Basic Research in Computer Science

BRICS LS-96-5 D. P. Dubhashi: What Can't You Do With LP?

What Can't You Do With LP?

Devdatt P. Dubhashi

BRICS Lecture Series **LS-96-5**

ISSN 1395-2048 **December 1996**

See back inner page for a list of recent publications in the BRICS Lecture Series. Copies may be obtained by contacting:

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through World Wide Web and anonymous FTP:

`http://www.brics.dk/`
`ftp://ftp.brics.dk/`
This document in subdirectory `LS/96/5/`

What Can't You Do With LP?

Devdatt P. Dubhashi

Devdatt P. Dubhashi
BRICS[1]
Department of Computer Science
University of Aarhus
Ny Munkegade
DK-8000 Aarhus C, Denmark

[1]**B**asic **R**esearch **I**n **C**omputer **S**cience,
Centre of the Danish National Research Foundation.

Preface

These notes from the BRICS course "Pearls of Theory" are an introduction to Linear Programming and its use in solving problems in Combinatorics and in the design and analysis of algorithms for combinatorial problems.

Devdatt P. Dubhashi
February, 1996

Contents

1 Dr. Cheng's Diet and LP **1**

2 LP versus NP: A Panacea? **7**

3 Duality **8**

4 Linear versus Integer **10**

5 Luck: Unimodularity **11**

6 Rounding **12**

7 Randomised Rounding **14**

8 Primal–Dual **16**

9 If You Want to Prospect for more Pearls ... **19**

10 Problems **19**

1 Dr. Cheng's Diet and LP

In anticipation of the lazy afternoons at the beaches in the coming summer, Dr. Cheng is prescribing the **Magnificent Body Diet**. He reckons one needs energy (2,000 kcal), protein (55g) and calcium (800 mg) every day. According to taste and dietary restrictions[2] of the participants in his programme, Dr. Cheng has narrowed down the following food items for consumption (listed according to the values corresponding to one serving):

Food	Energy(kcal)	Protein(g)	Calcium(mg)	Price (Dkr)
Gyldenmix	110	4	2	3
Eggs	160	13	54	13
Milk	160	8	285	9
Cherry Pie	420	4	22	20
Soyabean	260	14	80	19

There are of course some limits on how much of each kind one can consume per day:

Food	Max servings
Gyldenmix	4
Eggs	2
Milk	8
Cherry Pie	2
Soyabean	2

The problem is: how does one select a diet to meet all these requirements? Being a professional mathematician, Dr. Cheng abhores trial and error; instead he proposes the following systematic approach. A general diet consists of x_1 servings of Gyldenmix, x_2 of eggs, x_3 of milk, x_4 of cherry pie and x_5 of soyabean, then we can list the constraints on the diet systematically:

[2] One of us is vegetarian.

- The limits on servings:

$$\begin{aligned} 0 \leq x_1 &\leq 4 \\ 0 \leq x_2 &\leq 2 \\ 0 \leq x_3 &\leq 8 \\ 0 \leq x_4 &\leq 2 \\ 0 \leq x_5 &\leq 2 \end{aligned}$$

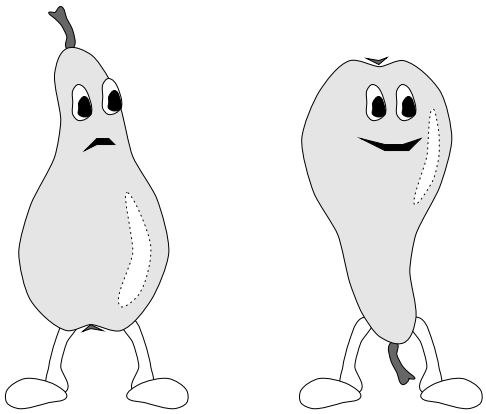
- The requirements for energy, proteins and calcium:

$$\begin{aligned} 110x_1 + 160x_2 + 160x_3 + 420x_4 + 260x_5 &\geq 2000 \\ 4x_1 + 13x_2 + 8x_3 + 4x_4 + 14x_5 &\geq 55 \\ 2x_1 + 54x_2 + 285x_3 + 22x_4 + 80x_5 &\geq 800 \end{aligned}$$

The cost of the diet would be (in Dkr):

$$3x_1 + 13x_2 + 9x_3 + 20x_4 + 19x_5.$$

Dr. Cheng's problem is to minimise this cost subject to the constraints listed above: a prototypical *linear programming* (LP) problem.



Before After

Dr. Schwartzbach and the Magnificent Body Diet

A general form of the linear programming problem for *minimisation* is the following: for positive integers m and n,

$$\begin{aligned} &\min \sum_{j\in[m]} c(j)y(j) \\ &\text{subject to} \\ &\sum_{j\in[m]} a(i,j)y(j) \geq b(i), \quad i \in [n] \\ &y(j) \geq 0, j \in [m]. \end{aligned} \tag{1}$$

or, more succinctly, in matrix notation [3],

$$\min\{cy \mid Ay \geq b, y \geq 0\}. \tag{2}$$

Symmetrically, a general *maximisation* LP problem can be framed as:

$$\begin{aligned} &\max \sum_{j\in[m]} c(j)x(j) \\ &\text{subject to} \\ &\sum_{j\in[m]} a(i,j)x(j) \leq b(i), \quad i \in [n] \\ &x(j) \geq 0, j \in [m]. \end{aligned} \tag{3}$$

or, more succinctly, in matrix notation

$$\max\{cx \mid Ax \leq b, x \geq 0\} \tag{4}$$

In Problem 10.1, you are asked to show how one can convert seemingly different formulations into this form.

[3]To try and keep notational fuss to a minimum, we shall write vectors and matrices using normal letters, and interchangeably write vectors as columns and rows, leaving the context to resolve ambiguities. Thus in writing matrix–vector products, if A is a $n \times m$ matrix, the notation Ax will imply by coherence, that x is a m–column vector while in the product yA, y is a n–row vector.

Exercise 1.1 *Why is it not particularly interesting to consider the following kind of problem:*

$$\max\{cx \mid Ax \geq b\},$$

or

$$\min\{cy \mid Ay \leq b\},$$

with A a non–negative matrix?

The good news is that linear programming problems can be solved efficiently. Beware however! For this statement means different things to different people! For the practitioner, there is the celebrated *Simplex* algorithm of George Dantzig which has been implemented and used for many years now. The theoretician however, regards the simplex algorithm as inefficient since it is known to take an exponential amount of time albeit on some isolated pathological instances. For him, there is the so–called *Ellipsoid* method developed by the Soviet school of mathematicians or Karmarkar's algorithm, which run provably in polynomial time. The perversity of the situation is completed by the fact that the practitioner would balk at implementing these latter algorithms regarding *them* as the truly inefficient ones!

We shall come away with the optimist's view: *Linear Programming problems are solvable efficiently in theory and practice.*

Let us quickly introduce some typical LP verbiage. In (3) or (1), the linear function $\sum_{j\in[m]} c(j)x(j)$ that is maximised or minimised is called the *objective function*. The inequalities $\sum_{j\in[m]} a(i,j)x(j) \leq (\geq) b(i), i \in [n]$ are called the LP*constraints*. Any vector x satisfying these constraints is called a *feasible solution*. Thus the LP problem is to maximise or minimise the objective function over the space of feasible solutions.

It is often helpful to think of the LP problem geometrically. The feasible solution space is then seen to be region of euclidean space carved out by the constraint inequalities. Such a set is called a *polytope*. The polytope corresponding to the inequalities $Ax \leq b$ is a *convex* set: i.e. if two points are in it, then so is the entire line between them. One can easily verify this fact algebraically: if $Ay \leq b$ and $Az \leq b$, then also $Ax \leq b$ for any $x = \lambda y + (1 - \lambda z)$ with $0 \leq \lambda \leq 1$, It is a general fact of optimisation over

convex sets that the optimum (maximum or minimum) is always attained at an extreme or corner point. Thus the corner points of the polytope $Ax \leq b$ are of special interest; in fact the simplex algorithm proceeds from one corner to a neighbouring one until it discovers the optimum.

We conclude this introduction to LP with some historical comments. A diet problem involving 77 different foods, to meet the minimum requirements of nine different nutrients at minimum cost was published by G.J. Stigler in 1945. The formulation was essentially a LP in 77 non–negative variables with 9 inequality constraints. Stigler was unaware of LP technology and obtained an approximate solution by trial and error leading to a diet that met the minimum requirements at an annual cost of $ 39.93 in 1939 prices. After George E. Dantzig developed the Simplex algorithm in 1947, Stigler's diet problem was one of the first large problems to be solved by the Simplex method and it gave the true optimum diet with an annual cost of $39.67 at the 1939 prices. Stigler's trial and error wasn't too far off after all!

Dantzig was employed at that time in the US Air Force Headquarters as a mathematician with the responsibility for devising improved methods for planning air force activities. Although Dantzig is generally credited (at least in the West) with the development of the theory of linear programming and the Simplex algorithm for solving LP problems, there were at least two precursors. During World war II, the Dutch mathematician Tjalling C. Koopmans actually developed the essential priciples of linear programming in a special case. later Koopmans went on to successfully apply linear programming priciples to the fundamental problems of economics as embodied in various models such as the famous one proposed by L. Walras in 1874. In the Soviet Union, Leonid V. Kantorovic working for a Soviet production enterprise made the discovery independently. In his mongraph published in 1939, he introduced the basic LP formulation, developed the general theory using methods very close to the ones used even today and outlined a method for solving LP problems on desk calculators (this was the pre–computer era). In view of all this, the continued attribution of the discovery of linear programming to Dantzig alone is explicable only on the grounds that Kantorovich couched his formulation in terms of maximising production while Dantzig phrased his in terms of maximising profit! However, recognition was accorded when on October 14, 1975, the Royal Sweedish Academy of Sciences

awarded the Nobel Prize in economics to L.V. Kantorovich and T.C. Koopmans "for their contributions to the optimal allocation of resources." This time, in a perverse compounding of errors, Dantzig was excluded because his work was considered "too mathematical"!

Another interesting anecdote from the history of LP is related by Dantzig describing his first encounter with the famous Hungarian–American mathematician John von Neumann:

> On October 1, 1947, I visited von Neumann for the first time at ... Princeton. I remember trying to describe .. the Air force problem. I began with the formulation of the linear programming model in terms of activities and items etc. ... "Get to the point", he said impatiently ... I said to myself: "OK, if he wants a quick version of the problem, then that's what he'll get."In under a minute I slapped the geometric and algebraic version of the problem on the board. Von Neumann stood up and said: "Oh that!". He then proceeded for the next hour and a half to lecture to me on the mathematical theory of linear programs ... At one point, seeing me sitting there with my eyes popping and my mouth open (after all, I had searched the literature and found nothing), von Neumann said: "I don't want you to think that I am generating all this on the spur of the moment. I have just recently completed a book with Oscar Morgenstern on the Theory of Games. What I am doing is conjecturing that the two are equivalent. The theory I am outlining to you is really an analogue of the one we have developed for the Theory of Games." Thus I learned about Farkas' Lemma and duality for the first time.

For a nice ccount of the discovery of Linear programming, see the article of that name by Dorfman [2].

2 LP versus NP: A Panacea?

What else can one do with LP besides prescribing diets? Let us take the following *maximum satisfiability* problem which is known to be a tough nut to crack (it is NP–hard). Given:

- a set of boolean variables $x_1, \ldots, x_m$;
- a CNF–formula over them i.e. a set of clauses $C_1, \ldots, C_n$ each in *conjunctive* normal form (each clause has the form $(y_1 \vee y_2 \vee \cdots \vee y_k)$ where each y_i is either a variable or a negated variable) and
- a non–negative weight $w(C)$ for each clause C.

The problem is to find an *assignment* of 0/1 values to the variables so as to maximise the total weight of the satisfied clauses. A clause is satisfied by a 0/1 assignment of the variables if on substituting the assigned values to the variables, it evaluates to 1. That is, if at least one of the variables in the set C^+ of *positively occuring literals* is assigned 1 or at least one of the variables in the set C^- of negatively occuring literals is assigned 0.

Let us introduce 0/1 variables $z(C)$ for each clause C with the interpretation that $z(C) = 1$ iff clause C is satisfied; then one can formulate this problem to look like the LP problem 3:

$$\max \textstyle\sum_C w(C) z(C) \tag{5}$$

subject to

$$\textstyle\sum_{x_j \in C^+} x_j + \sum_{x_j \in C^-} (1 - x_j) \geq z(C), \quad \text{for each clause } C, \tag{6}$$

$$x_j, z(C) \in \{0, 1\}. \tag{7}$$

Exercise 2.1 *Convince yourself that a feasible solution to the above program corresponds to an assignment and a set of satisfied clauses and hence that the solution to the above program is indeed the maximum weight of satisfied clauses under any assignment.*

For a second example, consider the *minimum–weight vertex cover problem* from graph theory: given a graph $G := (V, E)$ and non–negative weights $w(v), v \in V$, find a *vertex cover* of minimum weight. A vertex cover is a set of vertices $S \subseteq V$ such that for every edge $(u, v) \in E$, we have $u \in S$ or $v \in S$ i,e the edge e is *covered* by S. The weight of a cover S is defined naturally by $w(S) := \sum_{u \in S} w(u)$. The minimum vertex cover problem is known to be a NP–hard problem.

Let us introduce variables $x(v), v \in V$ so that each variable takes only the values 0 and 1 with the interpretation that $x(v) = 1$ iff $v \in S$. Then one can formulate the minimum vertex cover problem as the following program:

$$\min \sum_{v \in V} w(v)x(v) \tag{8}$$

subject to

$$x(u) + x(v) \geq 1, \quad \text{for each } (u, v) \in E, \tag{9}$$

$$x(v) \in \{0, 1\}, \quad v \in V. \tag{10}$$

These two examples illustrate the

Pearl 1 *Many combinatorial optimisation problems can be formulated as linear programming problems.*

Exercise 2.2 *Why is this not the end of the story? Given the polynomial time algorithms for the LP problem mentioned in § 1, why does this not imply that P=NP?*

3 Duality

While you are puzzling over this conundrum, let us quickly develop some basic LP theory. Multiplying the constraints in (3) by non–negative multipliers $y(1), \ldots, y(n)$ (so that the inequalities are preserved) and adding, we get:

$$\sum_{j \in [m]} \left(\sum_{i \in [n]} y(i)a(i, j) \right) x(j) \leq \sum_{i \in [n]} y(i)b(i).$$

Comparing with the objective function $\sum_{j\in[m]} c(j)x(j)$, we see that provided the multipliers $y(1),\ldots,y(n)$ are chosen so that

$$\sum_{i\in[n]} y(i)a(i,j) \geq c_j, \quad j \in [m],$$

the optimum sought is bounded from above by $\sum_{i\in[n]} y(i)b(i)$. Thus we are naturally led to consider the following *dual* problem:

$$\begin{aligned} &\min \textstyle\sum_{i\in[n]} y(i)b(i) \\ \text{subject to} \quad & \\ &\textstyle\sum_{i\in[n]} y(i)a(i,j) \geq c_j, \quad j \in [m] \qquad (11) \\ &y(i) \geq 0, \quad i \in [n]. \qquad (12) \end{aligned}$$

or, more succinctly,

$$\min\{yb \mid yA \geq c, y \geq 0\}. \qquad (13)$$

The problems (3) and (11) or equivalently, (4) and (13) are said to be *duals* of each other; the first in each pair is (rather arbitrarily) called the *primal* problem and other, the *dual*. Notice that

- a variable in the dual is paired with an inequality in the primal, (and vice–versa);
- the objective function of the dual is determined by the right hand side of the primal constraints (and vice–versa);
- the constraint matrix of the dual is the transpose of the constraint matrix of the primal.

Exercise 3.1 *In a similar manner, develop the dual of a primal problem posed as a minimisation problem, (1).*

Exercise 3.2 *What is the dual of the dual?*

There is a close relation between the values of the primal and dual programs[4]. We argued above that the so–called *Weak Duality Theorem* holds:

$$\max\{cx \mid Ax \leq b, x \geq 0\} \leq \min\{yb \mid yA \geq c, y \geq 0\}.$$

In fact the values are *equal*!

Theorem 3.3 (Duality Theorem) *The values of a LP and its dual are equal; that is:*

$$\max\{cx \mid Ax \leq b, x \geq 0\} = \min\{yb \mid yA \geq c, y \geq 0\}. \tag{14}$$

Moreover, there is a curious relation between the primal and dual solutions, the so—called *complementary slackness* conditions: if x^* is an optimal primary solution and y^* an optimal dual solution, then

(primal) If $x^*(j) > 0$ then $\sum_{i\in[n]} y^*(i)a(i,j) = c_j$.

(dual) If $y^*(i) > 0$ then $\sum_{j\in[m]} a(i,j)x^*(j) = b_i$.

That is, if in an optimal solution, a variable is non–zero, then the corresponding inequality in the dual is satisfied with equality.

4 Linear versus Integer

Let us return to the conundrum posed at the end of § 2. The catch is that the programs we wrote there were in fact, *not* linear programs! There were constraints that the variables involved must be *integers*, in particular 0/1–valued. Unfortunately, *integer linear programming* (ILP), that is, the LP problem under the restriction that feasible solutions must be integral is also a NP–complete problem (even if the variables are only 0/1 valued)!

[4]We shall always assume that all LPs we write have finite optimal solutions to avoid various technicalities.

5 Luck: Unimodularity

Sometimes however, we may be lucky: the optimal solution of the linear programming might just turn out to be integral! Geometrically, the polytope $Ax \leq b$ might just turn out to have integer vertices. In particular, this can happen if the constraint matrix has a special structure.

Pearl 2 *Exploit structure in the constraint matrix of a LP.*

A matrix is said to be *totally unimodular* if each subdeterminant of the matrix is $-1, 0$ or $+1$. (In particular, each entry of a totally unimodular matrix is one of these three values.) A beautiful result of Hoffman and Kruskal is the following:

Theorem 5.1 *An integral matrix A is totally unimodular iff*

- *for all integral vectors b, the polytope $Ax \leq b$ is integral i.e. has integer vertices.*
- *for all integral vectors b and c, both sides of the linear programming duality equation (14):*

$$\max\{cx \mid Ax \leq b, x \geq 0\} = \min\{yb \mid yA \geq c, y \geq 0\}.$$

  are achieved by integral vectors x^* and y^*.

To use this pearl, we must be able to recognise when the constraint matrix is totally unimodular. We can employ the following characterisation:

Proposition 5.2 *A matrix is totally unimodular iff each collection of its rows can be split into two parts such that the sum of the rows in one part minus the sum of the rows in the second part is a vector with entries $-1, 0$ or $+1$.*

Let M be the incidence matrix of a graph $G := (V, E)$; so M is indexed by vertices on the rows and by edges on the columns with $M(v, e) = 1$ if e is incident on v and 0 otherwise. We shall use the following fact to give slick proofs of some classical results in graph theory.

Exercise 5.3 *Use Proposition 5.2 to show that M is totally unimodular iff G is bipartite.*

Let G be a bipartite graph and M its incidence matrix. By the Hoffman–Kruskal theorem 5.1, we have,

$$\max\{y1 \mid yM \leq 1, y \geq 0, \text{integral}\} = \min\{1x \mid Mx \geq 1, x \geq 0, \text{integral}\}.$$

(Here 1 stands for the all 1 vector of dimension $|V|$ on the left side and $|E|$ on the right!). This is

Theorem 5.4 (König's Covering Theorem) *The maximum cardinality of an independent set in a bipartite graph is equal to the size of a minimum edge cover (i.e. a set of edges incident on every vertex).*

Similarly, we also have:

$$\max\{1x \mid Mx \leq 1, x \geq 0, \text{integral}\} = \min\{y1 \mid yM \geq 1, y \geq 0, \text{integral}\}.$$

This is the famous

Theorem 5.5 (König–Egerváry Theorem) *The maximum cardinality of a matching in a bipartite graph is equal to the minimum cardinality of a vertex cover (i.e. a set of vertices covering every edge).*

6 Rounding

Most often, though we are not so lucky! In that case, we could still try to *relax* the integrality constraints and see what the solution of the resulting

LP can tell us. Let us consider the *LP relaxation* of the vertex cover problem from § 2:

$$\begin{aligned} & \min \textstyle\sum_{v \in V} w(v)x(v) \\ \text{subject to} & \\ & x(u) + x(v) \geq 1, \quad \text{for each } (u,v) \in E, \\ & 0 \leq x(v) \leq 1, \quad v \in V. \end{aligned} \tag{15}$$

Notice that we have *relaxed* the integrality constraint $x(v) \in \{0,1\}$ into weaker linear constraints. This enlarges our space of feasible solutions, so if Z^*_{IP} is the optimum value of the integer program (8) (and hence the true optimum value of the vertex cover problem) and Z^*_{LP} is the optimum value of the LP relaxation, then we have the relation $Z^*_{IP} \geq Z^*_{LP}$, an observation which we will use repeatedly.

Exercise 6.1 *What is the analogous relation between a maximisation ILP problem and its LP relaxation?*

Solving the LP will give us an optimal solution x^* which is in general fractional. In order to extract from this a meaningful solution to the original problem, a natural strategy that suggests itself is to *round* the fractional solution. In our example, if x^* is the optimal LP solution, one can round it to an integral solution $\bar{x}$ according to the following simple rule:

$$\bar{x}(v) := \begin{cases} 1, & \text{if } x^*(v) \geq 0.5; \\ 0, & \text{otherwise.} \end{cases}$$

This 0/1 solution will be (the characteristic vector of) the vertex cover we output.

First, observe that $\bar{x}$ is indeed a feasible solution i.e. it is the incidence vector of a vertex cover. For any edge (u,v), the corresponding constraint forces $\max(x^*(u), x^*(u)) \geq 0.5$, hence at least one of $\bar{x}(u)$ or $\bar{x}(v)$ will be rounded to 1, that is, at least one of the two endpoints will be in our cover, so our set is indeed a vertex cover. Moreover, $\bar{x}(v) \leq 2x^*(v)$ for any $v \in V$, hence

$$\sum_{v \in V} w(v)\bar{x}(v) \leq 2 \sum_{v \in V} w(v)x^*(v).$$

Finally we bring in our observation that the value of the LP solution cannot exceed the value of the true optimal vertex cover; hence we have just found a conceptually simple algorithm that *approximates* the vertex cover problem within a factor of 2. In fact no better approximation is known to be attainable in polynomial time!

The example above was an illustration of

Pearl 3 *Seek help from the* **LP relaxation** *of an integer programming problem.* **Round** *the fractional LP solution to an integral one to get a reasonable approximation to the true value.*

This algorithm is simple to describe, but requires the solution of a linear program which can be substantial task. In § 8, we shall see an algorithm that has the same approximation guarentee but obviates the need to solve a linear program, although it still comes out of LP!

7 Randomised Rounding

Now let us consider the other example from § 2: the maximum satisfiability problem which we wrote as an integer program (5). In the spirit of Pearl 3, we write down the LP relaxation immediately:

$$\max \textstyle\sum_C w(C)z(C) \tag{16}$$

subject to

$$\begin{aligned} \textstyle\sum_{x_j \in C^+} x_j + \sum_{x_j \in C^-} (1 - x_j) &\geq z(C), \quad \text{for each clause } C, \\ 0 \leq x_j, z(C) &\leq 1. \end{aligned}$$

So we solve this LP and round the solution as before, right?

Exercise 7.1 *Suppose we employ the rounding rule we had before: round a variable to* 1 *if it exceeds* 0.5 *and to* 0 *otherwise. Argue that one cannot give any good performance guarantee in this case.*

It's time for a new

Pearl 4 *Introduce* **randomness***: interpret the fractional LP solutions as* **probabilities** *to guide the rounding process.*

We propose the following *randomised* algorithm for the MAXSAT problem: solve the LP relaxation (16), and then use the (fractional) LP solution x^* as probabilties for a rounding procedure: round $x(i)$ to 1 with probabilty $x^*(i)$ and to 0 with probability $1 - x^*(i)$.

To analyse the approximation one obtains, let us estimate the probability that a particular clause C is satsified. Clause C fails to be satisfied exactly if all the literals in it are set to 0; hence C is satisfied with probability

$$\Pr(C \quad \text{is satisfied}) = 1 - \prod_{x_j \in C^+} (1 - x^*(j)) \prod_{x_j \in C^-} x^*(j).$$

Let's take a look at the clause constraint in (16). Suppose we have a set S of k elements partitioned into two subsets $S := A \cup B$, and such that

$$\sum_{a \in A} a + \sum_{b \in B} (1 - b) \geq z.$$

Notice that this is equivalent to

$$\sum_{a \in A} (1 - a) + \sum_{b \in B} b \leq k - z.$$

Now, using the arithmetic–geometric mean inequality,

$$\begin{aligned} \prod_{a \in A} (1 - a) \prod_{b \in B} b &\leq (\frac{\sum_{a \in A} (1 - a) + \sum_{b \in B} b}{k})^k \\ &\leq (1 - z/k)^k. \end{aligned}$$

Taking S to be the set of literals in our clause C with $A := C^+$ and $B := C^-$, the partition into positively and negatively occuring literals respectively, we get that if C is a clause with k literals, then

$$\Pr(C \quad \text{is satisfied}) \geq 1 - (1 - z^*(C)/k)^k.$$

Exercise 7.2 *Verify by elementary calculus or otherwise, that in the interval* $[0, 1]$*,*

$$1 - (1 - z/k)^k \geq \beta_k z,$$

where $\beta_k := 1 - (1 - 1/k)^k \geq (1 - \frac{1}{e})$*.*

Let W_{OPT} be the optimum value of the MAXSAT problem, and let Z^*_{LP} be the optimal value of the LP–relaxation (16). Thus $W_{OPT} \leq Z^*_{LP}$. The expected weight of the assignment produced, $\hat{W}$ satisfies:

$$\begin{aligned}
\hat{W} &= \sum_C w(C)\Pr(C \quad \text{is satisfied}) \\
&\geq \sum_C w(C)(1 - \frac{1}{e})z^*(C) \\
&\geq (1 - \frac{1}{e})\sum_C w(C)z^*(C) \\
&= (1 - \frac{1}{e})Z^*_{LP} \\
&\geq (1 - \frac{1}{e})W_{OPT}.
\end{aligned}$$

Thus we have a randomised algorithm that delivers an approximation that is at least a factor of $(1 - \frac{1}{e})$ of the optimal. In Problem 10.7, we investigate how to remove the randomisation and get a deterministic algorithm wih the same performance. Should you care for our Delicious Oyster of the Week, Problem 10.8, you are guided towards an improved solution.

8 Primal–Dual

The basic idea of the *primal–dual* method may be summarised in

Pearl 5 *Manipulate solutions to the primal and dual problems together, using the complementary slackness conditions to drive towards optimality.*

Recall that if we have a primal solution $\hat{x}$ and a dual solution $\hat{y}$, that together satisfy the complementary slackness conditions, then this is a certificate that both solutions are optimal. Otherwise, one can improve either the primal or dual solution and continue.

In cases of interest to us, we must modify this strategy somewhat; since we are interested in *integer* solutions, we should not blindly pursue the complementary slackness conditions to optimality of the linear relaxations. A useful strategy in such cases is to maintain only a dual feasible solution and enforce only one half of the complementary clackness conditions, namely, the primal. This will lead to a good approximate solution.

We illustrate this by returning to the vertex cover problem which we formulated as a ILP (8) and whose LP relaxation, (15) is reproduced below for convenience:

$$\begin{aligned} & \min \textstyle\sum_{v \in V} w(v)x(v) \\ \text{subject to} & \\ & x(u) + x(v) \geq 1, \quad \text{for each } (u,v) \in E, \\ & 0 \leq x(v) \leq 1, \quad v \in V. \end{aligned} \tag{17}$$

We want to consider concurrently the dual:

$$\begin{aligned} & \max \textstyle\sum_{e \in E} y(e) \\ \text{subject to} & \\ & \textstyle\sum_{e \in \delta(v)} y(e) \leq w(v), \quad v \in V, \end{aligned} \tag{18}$$

$$y(e) \geq 0, \quad e \in E. \tag{19}$$

We will only enforce the primal complementary slackness conditions:

$$x^*(v) > 0 \quad \rightarrow \quad \sum_{e \in \delta(v)} y^*(e) = w(v).$$

To start, we can let $x := 0$ and $y := 0$; note that the latter is a dual feasible solution and the primal complemtary slackness conditions hold trivially. Now, let us try to improve the dual solution: pick any edge e and increase $y(e)$ as much as possible. That is, until the node constraint $\sum_{e \in \delta(v)} y(e) \leq w(v)$

becomes *tight* for one of its endpoints. At this stage, add this *tight node* v to the vertex cover by setting $x(v) := 1$, and delete the edge e from further consideration. Note that feasibility of the dual was maintained and so were the primal complementary slackness conditions. Continue until all edges have been processed in this fashion. This is a simple linear time algorithm.

We claim that the result is a vertex cover of value at most twice the optimal. That we do produce a cover is immediate from the fact that an edge is only deleted when one of its endpoints (the tight one) is added to the cover. For the second part of the claim, imagine that when the value of a variable $y(e)$ is increased by k, we pay k to each of its endpoints. Then the total value paid to all nodes is exactly twice the value of the dual solution, which is a lower bound on the value of the optimal cover. Finally note that a vertex is only added to the cover when it is "fully paid for", that is it is paid $w(v)$.

If this was too slick a sleigh of hand, here is a more careful verification that brings out exactly what linear–programming principles are being used. Let VC_{OPT} denote the value of an optimal vertex cover, and let Z^* be the optimal value of the primal LP problem and $\hat{Z}$ the value of any feasible dual solution. By the nature of a relaxation, $VC_{OPT} \geq Z$ and by the Weak Duality Theorem, $Z^* \geq \hat{Z}$. Hence we conclude that $VC_{OPT} \geq \hat{Z}$. In particular, with our specific dual feasible solution from the algorithm, $\hat{Z} = \sum_e y(e)$ and we have

$$\begin{aligned} VC_{OPT} &\geq \sum_{e \in E} y(e) \\ &= \frac{1}{2} \sum_{v \in V} \sum_{e \in \delta(v)} y(e) \\ &\geq \frac{1}{2} \sum_{v \in V, x(v) > 0} \sum_{e \in \delta(v)} y(e) \\ &= \frac{1}{2} \sum_{v \in V, x(v) > 0} w(v), \end{aligned}$$

where in the last line, we employed the fact that the primal complemantary slackness conditions were enforced. Thus the value of the solution produced saisfies $\sum_{v \in V, x(v) > 0} w(v) \leq 2VC_{OPT}$, as claimed.

9 If You Want to Prospect for more Pearls ...

The best introduction to linear programming is the elegant book of Chvátal [1] in which you can find many other applications of linear programming. A more advanced and comprehensive book for the theorist is that of Schrijver [3]. A nice, insightful survey of the utility of linear programming related techniques for approximation algorithms is given by Shmoys [4]. Our examples from § 6 through § 8 are discussed there together with many other examples, applications and references. A detailed discussion of randomised rounding in the context of packing and covering type of LP problems is given in [5].

10 Problems

Problem 10.1 *Show how to bring the following LP into the standard form in § 1:*

$$\begin{aligned} \max \quad & 3x_1 + 5x_2 - 7x_3 \\ \textit{subject to} \quad & x_1 + x_2 + 3x_4 = 5 \\ & x_1 + x_2 - x_3 \leq 6 \\ & x_1 \geq 0. \end{aligned}$$

Problem 10.2 *Apply the duality equation (14) to derive the following variant:*

$$\max\{cx \mid Ax \leq b\} = \min\{yb \mid yA = c, y \geq 0\}.$$

Problem 10.3 *Generalise the reasoning in § 3 to prove a Weak Duality Theorem that is suggestive of the following Duality Theorem for integer linear programming. Call a function $F : R^n \to R$* **super–additive** *if $F(u+v) \geq F(u) + F(v)$ and* **non–decreasing** *if $u \leq v$ implies $F(u) \leq F(v)$. Let $\mathcal{F}$ denote the class of super–additive, non–decreasing functions. Then*

$$\max\{cx \mid Ax \leq b, x \geq 0, \textit{integral}\} = \min\{F(b) \mid F(a(j)) \geq c_j, F \in \mathcal{F}\}.$$

Problem 10.4 *Use the complementary slackness conditions to set up and solve a system of linear equations to check if the alleged optimal solution*

$$x_1^* = 2, x_2^* = 4, x_3^* = 0, x_4^* = 0, x_5^* = 7, x_6^* = 0,$$

to the LP problem

$$\begin{array}{rc} \max & 18x_1 - 7x_2 + 12x_3 + 5x_4 + 8x_6 \\ \textit{subject to} & 2x_1 - 6x_2 + 2x_3 + 7x_4 + 3x_5 + 8x_6 \leq 1 \\ & -3x_1 - x_2 + 4x_3 - 3x_4 + x_5 + 2x_6 \leq -2 \\ & 8x_1 - 3x_2 + 5x_3 - 2x_4 + 2x_6 \leq 4 \\ & 4x_1 + 8x_3 + 7x_4 - x_5 + 3x_6 \leq 1 \\ & 5x_1 + 2x_2 - 3x_3 + 6x_4 - 2x_5 - x_6 \leq 5 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0. \end{array}$$

is indeed so.

Problem 10.5 (Oyster of the Week) *A* **permutation matrix** *is a square* $0/1$ *matrix with exactly one* 1 *in each column and row. A* **doubly–stochastic matrix** *is a square matrix with non–negative entries with each row and column sum equal to* 1*. (Thus, in particular, a permutation matrix is also doubly–stochastic.) By formulating a LP with a totally unimodular constraint matrix, establish the following famous theorem of Birkhoff and von Neumann:*

Theorem 10.1 (Birkhoff–von Neumann) *Every doubly stochastic matrix can be written as a convex combination of permutation matrices.*

That is, for every doubly stochastic $n \times n$ *matrix* D*, there exist non–negative reals* $\lambda_1, \ldots \lambda_{n!}$ *summing to* 1 *such that* $D = \sum_i \lambda_i P_i$*, where* $P_1, \ldots, P_{n!}$ *are all the permutation matrices.*

Problem 10.6 (Oyster of the Week: Alternate Menu) *A subset of vertices in a graph $G := (V, E)$ is said to be* **independent** *or* **stable**, *if no two vertices in the subset are adjacent. Let us assign real numbers $x(u), u \in V$ subject to the conditions:*

$$x(u) + x(v) \leq 1, \quad (u, v) \in E,$$

and $x(u) \geq 0, u \in V$.

1. *Argue that integer solutions to this set of inequalities correspond exactly to stable sets in the graph.*

2. *Deduce that there is a polynomial time algorithm for finding the maximum size of a stable set in a bipartite graph. (Note that this problem is NP–complete for general graphs.)*

Problem 10.7 *In this problem, we will outline a general technique for* **derandomisation**, *that is, for coverting a randomised algorithm into a deterministic one (with the same performance guarantees).*

1. *Consider the following simple randomised algorithm for the MAX–SAT problem: set each variable x_i independently to 1 or 0 equiprobably. By analysing the probability that a fixed clause is satisfied, show that for the special case of MAX–3SAT, namely where each clause has at most 3 literals, this gives a 7/8–approximation algorithm.*

2. *Now we outline a scheme to set the values of the variables $x_1, x_2, \ldots$ sequentially and deterministically. When setting the value of x_i, we will consider a 3–SAT formula ϕ_i involving only the variables $x_i, \ldots, x_n$; initially $i = 1$ and $\phi_1 := \phi$ the input formula. Let $W(\phi_i)$ be the expected weight of satisfied clauses if we set the variables $x_i, \ldots, x_n$ independently and equiprobably to 0 or 1. Let ϕ_i^1 be obtained from ϕ_i as follows: if x_i occurs positively in a clause C, replace the clause by 1 (which is always satisfied) and if it occurs negatively in C, delete x_i from the clause C. Here is the rule: if $W(\phi_i^1) \geq W(\phi_i^0)$, then set $x_i := 1$ and $\phi^{i+1} := \phi_i^1$, else $x_i := 0$ and $\phi_{i+1} := \phi_i^0$. Argue that this is indeed a deterministic scheme to set the variables.*

3. *Argue that* $W(\phi_i) \leq W(\phi_{i+1})$ *for each* $1 \leq i < n$. *Hence dedeuce that our deterministic scheme outputs an assignment with weight at least* $7/8$ *times the optimal: we have derandomised the algorithm from the first part.*

4. *Why is this called* **the method of conditional probabilities***?*

5. *Apply the method of conditional probabilities to derandomise the algorithm from § 7.*

Problem 10.8 (Delicious Oyster of the Week) *In this problem, we will outline an improvement to the randomised rounding procedure used in § 7.*

1. *For* $0 < \alpha \leq 1$, *say that a function* $f : [0,1] \to [0,1]$ *has the* α*–property, if for all* $y_1, \ldots, y_k \in [0,1]$ *and for any partition* $[k] = I \cup J$,

$$1 - \prod_{i \in I} f(y_i) \prod_{j \in J} f(1 - y_j) \geq \alpha \min(1, \sum_{i \in I} (1 - y_i) + \sum_{j \in J} y_j).$$

   Given a function f *with the* α *property, employ the following rounding rule: instead of using the LP solution* x^* *directly to do the rounding, apply* f *and round* x_i *to* 1 *with probability* $f(x^*(i))$ *and to* 0 *with probability* $1 - f(x^*(i))$. *Show that the resulting algorithm delivers a solution of value at least* α *times the optimal.*

2. *Show that the following functions have the* $3/4$ *property: any function* f *satisfying* $1 - 4^{-y} \leq f(y) \leq 4^{y-1}$ *or the linear function* $f_\gamma(y) := \gamma + (1 - 2\gamma)y$, *for* $2 - \frac{3}{4^{1/3}} \leq \gamma \leq \frac{1}{4}$. *Deduce that there is a* $3/4$ *approximation algorithm for MAXSAT.*

3. *Show that there are no functions with the* α*–property for* $\alpha > 3/4$.

4. *Derandomise the algorithm from above to obtain a deterministic* $3/4$*–approximation algorithm.*

Problem 10.9 (Delicious Oyster of the Week: Alternate Menu) *The* **Hitting Set Problem** *is as follows: given a ground set E, with non–negative costs $c(e), e \in E$, and subsets $T_1, \ldots, T_n \subseteq E$, find a subset $A \subseteq E$ of minimum cost that* **hits** *all the given sets, i.e. $A \cap T_i \neq \emptyset$ for each $i \in [n]$.*

1. *Formulate the vertex cover problem as a special case of the hitting set problem. Can you find other well known problems that are special cases of the hitting set problem?*

2. *Develop a primal–dual algorithm for the hitting set problem and analyse its performance.*

References

[1] V. Chvatál, *Linear Programming*, W.H. Freeman and co. 1980.

[2] R. Dorfman, "The Discovery of Linear Programming", *Annals of Computing*, 6(3) pp. 283–295, July 1984.

[3] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley and sons, 1986.

[4] D. Shmoys, " Computing Near–Optimal Solutions to Combinatorial Optimization Problems", in W. Cook, L. Lovasz and P. Seymour (eds.): *Combinatorial Optimization*, Papers from the DIMACS Special Year, DIMACS series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1995.

[5] A. Srinivasan, *The Role of Randomness in Computation*, BRICS Notes Series.

Recent Publications in the BRICS Lecture Series

LS-96-5 **Devdatt P. Dubhashi.** ***What Can't You Do With LP?*** **December 1996. viii+23 pp.**

LS-96-4 **Sven Skyum.** ***A Non-Linear Lower Bound for Monotone Circuit Size*****. December 1996. viii+14 pp.**

LS-96-3 **Kristoffer H. Rose.** ***Explicit Substitution – Tutorial & Survey*****. September 1996. v+150 pp.**

LS-96-2 **Susanne Albers.** ***Competitive Online Algorithms*****. September 1996. iix+57 pp.**

LS-96-1 **Lars Arge.** ***External-Memory Algorithms with Applications in Geographic Information Systems*****. September 1996. iix+53 pp.**

LS-95-5 **Devdatt P. Dubhashi.** ***Complexity of Logical Theories*****. September 1995. x+46 pp.**

LS-95-4 **Dany Breslauer and Devdatt P. Dubhashi.** ***Combinatorics for Computer Scientists*****. August 1995. viii+184 pp.**

LS-95-3 **Michael I. Schwartzbach.** ***Polymorphic Type Inference*****. June 1995. viii+24 pp.**

LS-95-2 **Sven Skyum.** ***Introduction to Parallel Algorithms*****. June 1995. viii+17 pp. Second Edition.**

LS-95-1 **Jaap van Oosten.** ***Basic Category Theory*****. January 1995. vi+75 pp.**