$$w_i = b_i - \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = 1, 2, \ldots, m$$

The solution associated with this dictionary is obtained by setting each x_j to zero and setting each w_i equal to the corresponding b_i. This solution is feasible if and only if all

the right-hand sides are nonnegative. In case, they are not, an *auxiliary problem* is introduced for which:
(1) a feasible dictionary is easy to find and
(2) the optimal dictionary provides a feasible dictionary for the original problem.

The auxiliary problem is

$$\text{maximize } -x_0$$

$$\text{subject to } \sum_{j=1}^{n} a_{ij} x_j - x_0 \le b_i \quad i = 1, 2, \ldots, m$$

$$x_j \ge 0 \quad j = 0, 1, 2, \ldots, n.$$

It is easy to give a feasible solution to this auxiliary problem by setting $x_j = 0$, for j = 1, …, n, and then picking x_0 sufficiently large. It is also easy to see that the original problem has a feasible solution if and only if the auxiliary problem has a feasible solution with $x_0 = 0$. In other words, the original problem has a feasible solution if and only if the optimal solution to the auxiliary problem has objective value zero. Even though the auxiliary problem clearly has feasible solutions, it has not yet be shown that it has an easily obtained feasible dictionary.

To obtain a feasible dictionary, after formulating the auxiliary problem, slack variables are introduced and an initial *infeasible dictionary* is written down. The initial dictionary is infeasible, but it is easy to convert it into a feasible dictionary. In fact, all that is needed to be done is one pivot with variable x_0 entering and the "most infeasible variable," leaving the basis. When a feasible dictionary is reached, the simplex method can be applied until an optimal for the auxiliary problem dictionary is found. x_0 is now dropped from the equations and the original objective function is reintroduced using the basis equations. Normally it is expected that the dictionary so obtained will be feasible for the original problem, at which point the simplex method keeps being applied until an optimal solution is reached.

The process of solving the auxiliary problem to find an initial feasible solution is often referred to as *Phase I*, whereas the process of going from a feasible solution to an optimal solution is called *Phase II*.

2.3.3 The Dual Simplex Method

In this section, it is examined what happens when the simplex method is applied to the dual problem. Someone can actually apply the simplex method to the dual problem without ever writing down the dual problem or its dictionaries. Instead, the so-called dual simplex method is seen simply as a new way of picking the entering and leaving variables in a sequence of primal dictionaries.

Indeed, it is first noted that the dictionary must be dual feasible. This means that all the coefficients of the nonbasic variables in the primal objective function must be nonpositive. Given this, first the leaving variable is selected by picking that basic variable whose constant term in the dictionary is the most negative (if there is none, then the current dictionary is optimal). Then the entering variable is picked by scanning across this row of the dictionary and comparing ratios of the coefficients in this row to the corresponding coefficients in the objective row, looking for the largest negated ratio just as in the primal simplex method. Once the entering and leaving variable are identified, a pivot to the next dictionary takes place and the procedure continues.

2.3.4 A Dual-Based Phase I Algorithm

The dual simplex method described in the previous section provides a new Phase I algorithm, which is more elegant than the one given in section 2.3.2.

Let us suppose that there is a problem for which neither the primal nor the dual dictionary is feasible. The primal objective function can be changed so as to produce a dual feasible dictionary. The dual simplex method is then applied to the modified problem until an optimal solution is reached. But, while the primal dictionary is optimal for the modified problem, it is not for the original one. It is, however, feasible for the latter. The intended objective function can be reinstated using the basis equations of the optimal primal dictionary of the modified problem and a phase II can be applied in the new starting dictionary. After some iterations of phase II, the problem will be unbounded and an optimal solution will have been reached.

It is interesting to note how infeasibility is detected with this new Phase I algorithm. The modified problem is guaranteed always to be dual feasible. It is easy to see that

the primal problem is infeasible if and only if the modified problem is dual unbounded (which the dual simplex method will detect just as the primal simplex method detects primal unboundedness).

This two-phase algorithm can be thought of as a dual– primal algorithm, since first the dual simplex method is applied to a modified dual feasible problem and then the primal simplex method is applied to the original problem, starting from the feasible dictionary produced by Phase I. One could consider turning this around and doing a primal–dual two-phase algorithm. Here, the right-hand side of the primal problem would be modified to produce an obvious primal feasible solution. The primal simplex method would then be applied. The optimal solution to this primal problem will then be feasible for the original dual problem but will not be optimal for it. But then the dual simplex method can be applied, starting with this dual feasible basis until an optimal solution for the dual problem is obtained.

2.4 The Simple Method for problems in general form

Up until now, all problems were given in standard form. However, for real-world problems as the one of this thesis, it is often convenient to formulate problems in the following form:

$$\begin{aligned} &\text{maximize } c^T x \\ &\text{subject to } a \le Ax \le b \qquad (2.5) \\ &\qquad\qquad l \le \ x \le u. \end{aligned}$$

Two-sided constraints such as those given here are called constraints with *ranges*. The vector l is called the vector of *lower bounds*, and u is the vector of *upper bounds*. Some of the data are allowed to take infinite values; that is, for each i = 1, 2, . . . ,m,

$$-\infty \le a_i \le b_i \le \infty,$$

and, for each j = 1, 2, . . . , n,

$$-\infty \le l_j \le u_j \le \infty.$$

In this section, it will be shown how to modify the simplex method to handle problems presented in this form.

2.4.1 The Primal Simplex Method

The first task is to introduce slack variables and a name for the objective function value:

$$\zeta = c^T x$$
$$w = Ax$$

With this formulation, instead of defining slack variables for each constraint, w_i is simply used to denote the value of the i-th constraint:

$$w_i = \sum_{j=1}^{n} a_{ij} x_j \text{ , } i = 1, 2, \ldots, m.$$

The constraints can be interpreted as upper and lower bounds on these variables. Now when recording the problem in a dictionary, explicit track of the upper and lower bound on the original x_j variables and the new w_i variables must be kept. Also, the value of a nonbasic variable is no longer implicit; it could be at either its upper or its lower bound. Hence, as an indication of the case a box around the relevant bound will be placed. Finally, track of the values of the basic variables needs to be kept.

Hence, for the example

$$\begin{aligned} \text{maximize} \quad & 3x_1 - 2x_2 \\ \text{subject to} \quad & 1 \le -x_1 + 2x_2 \le 6 \\ & 2 \le -3x_1 + 4x_2 \le 12 \\ & 3x_1 - x_2 \le 0 \\ & -3 \le x_1 \\ & 0 \le x_2 \le 8. \end{aligned}$$

the slack variables will be:

$$\begin{aligned} w_1 &= -x_1 + 2x_2 \\ w_2 &= -3x_1 + 4x_2 \\ w_3 &= 3x_1 - x_2. \end{aligned}$$

and the dictionary should be written as follows:

l								
				- 3		0		
	u			∞		8		
		ζ	=	$3x_1$	-	$2x_2$	=	- 9
1	6	w_1	=	$- x_1$	+	$2x_2$	=	3
2	12	w_2	=	$-3x_1$	+	$4x_2$	=	9
-∞	0	w_3	=	$3x_1$	-	x_2	=	- 9

If all the w_i's are between their upper and lower bounds, the dictionary is feasible. The variable that can be increased from its present value at the lower bound increasing the objective function's value shall be the entering variable in each iteration. For the above example, for the first iteration, the entering variable will be x_1. For each basic variable it is checked how much the entering variable can be increased before the basis one hits either its lower or its upper bound. As leaving variable is chosen the one enforcing the tightest increase. In the above example it will be w_2.

The iterations continue until a dictionary is reached where all basic variables are at their upper bounds and have positive coefficients in the formula for ζ. Hence, neither can be moved off from its bound to increase the objective function. Therefore, the current solution will be optimal.

2.4.2 The Dual Simplex Method

The problem considered in the previous section had an initial dictionary that was feasible. But as always, the case where the initial dictionary is not feasible must be addressed. That is, a Phase I algorithm must be defined. Following the ideas presented in section 2.3.4, the Phase I algorithm is based on a dual simplex method. To this end, the dual of (2.5) must be introduced. So first (2.5) is rewritten as

$$
\begin{aligned}
\text{maximize} \quad & c^T x \\
\text{subject to} \quad & Ax \le b \\
& -Ax \le -a \\
& x \le u \\
& -x \le -l,
\end{aligned}
$$

and adding slack variables, gives

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to} \quad & Ax + f = b \\ & -Ax + p = -a \\ & x + t = u \\ & -x + g = -l \\ & f, p, t, g \geq 0. \end{aligned}$$

It is immediately seen from the inequality form of the primal that the dual can be written as

$$\begin{aligned} \text{minimize} \quad & b^T v - a^T q + u^T s - l^T h \\ \text{subject to} \quad & A^T (v - q) - (h - s) = c \\ & v, q, s, h \geq 0. \end{aligned} \tag{2.6}$$

Furthermore, at optimality, the dual variables are complementary to the corresponding primal slack variables:

$$\begin{aligned} & f_i v_i = 0 \qquad i = 1, 2, \ldots, m, \\ & p_i q_i = 0 \qquad i = 1, 2, \ldots, m, \\ & t_j s_j = 0 \qquad j = 1, 2, \ldots, n, \\ & g_j h_j = 0 \qquad j = 1, 2, \ldots, n. \end{aligned} \tag{2.7}$$

For each i, if $b_i > a_i$, then at optimality v_i and q_i must be complementary to each other. Indeed, if both were positive, then they could be reduced by an equal number without destroying feasibility, and the objective function value would strictly decrease, thereby implying that the supposedly optimal solution is not optimal. Similarly, if for some i, $b_i = a_i$, then it is no longer required that v_i and q_i be complementary at optimality; but, given an optimal solution for which both v_i and q_i are positive, both these values can be decreased at the same rate until the smaller of the two reaches zero, all the while preserving feasibility of the solution and not changing the objective function value. Hence, there always exists an optimal solution in which every component of v is complementary to the corresponding component of q. The same argument shows that if there exists an optimal solution, then there exists one in which all the components of h and s are complementary to each other as well.

For a real variable ξ, its positive part ξ^+ is defined as

$$\xi^+ = \max\{\xi, 0\}$$

and its negative part ξ^- is defined similarly as

$$\xi^- = \max\{-\xi, 0\}.$$

Clearly, both ξ^+ and ξ^- are nonnegative. Furthermore, they are complementary,

$$\xi^+ = 0 \text{ or } \xi^- = 0,$$

and their difference represents ξ:

$$\xi = \xi^+ - \xi^-.$$

From the complementarity of the components of v against the components of q, they can be thought as the positive and negative parts of the components of just one vector y. So:

$$v = y^+ \text{ and } q = y^-.$$

Similarly, it can be written

$$h = z^+ \text{ and } s = z^-.$$

By imposing these complementarity conditions not just at optimality but also from the start, v, q, s, and h can be eliminated from the dual that can be written simply as

$$\begin{array}{ll} \text{minimize} & b^T y^+ - a^T y^- + u^T z^+ - l^T z^- \\ \text{subject to} & A^T y - z = c \end{array} \qquad (2.8)$$

where the notation y^+ denotes the componentwise positive part of y, etc. This problem is an example from the class of problems called piecewise linear programs. Usually, piecewise linear programs are solved by converting them into linear programs. Here, however, the other direction is desirable. An algorithm for (2.8) will be presented that will serve as an algorithm for (2.6). This algorithm will be called the *dual simplex method* for problems in general form.

For simplicity, the dual simplex method will be presented in the context of a Phase I algorithm for linear programs in general form. Also, to avoid cumbersome notation, the algorithm will be described with the following example:

$$
\begin{aligned}
&\text{maximize} && 2x_1 - 3x_2 \\
&\text{subject to} && 0 \le x_1 + 2x_2 \le 8 \\
& && 2 \le -x_1 + 6x_2 \le 14 \\
& && x_1 - 3x_2 \le 0 \qquad (2.9) \\
& && -3 \le x_1 \\
& && 1 \le x_2 \le 8
\end{aligned}
$$

The piecewise linear formulation of the dual is

$$
\begin{aligned}
&\text{minimize} && 8y_1^+ + 14y_2^+ + 3z_1^+ - z_2^+ \\
& && \quad - 2y_2^- + \infty y_3^- + \infty z_1^- + 8z_2^- \\
&\text{subject to} && y_1 - y_2 + y_3 - z_1 = 2 \\
& && 2y_1 + 6y_2 - 3y_3 - z_2 = -3
\end{aligned}
$$

The objective function has coefficients that are infinite. The correct convention is that infinity times a variable is plus infinity if the variable is positive, zero if the variable is zero and minus infinity if the variable is negative. Since the objective function is nonlinear (taking positive and negative parts of variables is certainly a nonlinear operation), the usual row operations can not take place on the objective function. Therefore, each iteration is simply being studied as it is. But thinking in terms of maximization the negative of the objective function is recorded:

$$
\begin{aligned}
-\xi = &- 8y_1^+ - 14y_2^+ - 3z_1^+ + z_2^+ \\
&+ 2y_2^- - \infty y_3^- - \infty z_1^- - 8z_2^-.
\end{aligned} \qquad (2.10)
$$

To perform row operations on the two constraints, the usual sort of dictionary is set for them:

$$
\begin{aligned}
z_1 &= -2 + y_1 - y_2 + y_3 \\
z_2 &= 3 + 2y_1 + 6y_2 - 3y_3
\end{aligned} \qquad (2.11)
$$

For the dual problem, all the action takes place at zero. That is, slopes in the objective function change when a variable goes from negative to positive. Since nonbasic variable are supposed to be set where the action is, a current solution is associated with each dictionary by setting the nonbasic variables to zero. Hence, the solution associated with the initial dictionary is

$$(y_1, y_2, y_3, z_1, z_2) = (0, 0, 0, -2, 3).$$

The fact that z_1 is negative implies that z_1^- is a positive number and hence that the objective function value associated with this solution is minus infinity. Whenever the objective function value is minus infinity, the solution and the associated dictionary are said to be *infeasible*. Hence, the initial dictionary given in (2.11) is infeasible.

The dual simplex method must start with a dual feasible solution. But since the dual simplex method will be used simply to find a feasible solution for (2.9), the objective function can freely be changed in (2.9) in any convenient way. In particular, it can be changed from

$$\zeta = 2x_1 - 3x_2$$

to

$$\eta = -2x_1 - 3x_2.$$

Making that change to the primal leaves the dual objective function unchanged, but produces a feasible dual dictionary:

$$\begin{aligned} z_1 &= 2 + y_1 - y_2 + y_3 \\ z_2 &= 3 + 2y_1 + 6y_2 - 3y_3 \end{aligned} \qquad (2.12)$$

For comparison purposes, the corresponding primal dictionary will be recorded. It is easy to write down the equations defining the w_i's, but how is it known whether the x_j 's are supposed to be at their upper or their lower bounds? The answer comes from the requirement that the primal and dual satisfy the complementarity conditions given in (2.7). Indeed, from the dual dictionary, it is seen that $z_1 = 2$. Hence, $z_1^+ = 2$. But since z_1^+ is just a surrogate for h_1, h_1 is positive and hence g_1 must be zero. This means that x_1 must be at its lower bound. Similarly, for the sake of complementarity, x_2 must also be at its lower bound. Hence, the primal dictionary is

l								
				-3		1		
	u			∞		8		
		η	=	- $2x_1$		- $3x_2$	=	3
0	8	w_1	=	x_1		+ $2x_2$	=	- 1
2	14	w_2	=	- x_1		+ $6x_2$	=	9
- ∞	0	w_3	=	x_1		- $3x_2$	=	- 6

Note that it is infeasible, since w_1 is not between its upper and lower bounds. The first iteration of the dual simplex method will now be described. To this end, it is asked whether the dual objective function value can be improved by moving one of the nonbasic variables (y_1, y_2, or y_3) away from zero. Of course, each of these three variables can be moved either to the positive or the negative side of zero; These six cases must be analyzed individually. First of all, since z_1 is positive at the current solution, it follows that $z_1^+ = z_1$ and $z_1^- = 0$ in a neighbourhood of the current solution. A similar statement can be made for z_2, and so (2.10) can be rewritten locally around the current solution as

$$-\xi = -8y_1^+ - 14y_2^+ - 3z_1 + z_2 + 2y_2^- - \infty y_3^-.$$

Now, as y_1 is increased from zero, the rate of increase of $-\xi$ is simply the derivative of the right-hand side with respect to y_1, where it must be kept in mind that z_1 and z_2 are functions of y_1 via the dictionary (2.12). Hence, the rate of increase is $-8-3+2 = -9$; i.e., the objective function decreases at a rate of 7 units per unit increase of y_1. If, on the other hand, y_1 is decreased from zero into negative territory, then the rate of increase of $-\xi$ is the negative of the derivative of the right-hand side. In this case y_1^- does not contributes, but z_1 and z_2 do for a total of $3 - 2 = 1$. Hence, the rate of increase while moving in this direction is one unit increase per unit move. Changes to y_2 and y_3 can be analyzed. The entire situation can be summarized as follows:

y_1	↗	- 8	- 3	+ 2	=	- 9
y_1	↘	0	+ 3	- 2	=	1
y_2	↗	- 14	+ 3	+ 6	=	- 5
y_2	↘	2	- 3	- 6	=	- 7
y_3	↗	0	- 3	- 3	=	- 6
y_3	↘	- ∞	+ 3	+ 3	=	- ∞

Of these six cases, the only one that brings about an increase in $-\xi$ is the one in which y_1 is sent negative. Hence, y_1 shall be the entering variable, and it will go negative. To find the leaving variable, there is the question: as y_1 goes negative, which of z_1 and z_2 will hit zero first? For the current dictionary, z_2 gets to zero first and so becomes the leaving variable. Performing the usual row operations, the new dictionary for the dual problem is

$$z_1 = 0{,}5 + 0{,}5z_2 - 4y_2 + 2{,}5y_3$$
$$y_1 = -1{,}5 + 0{,}5z_2 - 3y_2 + 1{,}5y_3.$$

Looking at the new primal dictionary, the fact that y_1 was the entering variable in the dual dictionary implies that w_1 is the leaving variable in the primal. Furthermore, the fact that y_1 has gone negative implies that y_1^- or equally q_1 is now positive, and so complementarity then demands that p_1 be zero; i.e., w_1 should go to its lower bound.

The fact that z_2 was the leaving variable in the dual dictionary implies that x_2 is the entering variable in the primal. Hence, the new primal dictionary is

l	u			-3	0		
				∞	8		
		η	=	- 0,5x_1	- 1,5w_1	=	1,5
1	8	x_2	=	- 0,5x_1	+ 0,5w_1	=	1,5
2	14	w_2	=	- 4x_1	+ 3w_1	=	12
- ∞	0	w_3	=	2,5x_1	- 1,5w_1	=	- 7,5

The second iteration will now begin. Therefore, it is asked which nonbasic variable should be moved away from zero (and in which direction). As before, it is noted that z_1 positive implies that $z_1^+ = z_1$ and $z_1^- = 0$ and that y_1 negative implies that $y_1^+ = 0$ and $y_1^- = -y_1$. Hence, the objective function can be written locally around the current solution as

$$-\xi = -14y_2^+ - 3z_1 + z_2^+ + 2y_2^- - \infty y_3^- - 8z_2^-.$$

Summarizing the possibilities in a small table:

z_2	↗	1	- 1,5 =	- 0,5
z_2	↘	- 8	+ 1,5 =	- 6,5
y_2	↗	- 14	+ 12 =	- 2
y_2	↘	2	- 12 =	- 10
y_3	↗	0	- 7,5 =	- 7,5
y_3	↘	- ∞	+ 7,5 =	- ∞

All the changes are negative, meaning that there are no possibilities to increase the objective function any further. That is, the current dual solution is optimal. Of course, this also could have been deduced by observing that the primal dictionary is feasible (which is the objective, after all). Even though this example of the dual simplex method has terminated after only one iteration, it should be clear how to proceed had it not terminated.

Now that a feasible solution for the primal was found, the problem to optimality can be solved by simply reinstating the original objective function and proceeding by applying the primal simplex method in a Phase II procedure to find the optimal solution. Since the primal simplex method has already been discussed, this problem stops here.

2.5 Linear programming tools

There are many linear programming tools, such as GIPALS, CPLEX, RIOT, GLPK, LPAKO, SixPap, Soplex, etc. Some of them implement the simplex method, others interior-point methods, but they are not all bug free.

In this thesis, Soplex [3] is used to run tests. Soplex is an implementation of the revised simplex algorithm. It features primal and dual solving routines for linear programs and is implemented as a C++ class library that can be used with other programs. It has been implemented as a part of Roland Wunderling's Ph.D. thesis Paralleler und Objektorientierter Simplex-Algorithmus (in German). It has been tested with compilers from GNU, Compaq, Intel, SUN, HP, SGI, IBM, and M$. It is distributed under the ZIB Academic License (see Appendix). People are allowed to retrieve SoPlex only for research purpose as members of a non-commercial and academic institution.

Chapter 3

Reduction to Integer Programming Problem

3.1 Formulation of the problem

In conferences, papers are submitted to be examined by committee members. It is of high importance that justice rules be applied through the examination, such us fair distribution of the work load to the examiners and equivalent evaluation of the papers. For each paper there must be a lower and an upper bound of examiners and for each committee member there must be a lower and an upper bound of papers to be examined. According to the topic of each paper, its authors and other factors, a matching degree is attributed with each committee member. The members chosen for each paper must be those having the greater matching degree with it.

As known, the committee members first declare the topics of the conference they are interested in. Then papers are submitted and distributed according to their subject to the appropriate topic(s). Finally, members declare their special interests, if any, in some papers, called bids. Each committee member has a profile consisting of information such as his name and the areas of his interest and each paper has a title, author(s) and areas to which it belongs. Both members and papers are given an identification number as reference to them.

Representing graphically the problem, we take a bipartite graph. On one side there are the committee members, denoted by m, while on the other there are the papers, denoted by n. Each member is connected to [min_ppm,max_ppm] papers that,

according to some criteria, he can / must examine and each paper is connected to [min_mpp,max_mpp] examiners. Assuming a full graph, each edge has a weight that is the matching degree between the committee member and the paper. When there is no matching between them, the weight of the corresponding edge is zero.

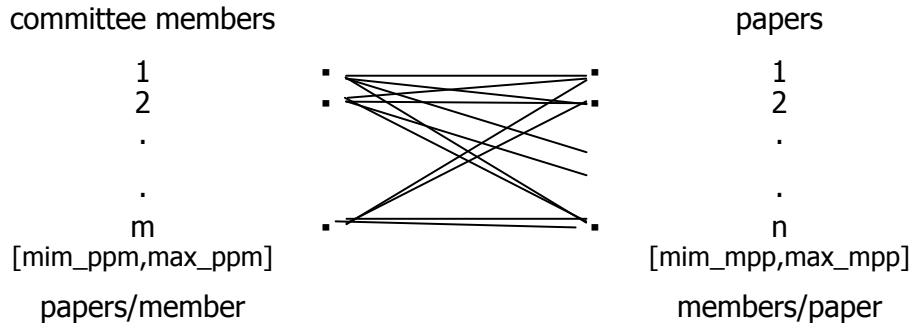

Figure 1: Graphical representation of the problem

It might be easier in understanding the problem to consider the above graph having intermediate nodes being the topics of the conference, denoted by s. Depending on whether a paper belongs to a topic of a member's interest, there is an edge between them (weighted by 1) or not (edge weighted by 0).

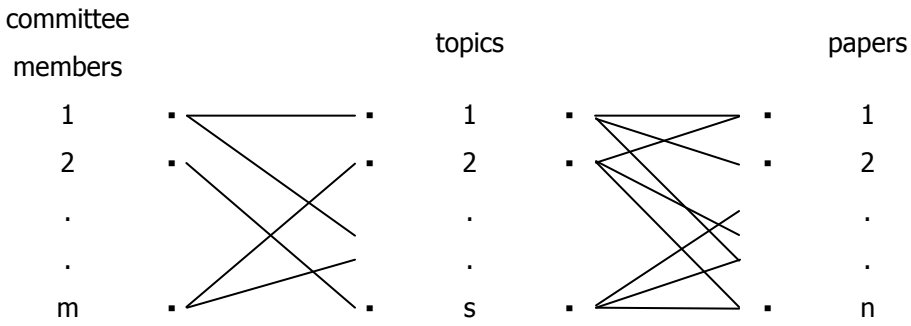

Figure 2: An edge between a member and a topic exists, if they are connected through a topic.

If there is an edge, we check for some special interest of the member for that paper by checking his bids. In case there is a high interest, the edge weight is increased by 3. If there is a medium interest by 2, if low interest by 1 and if there is no interest by 0 (no increase). Finally, if there is a conflict, like in cases where someone can not examine his own paper, the weight is decreased by 1 and thus no edge exists anymore or we have a zero weighted one. In solving the problem we will use the first graph, which is the one generated by the data processing.

Information considered as data of the problem are: the number of committee members, the number of submitted papers, the topics that a member is interested in, the bids of each member for some papers, the topic(s) to which a paper belongs and the constraints of: a) the minimum and the maximum papers that each member has to examine and b) the minimum and the maximum members that have to examine each paper. If there is a special reason, these constraints may vary from member to member and from paper to paper. The solution is the set of edges with the greatest possible weights that satisfies the constraints.

We consider that no member is allowed to show interest for a paper not belonging to a topic of his interests. Also all members should declare at least one topic and all papers should belong to at least one topic of the conference. Finally, cases where a member or a paper has only zero weighted edges in the final graph are detected and messages informing for infeasibility are returned to the user. Infeasibility message is also returned in case of inappropriate constraints, where (m * max_ppm < n * min_mpp). The user may change the data and rerun the procedure to obtain a solution.

3.2 Reduction to a known problem

It is actually an integer linear programming (ILP) problem.

In the general form of these problems, it is given:

1. An array A of rational numbers with M columns and N rows.
2. Two one column vectors a, b with N constants (rational numbers) each.
3. A row vector c with M values (weights – rational numbers).
4. A column vector x with M elements has value cx= $\sum_{j=1}^{M} c_j x_j$

and it is asked:

1. Whether there is x integer such as $a \leq Ax \leq b$.
2. If there is, which x has the greatest value?

that is,

maximize $c^T x$
subject to $a \leq Ax \leq b$
x integer

3.2.1 The idea of reduction [4]

Suppose there are two problems Q_1 and Q_2. d_1 and d_2 are their data, r_1 and r_2 their solution, respectively. The reduction idea is shown in the following figure.

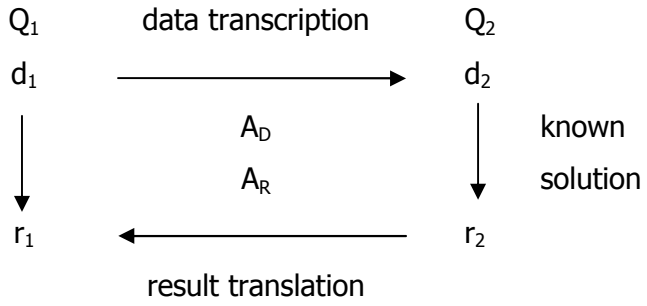

Figure 3: The idea of reduction

The data d_1 of Q_1 are transcribed in d_2 of Q_2 by an algorithm A_D. But for Q_2 a solution r_2 is known. An algorithm A_R translates r_2 to r_1. Thus, having a solution for Q_2 provides one for Q_1.

3.2.2 Reduction to ILP

An instance d_P of our problem P consists of a bipartite graph G=(V,E) and the weight of each edge.

An instance d_{ILP} consists of an array A of MxN dimension, two vectors a, b of Nx1 dimension, a vector c of 1xM dimension and a vector x of Mx1 dimension and $cx = \sum_{j=1}^{M} c_j x_j$ value.

To reduce P to ILP we have to show that for each d_P we can construct quickly and easily an instance d_{ILP} such as the answer to the first problem to be "yes" if and only if the answer to the second one is "yes".

First, using the graph's information, we construct an array A with E columns, one for each edge e=(u,v) of the graph, and V rows, one for each node. In every column e=(u,v) all the coefficients are 0 except for those in rows u, v (that the edge joins) that are 1.

As weight vector c we construct a row vector with E elements (as the edges of the graph) that are the matching degree between each committee member and each paper.

As vectors a, b we construct two column vectors with V elements (as the nodes of the graph) that constitute the minimum and the maximum number of papers per committee member and of members per paper depending on whether we have a node – member or node – paper.

Finally, for each edge e=(u,v) we define a variable x_e constructing a column vector x with E rows.

Thus, the representation of the problem as an integer programming one is:

$$\begin{aligned} &\text{maximize} && c^T x \\ &\text{subject to} && a \leq Ax \leq b \\ & && x \in \{0,1\}, \end{aligned}$$

where each x can take the values {0,1} since an edge either will be chosen or not.

And in matrix representation it is:

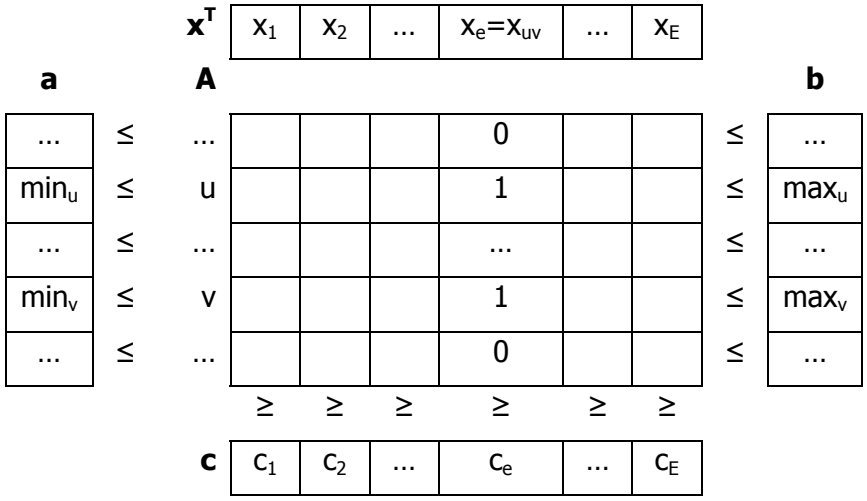

Figure 4: Matrix represantation of the problem

After having constructed the problem by the graph, we will now prove the reduction:

d_P = «yes» ⇨ d_{ILP} = «yes»:
If the instance P is true, which means that there is a set of edges that, given the constraints, has the greatest possible value, then there is a vector x for the ILP that satisfies the inequality $a \leq Ax \leq b$ and has the greatest value.

d_{ILP} = «yes» ⇨ d_P = «yes»:
If the instance ILP is true, which means that there is a vector x with the greatest value satisfying the inequality $a \leq Ax \leq b$, then there is a set of edges that for the given constraints per committee member and per paper has also the greatest value.

3.3 Solution Integrality

Our problem is a sub case of ILP which, in general, is NP-Complete. However, it is a special case, since its matrix of constraints is totally unimodular (see below).

In [2] there are theorems proving that given a bipartite graph and integral data of the problem, the basic feasible solution will be integral. Thus, ensuring the solution integrality, we can solve it as a simple linear program converting the constraint "$x \in \{0,1\}$" to "$x \in [0,1]$".

Before writing the theorems, we will first give the following definition: A matrix A is *totally unimodular* if each subdeterminant of A is 0, +1 or -1. In particular each entry in a total unimodular matrix is 0, +1 or -1.

Theorem 1: Let G=(V,E) be an undirected graph, and let A be the VxE-incidence matrix of G (i.e. A is the {0,1}-matrix with rows and columns indexed by the vertices and edges of G, respectively, where $A_{v,e}=1$ if and only if $v \in e$).Then:

"A is totally unimodular if G is bipartite".

Theorem 2: Let A be a matrix with entries 0, +1 or -1. Then for all integral vectors a, b, c, d the polyhedron {x|c<=x<=d;a<=Ax<=b} has only integral vertices.

Since our problem's graph is bipartite, according to theorem 1 the matrix A is totally unimodular. But by the second theorem, since A is totally unimodular and all of our constraints are integral, the problem will only have integral basic solutions.

Thus, we can find an integral feasible basic solution in polynomial time [2]. Although the guaranteed polynomial solution is given by the ellipsoid method [2], we use the Simplex method [1,2,6] to solve our problem since its average behavior is quicker. Replacing the constraint "$x \in \{0,1\}$" with "$x \in [0,1]$", we now write and solve the problem as:

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & a \le Ax \le b \\ & 0 \le x \le 1 \end{array}$$

Chapter 4

Implementation and Testing

Examples were executed in the linear programming tool Soplex 1.3.1 in operating system Suse 10.2. This tool takes files in lp-format as input and returns the value of the objective function along with the value of x.

Simple examples, being easy, can be written in lp-format by hand. To run more complicated problems an automatization of generating the problem to lp-format by the data is necessary. Thus, a file in programming language C has been written. This file, lpgeneration.c, takes four txt files as input, memtopic.txt, toppaper.txt, bids.txt and data.txt. After processing them, it generates two intermediate files, vari.txt and c.txt, and in the end, it generates the desirable output file, lpfile.lp, along with a file informing for infeasibility, infeasibility.txt.

We will now describe each file separately.

4.1 memtopic.txt

In each line of this file, there is the number of the committee member and the number of topics that the member is interested in followed by the number of each topic. The numbers of the committee members and of the topics must be incrementally ordered.

Thus, its format is:

1st line:	1st examiner	number_of_topics	topic's_no	topic's_no	...
2nd line:	2nd examiner	number_of_topics	topic's_no	topic's_no	...

.
.

Example

We have three examiners. The 1st one is interested in two topics, the topic with number 1 and the topic with number 3. The 2nd one is interested in one topic with number 2. Finally, the 3rd one is interested in 2 topics, with number 2 and 4 respectively. The file will be:

1	2	1	3
2	1	2	
3	2	2	4

4.2 toppaper.txt

In each line of this file, there is the number of the topic and the number of papers that refer to it followed by the number of each paper. The numbers of the topics and of the papers must be incrementally ordered.

Thus, its format is:

1st line:	1st topic	number_of_papers	paper_no	paper_no	...
2nd line:	2nd topic	number_of_papers	paper_no	paper_no	...

.
.
.

Example

We have three topics. The 1st one has 8 papers referring to it, those with numbers 1, 3, 4, 6, 10, 12, 15 and 17. The 2nd one has 3 papers with numbers 2, 5 and 16. Finally, the 3rd one has 5 papers with numbers 7, 8, 9, 11 and 13. The file will be:

1	8	1	3	4	6	10	12	15	17
2	3	2	5	16					
3	5	7	8	9	11	13			

4.3 vari.txt

It comes from the processing of the previous files. In each line, there is the number of the committee member, the number of the paper and a value in {0,1} for all the members and for all the papers. Value 1 means that the member of the line can examine the paper of that line, while 0 means that he can't. It is the equal of having or not an edge between this member and this paper. The numbers of the members and the papers must be incrementally ordered.

Its format is:

1st line:	1st member	1st paper	1 or 0
2nd line:	1st member	2nd paper	1 or 0

.
.
.

Example

1	1	1
1	2	1
1	3	0
2	1	0
2	2	1
2	3	0

4.4 bids.txt

This file contains the special preferences of the committee members for some, if any, papers. These preferences may be:

- 'H', which is high interest,
- 'M', which is medium interest,
- 'L', which is low interest,
- 'N', which is no interest and
- 'C', which is conflict.

Each line has the number of the member followed by the number of the paper and the bids for it. It is not necessary all of the members to specify bids for all of the papers. Again, both the numbers of the members and the papers must be incrementally ordered.

The file's format is:

1st line: number_of_member number_of_papers bids
2nd line: number_of_member number_of_papers bids
.
.
.

Example
We have three members and six papers. The 1st member is highly interested in paper with number 4 and the 3rd person is low interested in paper with number 2. The file will be:

1 4 H
3 2 L

4.5 c.txt

This is another intermediate file. In each line it has the matching degree between the committee member and the paper.

The greatest matching degree is considered to be number 4 and the lower 0. When a paper belongs to a topic that a member is interested in, the respective edge is valued by 1. If the member has also declared special interest for this paper (in bids.txt) this preference is added to 1. So, if he declares:

- 'H' (high interest), the degree will be 1+3=4,
- 'M' (medium interest), the degree will be 1+2=3,
- 'L' (low interest), the degree will be 1+1=2,
- 'N' (no interest), the degree will be 1+0=1 and
- 'C' (conflict), the degree will be 1-1=0.

Supposing we have m members and n papers, its format is:

1st line:	Weight of the edge joining the 1st member and the 1st paper
2nd line:	Weight of the edge joining the 1st member and the 2nd paper
3rd line:	Weight of the edge joining the 1st member and the 3rd paper
.	.
.	.
.	.
mn-th line:	Weight of the edge joining the m^{th} member and the n^{th} paper

Example

For three members and four papers, it could be:

2
1
0
1
0
1
3
2
1
0
1
4

4.6 data.txt

It is assumed that there might be special cases where not all the committee members will have the same upper and lower bound of papers to examine and / or not all papers will have the same upper and lower bound of members that should examine them.

It contains the following information:

1st line:	Number of committee members
2^{n} line:	Number of submitted papers

3η line: Number of different constraints for members
4η line: Number of different constraints for papers
5η line: 1st member of the 1st constraint
6η line: Last member of the 1st constraint
7η line: Maximum number of papers
8η line: Minimum number of papers
.
.
.
: 1st member of the last^t constraint
: Last member of the last constraint
: Maximum number of papers
: Minimum number of papers
: 1st paper of the 1st constraint
: Last paper of the 1st constraint
: Maximum number of members
: Minimum number of members
.
.
.
: 1st paper of the last constraint
: Last paper of the last constraint
: Maximum number of members
: Minimum number of members

Example

We have 3 members and 6 papers. Each member can examine [1, 3] papers. Papers 1 to 2 can be examined by [1, 2] members, papers 3 to 5 by [2,2] members and paper 6 by [1, 2]. The file will be:

3 (Number of members)
6 (Number of papers)
1 (Number of different constraints for members)
3 (Number of different constraints for papers)
1 (1st member of the 1st constraint)
3 (Last member of the 1st constraint)
3 (Maximum number of papers)

1 (Minimum number of papers)
1 (1st paper of the 1st constraint)
2 (Last paper of the 1st constraint)
2 (Maximum number of members)
1 (Minimum number of members)
3 (1st paper of the 2nd constraint)
5 (Last paper of the 2nd constraint)
2 (Maximum number of members)
2 (Minimum number of members)
6 (1st paper of the 3rd constraint)
6 (Last paper of the 3rd constraint)
2 (Maximum number of members)
1 (Minimum number of members)

4.7 lpgeneration.c

```
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{

	// file variables
	FILE *f1;
	FILE *f2;
	FILE *f3;
	FILE *f4;
	FILE *f5;
	FILE *f6;
	FILE *f7;
	FILE *f8;

	// other variables
	int m,n,member,paper,mem,pap,weight,wght,ctop,ntop,ntopic,cpaper,npaper,wvr;
	char bids;
	int i,j,k,l,c1,t;
	int *dt;
	int lm,ln,max,min;
	int fsbl,w;

	/***** FINDING THE WEIGHT OF EACH EDGE *****/

	/* open memtopic.txt in read mode */
	f1 = fopen("/root/lpgeneration/lpgeneration/src/memtopic.txt","r");

	/* open vari.txt in write mode */
	f3 = fopen("/root/lpgeneration/lpgeneration/src/vari.txt","w");
```

```
/* open bids.txt in read mode */
f4 = fopen("/root/lpgeneration/lpgeneration/src/bids.txt","r");

/* open c.txt in write mode */
f5 = fopen("/root/lpgeneration/lpgeneration/src/c.txt","w");

/* open data.txt in read mode */
f6 = fopen("/root/lpgeneration/lpgeneration/src/data.txt","r");

// reading data from data.txt
fscanf(f6,"%d",&m);
fscanf(f6,"%d",&n);


dt = (int *)malloc(n*sizeof(int));

for (i=1;i<=m;i++){        // for each member, check the topics he is interested in
        f2 = fopen("/root/lpgeneration/lpgeneration/src/toppaper.txt","r");
        for (k=0;k<n;k++)
                *(dt+k) = 0;
        fscanf(f1,"%d",&member);
        if (member == i) {
                fscanf(f1,"%d",&ctop);      // number of topics
                for (j=1;j<=ctop;j++) {             // what are the papers of each topic?
                        fscanf(f1,"%d",&ntop);
                        do {
                                fscanf(f2,"%d",&ntopic);
                                if (ntopic != ntop) {
                                        fscanf(f2,"%d",&cpaper);
                                        for (k=1;k<=cpaper;k++)
                                                fscanf(f2,"%d",&wvr);
                                }
                         } while (ntopic != ntop);
                        fscanf(f2,"%d",&cpaper);
                        for (k=1;k<=cpaper;k++) {
                                fscanf(f2,"%d",&npaper);
                                *(dt+(npaper-1)) = 1;       // for each paper of the topic,
                                                            // there is an edge weighted by 1
                        }
                }
                for (k=1;k<=n;k++) {      // print the weights
                        fprintf(f3,"%d %d %d\n",member,k,*(dt+(k-1)));
                }
        }
        else {
                weight = 0;
                for (k=1;k<=n;k++) {
                        fprintf(f3,"%d %d %d\n",member,k,weight);
                }
        }
        // close toppaper.txt
        fclose(f2);
}

// close memtopic.txt
fclose(f1);

free((int *)dt);

// close vari.txt
fclose(f3);

/* open vari.txt in read mode*/
```

```
f3 = fopen("/root/lpgeneration/lpgeneration/src/vari.txt","r");

fscanf(f4,"%d %d %s\n",&mem,&pap,&bids);
for (i=1;i<=m*n;i++) {
        fscanf(f3,"%d %d %d\n",&member,&paper,&weight);
        if ((member == mem) && (paper == pap) && (weight != 0)) {// find the total edge
                                                                  // weight and print it to c.txt
                switch (bids) {
                        case 'H': wght = 3;
                                break;
                        case 'M': wght = 2;
                                break;
                        case 'L': wght = 1;
                                break;
                        case 'N': wght = 0;
                                break;
                        case 'C': wght = (-1);
                                break;
                        default : wght = 0;
                }
                fprintf(f5,"%d\n",(weight+wght));
                fscanf(f4,"%d %d %s\n",&mem,&pap,&bids);
        }
        else
                fprintf(f5,"%d\n",weight);
}

// close the open files
fclose(f3);
fclose(f4);
fclose(f5);


/**** CHECKING IF THERE ARE COMMITTEE MEMBERS WITH NO PAPERS TO EXAMINE ****/

/* open infeasibility.txt in write mode */
f8 = fopen("/root/lpgeneration/lpgeneration/src/infeasibility.txt","w");

/* open c.txt in read mode */
f5 = fopen("/root/lpgeneration/lpgeneration/src/c.txt","r");

for (j=1; j<=m; j++){
        i=1;
        fsbl=0;

        while ((fsbl == 0) && (i <= n)) {
                fscanf(f5,"%d\n",&w);
                if (w > 0) {
                        fsbl = 1;
                        if (i < n)
                                for (t=0; t<n-i; t++)
                                        fscanf(f5,"%d\n",&w);
                }
                i++;
        }

        if (fsbl == 0)
                fprintf(f8,"Member %d has no papers to examine.\n",j);
}
fclose(f5);


/**** CHECKING IF THERE ARE PAPERS WITH NO MEMBER TO EXAMINE THEM ****/
for (j=0; j<n; j++) {
```

```
		/* open c.txt in read mode */
		f5 = fopen("/root/lpgeneration/lpgeneration/src/c.txt","r");
		i=j+1;
		fsbl=0;
		for (t=1; t<i; t++)
			fscanf(f5,"%d\n",&w);

		while ((fsbl == 0) && (i<=m*n)) {
			fscanf(f5,"%d\n",&w);
			if (w > 0)
				fsbl = 1;
			if (i < m*n)
				for (t=1; t<n; t++)
					fscanf(f5,"%d\n",&w);
			i = i + n;
		}
		if (fsbl == 0)
			fprintf(f8,"There are no members to examine paper %d.\n",j+1);

		fclose(f5);
	}

	// reading from data.txt
	fscanf(f6,"%d",&lm);
	fscanf(f6,"%d",&ln);

	dt = (int *)malloc(((lm+ln)*4)*sizeof(int));
	for (i=0;i<(lm+ln)*4;i++) {
		fscanf(f6,"%d",(dt+i));
	}


	/***** CHECKING FOR INAPPROPRIATE CONSTRAINTS *****/
	if ((lm == 1) && (ln == 1))
		if (m*(*(dt+2)) < n*(*(dt+7)))
			fprintf(f8,"Inappropriate constraints. Either the maximum papers per member
or the minimum members per paper should be changed.\n");


	fclose(f8);


	/***** GENERATING THE PROBLEM IN LP-FORMAT *****/

	/* open c.txt in read mode */
	f5 = fopen("/root/lpgeneration/lpgeneration/src/c.txt","r");

	/* open lpfile.lp in write mode */
	f7 = fopen("/root/lpgeneration/lpgeneration/src/lpfile.lp","w");

	fprintf(f7,"Maximize\n");

	// generating the objective function
	for (i=1;i<=m*n;i++){
		fscanf(f5,"%d",&c1);
		fprintf(f7," + %d",c1);
		fprintf(f7," x");
		fprintf(f7,"%d",i); // e.g. " + 5 x1"
	}
	fprintf(f7,"\n");

	// close c.txt
	fclose(f5);
```

```
fprintf(f7,"Subject to\n");

// constraints per member
j = 0;
i = 0;
for (k=0; k<4*lm; k=k+4)
	for (l=*(dt+k); l<=*(dt+(k+1)); l++){
		max = *(dt +(k+2));
		min = *(dt +(k+3));

		// for max
		for (i=j+1; i<=j+n; i++) {
			fprintf(f7," + x");
			fprintf(f7,"%d",i);
		}
		fprintf(f7," <= %d\n",max);

		// for min
		for (i=j+1; i<=j+n; i++) {
			fprintf(f7," + x");
			fprintf(f7,"%d",i);
		}
		fprintf(f7," >= %d\n",min);

		j = j + n;
	}


// constraints per paper
j = 0;
i = 0;
for (k=0; k<4*ln; k=k+4)
	for (l=*(dt+(4*lm+k)); l<=*(dt+(4*lm+(k+1))); l++){
		max = *(dt + (4*lm + (k+2)));
		min = *(dt + (4*lm + (k+3)));

		// for max
		for (i=j+1; i<=m*n; i=i+n) {
			fprintf(f7," + x");
			fprintf(f7,"%d",i);
		}
		fprintf(f7," <= %d\n",max);

		// for min
		for (i=j+1; i<=m*n; i=i+n) {
			fprintf(f7," + x");
			fprintf(f7,"%d",i);
		}
		fprintf(f7," >= %d\n",min);

		j = j + 1;
	}


// close data.txt
fclose(f6);

free((int *)dt);

fprintf(f7,"Bounds\n");

// each variable is bounded in [0,1]
for (j=1; j<=m*n; j++) {
	fprintf(f7,"x");
```

```
                fprintf(f7,"%d >= 0\n",j);  // e.g. x1 >= 0
                fprintf(f7,"x");
                fprintf(f7,"%d <= 1\n",j);
        }

        fprintf(f7,"END");

        // close lpile.lp
        fclose(f7);

        return EXIT_SUCCESS;
}
```

4.8 lpfile.lp

This is the output file with the problem in lp-format, that is:

Maximize	(in the first line)
objective function	(in the second line)
Subject to	(in the third one)
constraints per member and per paper	(in the next lines…)
Bounds	(after the constraints)
all variables bounded in [0,1]	(in the next lines)
END	(in the last line)

As we can see, there are defined m*n variables. The first n variables correspond to the edges joining the first member with each of the n papers. The next n variables correspond to the edges joining the second member with each of the n papers. So goes on until the last n variables that correspond to the edges joining the last member with each of the n papers.

Example

```
Maximize
 + 1 x1 + 4 x2 + 0 x3 + 1 x4 + 2 x5 + 0 x6 + 2 x7 + 1 x8 + 1 x9 + 3 x10 + 1 x11 + 1 x12 + 1 x13 + 0
x14 + 2 x15 + 0 x16 + 0 x17 + 1 x18
Subject to
 + x1 + x2 + x3 + x4 + x5 + x6 <= 3
 + x1 + x2 + x3 + x4 + x5 + x6 >= 2
 + x7 + x8 + x9 + x10 + x11 + x12 <= 3
 + x7 + x8 + x9 + x10 + x11 + x12 >= 2
 + x13 + x14 + x15 + x16 + x17 + x18 <= 3
 + x13 + x14 + x15 + x16 + x17 + x18 >= 2
 + x1 + x7 + x13 <= 3
 + x1 + x7 + x13 >= 1
 + x2 + x8 + x14 <= 3
 + x2 + x8 + x14 >= 1
 + x3 + x9 + x15 <= 3
 + x3 + x9 + x15 >= 1
 + x4 + x10 + x16 <= 3
```

```
 + x4 + x10 + x16 >= 1
 + x5 + x11 + x17 <= 3
 + x5 + x11 + x17 >= 1
 + x6 + x12 + x18 <= 3
 + x6 + x12 + x18 >= 1
Bounds
x1 >= 0
x1 <= 1
x2 >= 0
x2 <= 1
x3 >= 0
x3 <= 1
x4 >= 0
x4 <= 1
x5 >= 0
x5 <= 1
x6 >= 0
x6 <= 1
x7 >= 0
x7 <= 1
x8 >= 0
x8 <= 1
x9 >= 0
x9 <= 1
x10 >= 0
x10 <= 1
x11 >= 0
x11 <= 1
x12 >= 0
x12 <= 1
x13 >= 0
x13 <= 1
x14 >= 0
x14 <= 1
x15 >= 0
x15 <= 1
x16 >= 0
x16 <= 1
x17 >= 0
x17 <= 1
x18 >= 0
x18 <= 1
END
```

It is easily seen by the example that the double constraints $a \leq Ax \leq b$ have been written as $Ax \leq b$ and $Ax \geq a$ and the bounds $0 \leq x \leq 1$ as $x \leq 1$ and $x \geq 0$. However, Soplex identifies them and handles them as double.

4.9 infeasibility.txt

In this file there are messages informing the user in cases the problem is infeasible, before giving it as an input to a linear programming tool, such as Soplex.

It informs for three cases of infeasibility. The case where one or more members are interested in only one topic with no submitted papers or where papers are submitted in

a topic that has not been in the interest of any member. Finally, the case where the constraints are not well defined and have to be changed. The user, considering the infeasibility message(s), may make the suitable changes and rerun the procedure to obtain a feasible solution.

When no infeasibility is detected, this file is empty.

Chapter 5

Results

We will now see some simple examples of those we have run in Soplex and their results.

5.1 Three members and six papers

In the first three examples, we consider the case of three committee members and six submitted papers. 3x6=18 variables are defined by the lpgeneration.c:

- Variables x1 to x6 correspond to the edges joining the first member with each of the six papers,
- Variables x7 to x12 correspond to the edges joining the second member with each of the six papers,
- Variables x13 to x18 correspond to the edges joining the third member with each of the six papers.

As maximum matching degree is considered number 4.

5.1.1 First example

We will describe each file separately. It is reminded that the input files are: memtopic.txt, toppaper.txt, bids.txt and data.txt and the output files are lpfile.lp and infeasibility.txt. Giving lpfile.lp as input to Soplex, we take the results.

5.1.1.1 memtopic.txt

This file contains:

1 1 2

2 2 1 2

3 1 1

The 1st member is interested in one topic numbered with 2. The 2nd member is interested in two topics numbered by 1 and 2 respectively. The 3rd one is interested in one topic numbered by 1.

5.1.1.2 toppaper.txt

This file contains:

1 4 1 3 5 6

2 3 2 4 5

The 1st topic has 4 papers numbered with 1, 3, 5 and 6 respectively. The 2nd one has 3 papers numbered with 2, 5 and 6 respectively.

5.1.1.3 vari.txt

It is an intermediate file and contains:

1 1 0

1 2 1

1 3 0

1 4 1

1 5 1

1 6 0

2 1 1

2 2 1

2 3 1

2 4 1

2 5 1

2 6 1

3 1 1

3 2 0

3 3 1

3 4 0

3 5 1

3 6 1

The 1st member can examine papers 2, 4 and 5 (those having edge-value 1). The 2nd one can examine all of the papers. The 3rd member can examine papers 1, 3, 5 and 6.

5.1.1.4 bids.txt

This file contains:

1 2 H

1 5 L

2 1 L

2 4 M

3 3 L

3 5 C

3 6 N

The 1st member is highly interested in paper 2 and low in 5. The 2nd member is low interested in paper 1 and medium in 4. The 3rd one is low interested in paper 3, declares conflict for 5 and has no interest for 6.

5.1.1.5 c.txt

This file contains:

0

4

0

1

2

0

2

1

1

3

1

1

1

0

2

0

0

1

The 1st edge joining the 1st member to the 1st paper has weight 0. The 2nd edge joining the 1st member to the 2nd paper has weight 4. And so goes on.

5.1.1.6 data.txt

This files contains:

3

6

2

1

1

2

3

1

3

3

2

1

1

6

2

1

We have 3 members and 6 papers. There are 2 different constraints for the members and 1 for papers. Members 1 to 2 may examine [1,3] papers. Member 3 (3 to 3) may examine [1,2] papers. Papers 1 to 6 may be examined by [1,2] members.

5.1.1.7 lpfile.lp

It has the problem in lp-format:

```
Maximize
 + 0 x1 + 4 x2 + 0 x3 + 1 x4 + 2 x5 + 0 x6 + 2 x7 + 1 x8 + 1 x9 + 3 x10 + 1 x11 + 1 x12 + 1 x13 + 0
x14 + 2 x15 + 0 x16 + 0 x17 + 1 x18
Subject to
 + x1 + x2 + x3 + x4 + x5 + x6 <= 3
 + x1 + x2 + x3 + x4 + x5 + x6 >= 1
 + x7 + x8 + x9 + x10 + x11 + x12 <= 3
 + x7 + x8 + x9 + x10 + x11 + x12 >= 1
 + x13 + x14 + x15 + x16 + x17 + x18 <= 2
```

```
 + x13 + x14 + x15 + x16 + x17 + x18 >= 1
 + x1 + x7 + x13 <= 2
 + x1 + x7 + x13 >= 1
 + x2 + x8 + x14 <= 2
 + x2 + x8 + x14 >= 1
 + x3 + x9 + x15 <= 2
 + x3 + x9 + x15 >= 1
 + x4 + x10 + x16 <= 2
 + x4 + x10 + x16 >= 1
 + x5 + x11 + x17 <= 2
 + x5 + x11 + x17 >= 1
 + x6 + x12 + x18 <= 2
 + x6 + x12 + x18 >= 1
Bounds
x1 >= 0
x1 <= 1
x2 >= 0
x2 <= 1
x3 >= 0
x3 <= 1
x4 >= 0
x4 <= 1
x5 >= 0
x5 <= 1
x6 >= 0
x6 <= 1
x7 >= 0
x7 <= 1
x8 >= 0
x8 <= 1
x9 >= 0
x9 <= 1
x10 >= 0
x10 <= 1
x11 >= 0
x11 <= 1
x12 >= 0
x12 <= 1
x13 >= 0
x13 <= 1
x14 >= 0
x14 <= 1
x15 >= 0
x15 <= 1
x16 >= 0
x16 <= 1
x17 >= 0
x17 <= 1
x18 >= 0
x18 <= 1
END
```

5.1.1.8 infeasibility.txt

Since no infeasibility was detected, this file is empty.

5.1.1.9 results

Variables:

x2

x4

x5

x7

x10

x12

x15

x18 are one. All others are zero.

As it is seen, all variables value is either 0 or 1, so the solution is, indeed, integral. The following table shows results analytically.

members / papers	1st	2nd	3rd	number of members per paper ([1,2])
1st	0	2 x	1	1
2nd	4 x	1	0	1
3rd	0	1	2 x	1
4th	1 x	3 x	0	2
5th	2 x	1	0	1
6th	0	1 x	1 x	2
number of papers per member ([1,3] or [1,2])	3	3	2	

Table 1: Case 1 of 3 members and 6 papers.

So, each member will examine the maximum possible number of papers and each paper will be examined by 1 or 2 members. The edges chosen are those with the greatest possible value.

5.1.2 Second example

This example is the same with the first one with one change in file toppaper.txt. So, we will only describe files toppaper.txt, lpfile.lp and the results. Infeasibility.txt is again empty.

5.1.2.1 toppaper.txt

1 4 1 3 5 6

2 4 1 2 4 5

Now, topic 2 has 4 instead of 3 papers. Paper 1 is added.

5.1.2.2 lpfile.lp

This file contains the new lp-format:

```
Maximize
 + 1 x1 + 4 x2 + 0 x3 + 1 x4 + 2 x5 + 0 x6 + 2 x7 + 1 x8 + 1 x9 + 3 x10 + 1 x11 + 1 x12 + 1 x13 + 0
x14 + 2 x15 + 0 x16 + 0 x17 + 1 x18
Subject to
 + x1 + x2 + x3 + x4 + x5 + x6 <= 3
 + x1 + x2 + x3 + x4 + x5 + x6 >= 1
 + x7 + x8 + x9 + x10 + x11 + x12 <= 3
 + x7 + x8 + x9 + x10 + x11 + x12 >= 1
 + x13 + x14 + x15 + x16 + x17 + x18 <= 2
 + x13 + x14 + x15 + x16 + x17 + x18 >= 1
 + x1 + x7 + x13 <= 2
 + x1 + x7 + x13 >= 1
 + x2 + x8 + x14 <= 2
 + x2 + x8 + x14 >= 1
 + x3 + x9 + x15 <= 2
 + x3 + x9 + x15 >= 1
 + x4 + x10 + x16 <= 2
 + x4 + x10 + x16 >= 1
 + x5 + x11 + x17 <= 2
 + x5 + x11 + x17 >= 1
 + x6 + x12 + x18 <= 2
 + x6 + x12 + x18 >= 1
Bounds
x1 >= 0
x1 <= 1
x2 >= 0
x2 <= 1
x3 >= 0
x3 <= 1
x4 >= 0
x4 <= 1
x5 >= 0
x5 <= 1
x6 >= 0
x6 <= 1
x7 >= 0
x7 <= 1
x8 >= 0
x8 <= 1
x9 >= 0
x9 <= 1
x10 >= 0
x10 <= 1
x11 >= 0
x11 <= 1
x12 >= 0
x12 <= 1
x13 >= 0
x13 <= 1
x14 >= 0
x14 <= 1
x15 >= 0
x15 <= 1
x16 >= 0
x16 <= 1
```

```
x17 >= 0
x17 <= 1
x18 >= 0
x18 <= 1
END
```

5.1.2.3 results

Variables:
x2
x4
x5
x7
x10
x12
x15
x18 are one. All others are zero.

The results are shown in the following table analytically.

members / papers	1st	2nd	3rd	number of members per paper ([1,2])
1st	1	2 x	1	1
2nd	4 x	1	0	1
3rd	0	1	2 x	1
4th	1 x	3 x	0	2
5th	2 x	1	0	1
6th	0	1 x	1 x	2
number of papers per member ([1,3] or [1,2])	3	3	2	

Table 2: Case 2 of 3 members and 6 papers.

The solution hasn't change at all. The only thing that it has been changed from 0 to 1 is the weight of the first edge.

5.1.3 Third example

This example is the same with the second one with a change this time in file data.txt. So, we will only describe files data.txt, lpfile.lp and the results. Again no infeasibility was detected.

5.1.3.1 data.txt

3

6

1

1

1

3

3

2

1

6

3

1

In this example, we have 3 members, 6 papers and one constraint per member and per paper. Al members from 1 to 3 may examine [2,3] papers and all papers from 1 to 6 may be examined by [1,3] members.

5.1.3.2 lpfile.lp

The new lp file is:

```
Maximize
 + 1 x1 + 4 x2 + 0 x3 + 1 x4 + 2 x5 + 0 x6 + 2 x7 + 1 x8 + 1 x9 + 3 x10 + 1 x11 + 1 x12 + 1 x13 + 0
x14 + 2 x15 + 0 x16 + 0 x17 + 1 x18
Subject to
 + x1 + x2 + x3 + x4 + x5 + x6 <= 3
 + x1 + x2 + x3 + x4 + x5 + x6 >= 2
 + x7 + x8 + x9 + x10 + x11 + x12 <= 3
 + x7 + x8 + x9 + x10 + x11 + x12 >= 2
 + x13 + x14 + x15 + x16 + x17 + x18 <= 3
 + x13 + x14 + x15 + x16 + x17 + x18 >= 2
 + x1 + x7 + x13 <= 3
 + x1 + x7 + x13 >= 1
 + x2 + x8 + x14 <= 3
 + x2 + x8 + x14 >= 1
 + x3 + x9 + x15 <= 3
 + x3 + x9 + x15 >= 1
 + x4 + x10 + x16 <= 3
 + x4 + x10 + x16 >= 1
 + x5 + x11 + x17 <= 3
 + x5 + x11 + x17 >= 1
 + x6 + x12 + x18 <= 3
 + x6 + x12 + x18 >= 1
Bounds
x1 >= 0
x1 <= 1
x2 >= 0
x2 <= 1
```

```
x3 >= 0
x3 <= 1
x4 >= 0
x4 <= 1
x5 >= 0
x5 <= 1
x6 >= 0
x6 <= 1
x7 >= 0
x7 <= 1
x8 >= 0
x8 <= 1
x9 >= 0
x9 <= 1
x10 >= 0
x10 <= 1
x11 >= 0
x11 <= 1
x12 >= 0
x12 <= 1
x13 >= 0
x13 <= 1
x14 >= 0
x14 <= 1
x15 >= 0
x15 <= 1
x16 >= 0
x16 <= 1
x17 >= 0
x17 <= 1
x18 >= 0
x18 <= 1
END
```

5.1.3.3 results

Variables:
x2
x4
x5
x7
x10
x12
x13
x15
x18 are one. All others are zero.

The following table shows results analytically.

members / papers	1st	2nd	3rd	number of members per paper ([1,3])
1st	0	2 x	1 x	2
2nd	4 x	1	0	1
3rd	0	1	2 x	1
4th	1 x	3 x	0	2
5th	2 x	1	0	1
6th	0	1 x	1 x	2
number of papers per member ([2,3])	3	3	3	

Table 3: Case 3 of 3 members and 6 papers.

The only change in solution is that the 1st member will also examine the 3rd paper.

5.2 Four members and twelve papers

In the following three examples, we consider the case of four committee members and twelve submitted papers. 4x12=48 variables are defined by the lpgeneration.c:

- Variables x1 to x12 correspond to the edges joining the first member with each of the twelve papers,
- Variables x13 to x24 correspond to the edges joining the second member with each of the twelve papers,
- Variables x25 to x36 correspond to the edges joining the third member with each of the twelve papers,
- Variables x37 to x48 correspond to the edges joining the fourth member with each of the twelve papers.

As maximum matching degree is considered number 4.

5.2.1 Fisrt example

We will describe each file separately.

5.2.1.1 memtopic.txt

This file contains:

1 2 1 3

2 1 2

3 1 3

4 2 1 2

The 1st member is interested in two topics numbered with 1 and 3 respectively. The 2nd member is interested in one topic numbered with 2. The 3rd one is interested in one topic numbered by 3. And the 4th member is interested in two topics numbered with 1 and 2 respectively.

5.2.1.2 toppaper.txt

This file contains:

1 4 1 4 6 8

2 6 1 3 5 10 11 12

3 5 2 5 7 9 11

The 1st topic has 4 papers numbered with 1, 4, 6 and 8 respectively. The 2nd one has 6 papers numbered with 1, 3, 5, 10, 11 and 12 respectively. The 3rd one has 5 papers numbered with 2, 5, 7, 9 and 11 respectively.

5.2.1.3 vari.txt

It is an intermediate file and contains:

1 1 1

1 2 1

1 3 0

1 4 1

1 5 1

1 6 1

1 7 1

1 8 1

1 9 1

1 10 0

1 11 1
1 12 0
2 1 1
2 2 0
2 3 1
2 4 0
2 5 1
2 6 0
2 7 0
2 8 0
2 9 0
2 10 1
2 11 1
2 12 1
3 1 0
3 2 1
3 3 0
3 4 0
3 5 1
3 6 0
3 7 1
3 8 0
3 9 1
3 10 0
3 11 1
3 12 0
4 1 1
4 2 0
4 3 1
4 4 1
4 5 1
4 6 1
4 7 0
4 8 1
4 9 0

4 10 1

4 11 1

4 12 1

The 1st member can examine papers 1, 2, 4, 5, 6, 7, 8, 9 and 11 (those having edge-value 1). The 2nd one can examine papers 1, 3, 5, 10, 11 and 12. The 3rd member can examine papers 2, 5, 7, 9 and 11 The 4th one can examine papers 1, 3, 4, 5, 6, 8, 10, 11 and 12.

5.2.1.4 bids.txt

This file contains:

1 2 H

1 7 L

1 8 C

2 10 H

3 5 H

3 11 M

4 12 M

The 1st member is highly interested in paper 2, low in 7 and declares conflict for 8. The 2nd member is highly interested in paper 10. The 3rd one is highly interested in paper 5 and medium in 11. The 4th is medium interested in paper 12.

5.2.1.5 c.txt

This file contains:

1

4

0

1

1

1

2

0

1

0

1

0

1
0
1
0
1
0
0
0
0
4
1
1
0
1
0
0
4
0
1
0
1
0
3
0
1
0
1
1
1
1
0
1
0
1
1

3

The 1st edge joining the 1st member to the 1st paper has weight 1. The 2nd edge joining the 1st member to the 2nd paper has weight 4. And so goes on.

5.2.1.6 data.txt

This files contains:

4

12

1

1

1

4

7

3

1

12

3

1

We have 4 members and 12 papers. There is 1 constraint for the members and 1 for the papers. Members 1 to 4 may examine [3,7] papers. Papers 1 to 12 may be examined by [1,3] members.

5.2.1.7 lpfile.lp

It has the problem in lp-format:

```
Maximize
 + 1 x1 + 4 x2 + 0 x3 + 1 x4 + 1 x5 + 1 x6 + 2 x7 + 0 x8 + 1 x9 + 0 x10 + 1 x11 + 0 x12 + 1 x13 + 0
x14 + 1 x15 + 0 x16 + 1 x17 + 0 x18 + 0 x19 + 0 x20 + 0 x21 + 4 x22 + 1 x23 + 1 x24 + 0 x25 + 1 x26
+ 0 x27 + 0 x28 + 4 x29 + 0 x30 + 1 x31 + 0 x32 + 1 x33 + 0 x34 + 3 x35 + 0 x36 + 1 x37 + 0 x38 + 1
x39 + 1 x40 + 1 x41 + 1 x42 + 0 x43 + 1 x44 + 0 x45 + 1 x46 + 1 x47 + 3 x48
Subject to
 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 <= 7
 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 >= 3
 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 <= 7
 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 >= 3
 + x25 + x26 + x27 + x28 + x29 + x30 + x31 + x32 + x33 + x34 + x35 + x36 <= 7
 + x25 + x26 + x27 + x28 + x29 + x30 + x31 + x32 + x33 + x34 + x35 + x36 >= 3
 + x37 + x38 + x39 + x40 + x41 + x42 + x43 + x44 + x45 + x46 + x47 + x48 <= 7
 + x37 + x38 + x39 + x40 + x41 + x42 + x43 + x44 + x45 + x46 + x47 + x48 >= 3
 + x1 + x13 + x25 + x37 <= 3
 + x1 + x13 + x25 + x37 >= 1
 + x2 + x14 + x26 + x38 <= 3
 + x2 + x14 + x26 + x38 >= 1
 + x3 + x15 + x27 + x39 <= 3
 + x3 + x15 + x27 + x39 >= 1
```

```
 + x4 + x16 + x28 + x40 <= 3
 + x4 + x16 + x28 + x40 >= 1
 + x5 + x17 + x29 + x41 <= 3
 + x5 + x17 + x29 + x41 >= 1
 + x6 + x18 + x30 + x42 <= 3
 + x6 + x18 + x30 + x42 >= 1
 + x7 + x19 + x31 + x43 <= 3
 + x7 + x19 + x31 + x43 >= 1
 + x8 + x20 + x32 + x44 <= 3
 + x8 + x20 + x32 + x44 >= 1
 + x9 + x21 + x33 + x45 <= 3
 + x9 + x21 + x33 + x45 >= 1
 + x10 + x22 + x34 + x46 <= 3
 + x10 + x22 + x34 + x46 >= 1
 + x11 + x23 + x35 + x47 <= 3
 + x11 + x23 + x35 + x47 >= 1
 + x12 + x24 + x36 + x48 <= 3
 + x12 + x24 + x36 + x48 >= 1
Bounds
x1 >= 0
x1 <= 1
x2 >= 0
x2 <= 1
x3 >= 0
x3 <= 1
x4 >= 0
x4 <= 1
x5 >= 0
x5 <= 1
x6 >= 0
x6 <= 1
x7 >= 0
x7 <= 1
x8 >= 0
x8 <= 1
x9 >= 0
x9 <= 1
x10 >= 0
x10 <= 1
x11 >= 0
x11 <= 1
x12 >= 0
x12 <= 1
x13 >= 0
x13 <= 1
x14 >= 0
x14 <= 1
x15 >= 0
x15 <= 1
x16 >= 0
x16 <= 1
x17 >= 0
x17 <= 1
x18 >= 0
x18 <= 1
x19 >= 0
x19 <= 1
x20 >= 0
x20 <= 1
x21 >= 0
x21 <= 1
x22 >= 0
x22 <= 1
x23 >= 0
```

```
x23 <= 1
x24 >= 0
x24 <= 1
x25 >= 0
x25 <= 1
x26 >= 0
x26 <= 1
x27 >= 0
x27 <= 1
x28 >= 0
x28 <= 1
x29 >= 0
x29 <= 1
x30 >= 0
x30 <= 1
x31 >= 0
x31 <= 1
x32 >= 0
x32 <= 1
x33 >= 0
x33 <= 1
x34 >= 0
x34 <= 1
x35 >= 0
x35 <= 1
x36 >= 0
x36 <= 1
x37 >= 0
x37 <= 1
x38 >= 0
x38 <= 1
x39 >= 0
x39 <= 1
x40 >= 0
x40 <= 1
x41 >= 0
x41 <= 1
x42 >= 0
x42 <= 1
x43 >= 0
x43 <= 1
x44 >= 0
x44 <= 1
x45 >= 0
x45 <= 1
x46 >= 0
x46 <= 1
x47 >= 0
x47 <= 1
x48 >= 0
x48 <= 1
END
```

5.2.1.8 infeasibility.txt

No infeasibility was detected, thus this file is empty.

5.2.1.9 results

Variables:

x1

x2
x4
x5
x6
x7
x9
x13
x15
x17
x22
x23
x24
x26
x29
x31
x33
x35
x39
x40
x42
x44
x46
x47
x48 are one. All others are zero.

The results are shown analytically in the following table.

members / papers	1st	2nd	3rd	4th	number of members per paper ([1,3])
1st	1 x	1 x	0	1	2
2nd	4 x	0	1 x	0	2
3rd	0	1 x	0	1 x	2
4th	1 x	0	0	1 x	2
5th	1 x	1 x	4 x	1	3
6th	1 x	0	0	1 x	2
7th	2 x	0	1 x	0	2
8th	0	0	0	1 x	1
9th	1 x	0	1 x	0	2
10th	0	4 x	0	1 x	2
11th	1	1 x	3 x	1 x	3
12th	0	1 x	0	3 x	2
number of papers per member ([3,7])	7	6	5	7	

Table 4: Case 1 of 4 members and 12 papers.

So, members will examine 5 to 7 papers and papers will be examined by 1 to 3 members. The edges chosen are those with the greatest possible value.

5.2.2 Second example

This example is the same with the previous one with a change in file bids.txt. So, we will only describe files bids.txt, lpfile.lp and the results. Infeasibility.txt is empty.

5.2.2.1 bids.txt

1 2 C

1 7 L

1 8 N

2 5 H

2 10 H

3 5 H

4 5 H

4 12 M

The 1st member now declares conflict for paper 2, is low interested in paper 7 and has no interest in paper 8. The 2nd member is highly interested in papers 5 and 10. The 3rd one is highly interested in paper 5. The 4th is highly interested in paper 5 and medium in paper 12. These changes affect c.txt which has the weights of the graph edges. The new values can be seen in the objective function of the following lp-format, where they are the co-efficients.

5.2.2.2 lpfile.lp

The problem in lp-format will now be

```
Maximize
 + 1 x1 + 0 x2 + 0 x3 + 1 x4 + 1 x5 + 1 x6 + 2 x7 + 1 x8 + 1 x9 + 0 x10 + 1 x11 + 0 x12 + 1 x13 + 0
x14 + 1 x15 + 0 x16 + 4 x17 + 0 x18 + 0 x19 + 0 x20 + 0 x21 + 4 x22 + 1 x23 + 1 x24 + 0 x25 + 1 x26
+ 0 x27 + 0 x28 + 4 x29 + 0 x30 + 1 x31 + 0 x32 + 1 x33 + 0 x34 + 1 x35 + 0 x36 + 1 x37 + 0 x38 + 1
x39 + 1 x40 + 4 x41 + 1 x42 + 0 x43 + 1 x44 + 0 x45 + 1 x46 + 1 x47 + 3 x48
Subject to
 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 <= 7
 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 >= 3
 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 <= 7
 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 >= 3
 + x25 + x26 + x27 + x28 + x29 + x30 + x31 + x32 + x33 + x34 + x35 + x36 <= 7
 + x25 + x26 + x27 + x28 + x29 + x30 + x31 + x32 + x33 + x34 + x35 + x36 >= 3
 + x37 + x38 + x39 + x40 + x41 + x42 + x43 + x44 + x45 + x46 + x47 + x48 <= 7
 + x37 + x38 + x39 + x40 + x41 + x42 + x43 + x44 + x45 + x46 + x47 + x48 >= 3
 + x1 + x13 + x25 + x37 <= 3
 + x1 + x13 + x25 + x37 >= 1
 + x2 + x14 + x26 + x38 <= 3
 + x2 + x14 + x26 + x38 >= 1
 + x3 + x15 + x27 + x39 <= 3
 + x3 + x15 + x27 + x39 >= 1
 + x4 + x16 + x28 + x40 <= 3
 + x4 + x16 + x28 + x40 >= 1
 + x5 + x17 + x29 + x41 <= 3
 + x5 + x17 + x29 + x41 >= 1
 + x6 + x18 + x30 + x42 <= 3
 + x6 + x18 + x30 + x42 >= 1
 + x7 + x19 + x31 + x43 <= 3
 + x7 + x19 + x31 + x43 >= 1
 + x8 + x20 + x32 + x44 <= 3
 + x8 + x20 + x32 + x44 >= 1
 + x9 + x21 + x33 + x45 <= 3
 + x9 + x21 + x33 + x45 >= 1
 + x10 + x22 + x34 + x46 <= 3
 + x10 + x22 + x34 + x46 >= 1
 + x11 + x23 + x35 + x47 <= 3
 + x11 + x23 + x35 + x47 >= 1
 + x12 + x24 + x36 + x48 <= 3
 + x12 + x24 + x36 + x48 >= 1
Bounds
x1 >= 0
x1 <= 1
x2 >= 0
x2 <= 1
x3 >= 0
```

x3 <= 1
x4 >= 0
x4 <= 1
x5 >= 0
x5 <= 1
x6 >= 0
x6 <= 1
x7 >= 0
x7 <= 1
x8 >= 0
x8 <= 1
x9 >= 0
x9 <= 1
x10 >= 0
x10 <= 1
x11 >= 0
x11 <= 1
x12 >= 0
x12 <= 1
x13 >= 0
x13 <= 1
x14 >= 0
x14 <= 1
x15 >= 0
x15 <= 1
x16 >= 0
x16 <= 1
x17 >= 0
x17 <= 1
x18 >= 0
x18 <= 1
x19 >= 0
x19 <= 1
x20 >= 0
x20 <= 1
x21 >= 0
x21 <= 1
x22 >= 0
x22 <= 1
x23 >= 0
x23 <= 1
x24 >= 0
x24 <= 1
x25 >= 0
x25 <= 1
x26 >= 0
x26 <= 1
x27 >= 0
x27 <= 1
x28 >= 0
x28 <= 1
x29 >= 0
x29 <= 1
x30 >= 0
x30 <= 1
x31 >= 0
x31 <= 1
x32 >= 0
x32 <= 1
x33 >= 0
x33 <= 1
x34 >= 0
x34 <= 1
x35 >= 0

```
x35 <= 1
x36 >= 0
x36 <= 1
x37 >= 0
x37 <= 1
x38 >= 0
x38 <= 1
x39 >= 0
x39 <= 1
x40 >= 0
x40 <= 1
x41 >= 0
x41 <= 1
x42 >= 0
x42 <= 1
x43 >= 0
x43 <= 1
x44 >= 0
x44 <= 1
x45 >= 0
x45 <= 1
x46 >= 0
x46 <= 1
x47 >= 0
x47 <= 1
x48 >= 0
x48 <= 1
END
```

5.2.2.3 results

Variables:

x1

x4

x6

x7

x8

x9

x11

x13

x15

x17

x22

x23

x24

x26

x29

x31

x33

x35

x39

x40

x41

x42

x44

x46

x48 are one. All others are zero.

The following table shows results analytically.

members / papers	1st	2nd	3rd	4th	number of members per paper ([1,3])
1st	1 x	1 x	0	1	2
2nd	0	0	1 x	0	1
3rd	0	1 x	0	1 x	2
4th	1 x	0	0	1 x	2
5th	1	4 x	4 x	4 x	3
6th	1 x	0	0	1 x	2
7th	2 x	0	1 x	0	2
8th	1 x	0	0	1 x	2
9th	1 x	0	1 x	0	2
10th	0	4 x	0	1 x	2
11th	1 x	1 x	1 x	1	3
12th	0	1 x	0	3 x	2
number of papers per member ([3,7])	7	6	5	7	

Table 5: Case 2 of 4 members and 12 papers.

There are some changes in the solution regarding the papers that the 1st and the 4th member will examine.

5.2.3 Third example

This example is the same with the previous one with a change this time in file data.txt. So, we will only describe files data.txt, lpfile.lp, infeasibility.txt and the results.

5.2.3.1 data.txt

4

12

1

1

1

4

4

2

1

12

3

2

In this example, we have 4 members, 12 papers and one constraint per member and per paper. Now, all members from 1 to 4 may examine [2,4] papers and all papers from 1 to 12 may be examined by [2,3] members. This change only affects the lp-formulation of the problem which follows.

5.2.3.2 lpfile.lp

The problem in lp-format is:

```
Maximize
 + 1 x1 + 0 x2 + 0 x3 + 1 x4 + 1 x5 + 1 x6 + 2 x7 + 1 x8 + 1 x9 + 0 x10 + 1 x11 + 0 x12 + 1 x13 + 0
x14 + 1 x15 + 0 x16 + 4 x17 + 0 x18 + 0 x19 + 0 x20 + 0 x21 + 4 x22 + 1 x23 + 1 x24 + 0 x25 + 1 x26
+ 0 x27 + 0 x28 + 4 x29 + 0 x30 + 1 x31 + 0 x32 + 1 x33 + 0 x34 + 1 x35 + 0 x36 + 1 x37 + 0 x38 + 1
x39 + 1 x40 + 4 x41 + 1 x42 + 0 x43 + 1 x44 + 0 x45 + 1 x46 + 1 x47 + 3 x48
Subject to
 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 <= 4
 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 >= 2
 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 <= 4
 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 >= 2
 + x25 + x26 + x27 + x28 + x29 + x30 + x31 + x32 + x33 + x34 + x35 + x36 <= 4
 + x25 + x26 + x27 + x28 + x29 + x30 + x31 + x32 + x33 + x34 + x35 + x36 >= 2
 + x37 + x38 + x39 + x40 + x41 + x42 + x43 + x44 + x45 + x46 + x47 + x48 <= 4
 + x37 + x38 + x39 + x40 + x41 + x42 + x43 + x44 + x45 + x46 + x47 + x48 >= 2
 + x1 + x13 + x25 + x37 <= 3
 + x1 + x13 + x25 + x37 >= 2
 + x2 + x14 + x26 + x38 <= 3
 + x2 + x14 + x26 + x38 >= 2
 + x3 + x15 + x27 + x39 <= 3
 + x3 + x15 + x27 + x39 >= 2
```

```
 + x4 + x16 + x28 + x40 <= 3
 + x4 + x16 + x28 + x40 >= 2
 + x5 + x17 + x29 + x41 <= 3
 + x5 + x17 + x29 + x41 >= 2
 + x6 + x18 + x30 + x42 <= 3
 + x6 + x18 + x30 + x42 >= 2
 + x7 + x19 + x31 + x43 <= 3
 + x7 + x19 + x31 + x43 >= 2
 + x8 + x20 + x32 + x44 <= 3
 + x8 + x20 + x32 + x44 >= 2
 + x9 + x21 + x33 + x45 <= 3
 + x9 + x21 + x33 + x45 >= 2
 + x10 + x22 + x34 + x46 <= 3
 + x10 + x22 + x34 + x46 >= 2
 + x11 + x23 + x35 + x47 <= 3
 + x11 + x23 + x35 + x47 >= 2
 + x12 + x24 + x36 + x48 <= 3
 + x12 + x24 + x36 + x48 >= 2
Bounds
x1 >= 0
x1 <= 1
x2 >= 0
x2 <= 1
x3 >= 0
x3 <= 1
x4 >= 0
x4 <= 1
x5 >= 0
x5 <= 1
x6 >= 0
x6 <= 1
x7 >= 0
x7 <= 1
x8 >= 0
x8 <= 1
x9 >= 0
x9 <= 1
x10 >= 0
x10 <= 1
x11 >= 0
x11 <= 1
x12 >= 0
x12 <= 1
x13 >= 0
x13 <= 1
x14 >= 0
x14 <= 1
x15 >= 0
x15 <= 1
x16 >= 0
x16 <= 1
x17 >= 0
x17 <= 1
x18 >= 0
x18 <= 1
x19 >= 0
x19 <= 1
x20 >= 0
x20 <= 1
x21 >= 0
x21 <= 1
x22 >= 0
x22 <= 1
x23 >= 0
```

```
x23 <= 1
x24 >= 0
x24 <= 1
x25 >= 0
x25 <= 1
x26 >= 0
x26 <= 1
x27 >= 0
x27 <= 1
x28 >= 0
x28 <= 1
x29 >= 0
x29 <= 1
x30 >= 0
x30 <= 1
x31 >= 0
x31 <= 1
x32 >= 0
x32 <= 1
x33 >= 0
x33 <= 1
x34 >= 0
x34 <= 1
x35 >= 0
x35 <= 1
x36 >= 0
x36 <= 1
x37 >= 0
x37 <= 1
x38 >= 0
x38 <= 1
x39 >= 0
x39 <= 1
x40 >= 0
x40 <= 1
x41 >= 0
x41 <= 1
x42 >= 0
x42 <= 1
x43 >= 0
x43 <= 1
x44 >= 0
x44 <= 1
x45 >= 0
x45 <= 1
x46 >= 0
x46 <= 1
x47 >= 0
x47 <= 1
x48 >= 0
x48 <= 1
END
```

5.2.3.3 infeasibility.txt

This example turns out to be infeasible. The message in this file is: "Inappropriate constraints. Either the maximum papers per member or the minimum members per paper should be changed.".

The inequality checking the constraints is m*max_ppm < n*min_mpp. Since m=4, n=12, max_ppm=4 and min_mpp=2, it is true, so constraints are not suitable. The user may alter them and rerun the procedure.

5.2.3.4 results

Indeed, running this problem to Soplex, we informed that it is infeasible.

5.3 Three members and four papers

We will show this example just to see the other two cases of infeasibility that can be detected.

We will describe each file separately.

5.3.1 memtopic.txt

This file contains:

1 1 1

2 1 1

3 2 1 2

The 1st member is interested in one topic numbered with 1. The 2nd member is interested in one topic numbered with 1. The 3rd one is interested in two topics numbered by 1 and 2 respectively.

5.3.2 toppaper.txt

This file contains:

1 3 1 3 4

2 2 2 3

The 1st topic has 3 papers numbered with 1, 3 and 4 respectively. The 2nd one has 2 papers numbered with 2 and 3 respectively.

5.3.3 vari.txt

It is an intermediate file and contains:

1 1 1

1 2 0

1 3 1

1 4 1

2 1 1

2 2 0

2 3 1

2 4 1

3 1 1

3 2 1

3 3 1

3 4 1

The 1st member can examine papers 1, 3 and 4 (those having edge-value 1). The 2nd one can examine papers 1, 3 and 4. The 3rd member can examine all the papers.

5.3.4 bids.txt

This file contains:

1 1 H

1 3 L

2 1 C

2 3 C

2 4 C

3 1 L

3 2 C

3 4 N

The 1st member is highly interested in paper 1 and low in 7. The 2nd member declares conflict for papers 1, 3 and 4. The 3rd one is low interested in paper 1, declares conflict for 2 and no interest for 4.

5.3.5 c.txt

This file contains:

4

0

2

1
0
0
0
0
2
0
1
1

The 1st edge joining the 1st member to the 1st paper has weight 4. The 2nd edge joining the 1st member to the 2nd paper has weight 0. And so goes on. It is not difficult to see that the second member has only zero weighted edges and no member has a non zero weighted edge with paper 2.

5.3.6 data.txt

This file contains:

3
4
1
1
1
3
4
2
1
4
3
2

We have 3 members and 4 papers. There is 1 constraint for the members and 1 for the papers. Members 1 to 3 may examine [2,4] papers. Papers 1 to 4 may be examined by [2,3] members.

5.3.7 lpfile.lp

It has the problem in lp-format:

```
Maximize
 + 4 x1 + 0 x2 + 2 x3 + 1 x4 + 0 x5 + 0 x6 + 0 x7 + 0 x8 + 2 x9 + 0 x10 + 1 x11 + 1 x12
Subject to
 + x1 + x2 + x3 + x4 <= 4
 + x1 + x2 + x3 + x4 >= 2
 + x5 + x6 + x7 + x8 <= 4
 + x5 + x6 + x7 + x8 >= 2
 + x9 + x10 + x11 + x12 <= 4
 + x9 + x10 + x11 + x12 >= 2
 + x1 + x5 + x9 <= 3
 + x1 + x5 + x9 >= 2
 + x2 + x6 + x10 <= 3
 + x2 + x6 + x10 >= 2
 + x3 + x7 + x11 <= 3
 + x3 + x7 + x11 >= 2
 + x4 + x8 + x12 <= 3
 + x4 + x8 + x12 >= 2
Bounds
x1 >= 0
x1 <= 1
x2 >= 0
x2 <= 1
x3 >= 0
x3 <= 1
x4 >= 0
x4 <= 1
x5 >= 0
x5 <= 1
x6 >= 0
x6 <= 1
x7 >= 0
x7 <= 1
x8 >= 0
x8 <= 1
x9 >= 0
x9 <= 1
x10 >= 0
x10 <= 1
x11 >= 0
x11 <= 1
x12 >= 0
x12 <= 1
END
```

5.3.8 infeasibility.txt

As it was expected looking file c.txt, this file contains the messages:

```
Member 2 has no papers to examine.
There are no members to examine paper 2.
```

5.3.9 results

Indeed, running this example in Soplex returns infeasibility message.

5.4 Execution time

In the following table, we can see the execution time of problems of various sizes using Soplex. As size of the problem we consider the number of members and papers. The time is measured in seconds. We see that increase in size results in increase in time.

Size of LP (members, papers)	Execution time in Soplex (sec)
(3, 6)	0.016 sec
(4, 12)	0.018 sec
(10, 50)	0.062 sec
(10, 85)	0.067 sec
(10, 100)	0.078 sec
(25, 120)	0.251 sec
(100, 900)	402.789 sec (~6 min 43 sec)

Table 6: Execution time of LPs in Soplex

However, improved measurements can be obtained by using more sophisticated simplex implementations, such as the commercial CPLEX [5].

Chapter 6

Conclusions

In conferences, papers are submitted to be examined by committee members. It is of high importance that justice rules be applied through the examination, such us fair distribution of the work load to the examiners and equivalent evaluation of the papers. This thesis accomplishes this assignment through an automated procedure by reducing the problem to a special case of integer linear programming.

Results show that given the members' and the papers' data, the solution is the set of edges with the greatest possible weights that satisfies the constraints and that is the best possible assignment of the papers to the members given the current data. But there might be cases where the linear program is infeasible. Three of the cases are detected and relevant messages are given as result. Such cases are that of a member being interested in only one topic with no submitted papers or of a paper submitted in a topic that is not within the interest of any member, and, finally, the case where the constraints are not well defined and have to be changed. The user, considering the infeasibility message(s), may make the appropriate changes and rerun the procedure to obtain a feasible solution.

It would be extremely useful, when simplex is used to solve a linear program which turns out to be infeasible, to know where and why simplex has stopped. In this way, the user may change the problem in the specific point to obtain a feasible solution. Finally, it would be of high interest to examine the second, third and other solution of simplex, since there is a possibility one or more of them to be better than the first one.

Bibliography

[1] "Linear programming: Foundation and Extensions, Second Edition", Robert J. Vanderbei

[2] "Theory of Linear and Integer Programming", Alexander Schrijver

[3] "Paralleler und Objekt-orientierter Simplex-Algorithmus", Roland Wunderling, Ph.D.thesis TR96-09, ZIB technical report, Berling 1996.

[4] «Θέματα σχεδίασης αλγορίθμων: Ευεπίλυτα & Δυσεπίλυτα προβλήματα», Γεώργιος Φρ. Γεωργακόπουλος, Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών, Μεταπτυχιακό πρόγραμμα σπουδών, Σημειώσεις μαθήματος HY580.

[5] CPLEX OPTIMIZATION INC. CPLEX linear optimizer 4.0.8. Incline Village, NV, USA, 1995.

[6] "The Average number of pivot steps required by the Simplex-Method is polynomial", K.H.Borgwardt, Mathematical Methods of Operations Research 26(1982), 157-177.

Appendix

ZIB ACADEMIC LICENSE

This license for ZIB software is designed to guarantee freedom to share and change software for academic use, but restricting commercial firms from exploiting your know-how for their benefit. The precise terms and conditions for using, copying, distribution, and modification follow.

Terms and Conditions for Using, Copying, Distribution, and Modification

The "Program" below refers to source, object and executable code, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. Each licensee is addressed as "you".

1. This license applies to you only if you are a member of a noncommercial and academic institution, e.g., a university. The license expires as soon as you are no longer a member of this institution.

2. Every publication and presentation for which work based on the Program or its output has been used must contain an appropriate citation and aknowledgement of the author(s) of the Program.

3. You may copy and distribute the Program or work based on the Program in source, object, or executable form provided that you also meet all of the following conditions:

    a. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge under the terms of this License. You must accompany it with this unmodified license text.

    These requirements apply to the Program or work based on the Program as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, this License does not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole and, thus, to each and every part regardless of who wrote it.

    b. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

    c. You must keep track of access to the Program (e.g., similar to the registration procedure at ZIB).

    d. You must accompany it with the complete corresponding machine-readable source code.

    The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on

which the executable runs, unless that component itself accompanies the executable.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute the Program is void and will automatically terminate your rights under this License. However, parties who have received copies or rights from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to use, modify, or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by using, modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipient's exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement, or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License.

8. If you wish to incorporate parts of the Program into other programs whose distribution conditions are different, write to ZIB to ask for permission.

NO WARRANTY

9. Because the program is licensed free of charge, there is no warranty for the program to the extent permitted by applicable law. The copyright holders provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair, or correction.

10. In no event will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.