Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Operations Research Letters 32 (2004) 49–58

Operations Research Letters

www.elsevier.com/locate/dsw

Monotonizing linear programs with up to two nonzeroes per column[☆]

Dorit S. Hochbaum

Department of Industrial Engineering and Operations Research, Walter A. Haas School of Business, University of California, Berkeley, CA 94720, USA

Received 5 March 2003; received in revised form 2 May 2003; accepted 5 May 2003

Abstract

Linear programming problems with up to two nonzeroes per column in the constraint matrix are shown equivalent to generalized network flow problem. The transformation is applied for solving the maximum cut problem, the b-matching problem in strongly polynomial time and for approximation algorithms for certain integer versions of the problem.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Half integral solutions; Generalize flow; Monotone constraints; Approximation algorithms

1. Introduction

We consider here a general linear programming problem on m variables and n constraints where each column of the $n \times m$ matrix $\mathbf{A}$ has at most two integer nonzeroes. We call such problems LP2.

$$\text{(LP2)} \quad \begin{aligned} &\text{Min} && \mathbf{cx} \\ &\text{s.t.} && \mathbf{Ax} = \mathbf{b} \\ & && \mathbf{x} \geqslant 0. \end{aligned}$$

The nonnegativity constraints can be replaced by general upper and lower bound constraints, $\ell_i \leqslant x_i \leqslant u_i$ for all i, without affecting the results reported here. Let $U = \max_{i=1,\dots,m}\{u_i - \ell_i\}$.

When the two nonzeroes in each column of the matrix $\mathbf{A}$ are of opposite signs the matrix is called *pre-Leontief*. We call such matrix *column-monotone* to stress the analogy to the monotone rows (or inequalities) concept. The procedure presented here converts a nonmonotone LP2 into a column-monotone LP2. This procedure is a "dual" analogue of the respective procedure for linear and integer programming problems on monotone constraints.

The interest in monotonizing a matrix arises from properties of monotone systems that allow to solve the respective optimization problem more efficiently than the nonmonotone system. Consider a matrix that is *row-monotone*, that is, each inequality in the constraint matrix is monotone. An inequality in two variables is called *monotone* if it is of

[☆] This is to gratefully acknowledge the support of this research by NSF awards Nos DMI-0085690 and DMI-0084857. I wish to extend my thanks to Klaus Truemper for his encouragement and help on linking the results here to the maximum cut application. Without his prodding this paper would not have been written. I am also indebted to Giovanni Rinaldi for providing the details on the maximum cut research project and the use of the monotonization technique described here.

E-mail address: hochbaum@ieor.berkeley.edu (D.S. Hochbaum).

the form

$$ax_{j_i} - bx_{k_i} \geqslant c,$$

where a and b are both nonnegative.

Solving integer programming problems constrained on a system of monotone inequalities is easier than solving a general integer programming. Although, as proved by Lagarias [15], the problem of finding a *feasible* solution of a system of monotone inequalities in integers is NP-complete, the algorithm of Hochbaum and Naor [14] finds an *optimal* solution for integer programming optimization in n variables over m monotone constraints in time $O(mnU^2 \log(Un^2/m))$ where U is the largest range of a variable. This proves that the feasibility problem is *weakly* NP-complete—in unary representation it is polynomial time solvable. The gap in complexity between monotone and nonmonotone integer programs is substantial. For example, the vertex cover problem is defined on binary variables, so that $U = 1$, and the constraints are of the type $x_i + x_j \geqslant 1$. The vertex cover is a nonmonotone problem which is strongly NP-complete. The monotone version of vertex cover by contrast is solved in integers by the algorithm of Hochbaum and Naor in the complexity of finding a minimum cut on a graph with n nodes and m arcs, $O(mn \log n^2/m)$.

The monotonicity also affects the complexity of the respective linear programming problems. A linear program on a system of constraints with at most two variables per inequality is solved for a *feasible* solution in strongly polynomial time [14,3], whereas for general linear programming problem there is no known strongly polynomial algorithm to find either feasible or optimal solutions. (An algorithm is said to be *strongly polynomial* if its complexity depends only on the size of the problem and not on the value of data coefficients or parameters. For linear programming a strongly polynomial algorithm will have run time complexity which is a polynomial function of the number of variables and constraints only).

When the inequalities are monotone, then the set of feasible solutions forms a *lattice* in the sense that a component-wise maximum and component-wise minimum of a pair of feasible solutions are also feasible. The lattice property allows to find an *optimal* solution to the linear programming problem LP2 when all the objective function coefficients are nonnegative or when all are nonpositive. This is since the optimal solution in this case is the unique *maximal* solution vector or the unique *minimal* solution vector. This optimal (maximal or minimal) solution for the row-monotone case of linear programming, or the feasible solution to a nonmonotone linear programming in two variables per constraint, is attained in strongly polynomial time $O(mn^2 \log m)$ in [14]. (For an additional discussion of the lattice properties of row-monotone problem see [14]).

Hochbaum et al. [13] discuss a transformation that maps a linear or integer program on nonmonotone inequalities in two variables into a respective linear or integer program on monotone inequalities. Hochbaum [12] extends this transformation to monotonizing inequalities with up to three variables per inequality. With this transformation, one can solve the monotone problem first and then map back the feasible or optimal solution to a feasible or superoptimal solution to the nonmonotone problem. For the integer programming problem, integer optimal solutions to the monotone problem are mapped to integer multiples of half solutions to the nonmonotone problem, which are guaranteed to have an objective value only better than the optimal solution—in that sense these are superoptimal solutions. Both papers [13,12] show how this leads to 2-approximation algorithms for many intractable problems.

For column-monotone LP2s the complexity case is somewhat less compelling than for the row-monotone problem as we discuss next. The primal-dual relationship can be exploited to find optimal solutions in strongly polynomial time for column-monotone LP2s with right-hand side vector $\mathbf{b} \geqslant 0$ or $\mathbf{b} \leqslant 0$, as shown by Adler and Cosares [1]. Column-monotone LP2s are effectively the *generalized network flow* problem. In the generalized network flow problem the flow on arc (i, j), f_{ij}, is subject to a gain factor of g_{ij} at the head of the arc. That is, a unit flowing from i towards j arrives at j in the amount g_{ij}. The flow balance constraint at node i for a generalized flow problem is thus of the form,

$$\sum_{j|(i,j)\in A} x_{ij} - \sum_{k|(k,i)\in A} g_{ki}x_{ki} = b_i.$$

The constraint matrix for the generalized flow problem has in each column one 1 and one $-g_{ij}$. Unlike the minimum cost flow problem, where all gainfactors are 1, there are no strongly polynomial algorithms known for solving the generalized network flow problem. Wayne [22] showed how to solve the generalized flow problem in $O(m^3n^2 \log m \log B)$ using a combinatorial algorithm (for B the largest integer in the problem constraints data). There are other, more efficient but noncombinatorial, interior point based algorithms for solving the problem (see [17] for a comprehensive survey).

As for the integer column-monotone case, even finding a *feasible integer* solution to the LP2 problem is NP-hard. To see that consider the decision version of the minimizing makespan scheduling problem on unrelated parallel machines. This problem can be represented as a generalized flow problem on a bipartite graph with jobs on one side and machines on the other. There is one unit demand of each job i which requires processing time p_{ij} on machine j. The decision problem is to assign jobs to machines (without preemption) so that the makespan does not exceed a threshold M. The gain factor on each arc (i,j) in the bipartite graph is p_{ij} and the supply of each machine node is equal to M. Observe that for the scheduling problem the graph is bipartite with supplies on one side, and demands on the other one—namely, it is a generalized transportation problem (we thank Èva Tardos for this observation). In that sense the gap between the column-monotone and nonmonotone LP2 is smaller than the gap that exists between the row-monotone and row nonmonotone LP2s.

In order to clarify the type of problems discussed here we present a typical column for each such problem in Fig. 1. For problems of type LP2 a column of the constraint matrix would be of the form (1) in Fig. 1 without a restriction on the sign of the nonzero entries a and b. A column is said to be monotone if it is of the form (2) with $a, b \geqslant 0$. A column of type (3) with $g \geqslant 0$ characterizes the constraints of a *generalized network flow* problem with g the gain factor for the respective arc corresponding to the flow variable of that column. It is easy to see that a column of type (3) can be generated from a column of type (2) by scaling the variable corresponding to that column by, say, a. This also implies that g is expressed as the ratio of two integers. A column of type (4) characterizes the constraints of the *minimum cost network flow* problem. The constraint matrix that has at most one 1 and at most one -1 in each column is also totally unimodular and thus an optimal and basic solution for the respective linear programming with $\mathbf{b}$ integer is integer as well. The minimum cost network flow can be solved in strongly polynomial time, that is, in run time depends only on the number of arcs and nodes in the network.

$$\begin{array}{cccc}
\begin{bmatrix} 0 \\ \vdots \\ 0 \\ a \\ 0 \\ \vdots \\ 0 \\ b \\ 0 \\ 0 \\ 0 \end{bmatrix} &
\begin{bmatrix} 0 \\ \vdots \\ 0 \\ a \\ 0 \\ \vdots \\ 0 \\ -b \\ 0 \\ 0 \\ 0 \end{bmatrix} &
\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ -g \\ 0 \\ 0 \\ 0 \end{bmatrix} &
\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
(1) & (2) & (3) & (4)
\end{array}$$

Fig. 1. Four types of columns of LP2.

The contribution here is a formalization of the transformation of a linear program LP2 to a respective column-monotone linear program LP2M (M stands for monotone) with twice as many variables and twice as many constraints. This transformation can be regarded as a "dual" analogue of the monotonizing transformation for inequalities with at most two variables in each [13]. The transformation has the property that an integer optimal solution to the transformed monotone LP2M maps to an integer multiple of $\frac{1}{2}$ solution to the nonmonotone LP2 which is a superoptimal solution to problem LP2 in integers. That means that the objective value of this half integral solution is only better (lower for minimization, higher for maximization) than that of the integer optimal solution. This information is useful in generating tight bounds for enumerative algorithms, or if some form of rounding is feasible, it can lead to a 2-approximation solution to the LP2 problem in integers.

In the case where the nonzero entries of $\mathbf{A}$ are 1 or -1 the algorithm presented here delivers a solution which is an integer multiple of half and solves the problem in integers in strongly polynomial time when the right-hand sides vector $\mathbf{b}$ has all even entries. This problem is also known as the *bidirected network flow* problem. This problem was shown by Edmonds [7] to be equivalent to the b-matching problem, and thus polynomially solvable. A proof of this equivalence is given by Lawler ([16], Chapter 6.3). (We are grateful to Mihalis Yannakakis for pointing out this reference). This is discussed in more detail in Section 1.1.

In sum, the transformation procedure described here reduces the general LP2 when the nonzeroes are arbitrary integers (or rational numbers) to a generalized flow problem, and when the nonzeroes are 1 or -1 it reduces it to a minimum cost flow problem.

The remainder of the paper is organized as follows. The next subsection reviews related transformations to monotone formulations known in the literature. We then describe the monotonization transformation algorithm and a heuristic improvement by finding maximal bipartite subgraphs in a corresponding graph. We conclude with several applications of the monotonizing transformation:

1. Solving the b-matching problem in strongly polynomial time. This is a case where the nonzero entries of $\mathbf{A}$ are all 1. This is equivalent to Anstee's approach in [2].
2. Solving efficiently (and in strongly polynomial time) a relaxation of the maximum cut problem. This is the case where the nonzero entries of $\mathbf{A}$ are all 1 or -1, namely a *bidirected network flow problem*. In this case the solution delivered is superoptimal and in integer multiples of $\frac{1}{2}$, or if all right-hand sides are even, it is an optimal integer solution.
3. A strongly polynomial approximation scheme for LP2.
4. An approximation result for LP2 on nonnegative matrices with nonnegative right-hand sides that provides a useful bound for the problem. This is done by demonstrating a transformation to the generalized assignment problem studied by Shmoys and Tardos [21].

1.1. Related results

Fulkerson et al. [10] devised a transformation for converting the b-matching problem to a matrix with at most one 1 and one -1 in each column. There they refer to viewing the adjacency matrix of the b-matching as a "symmetric matrix", which represents a (bipartite) transportation problem. This approach was used by Anstee [2] in order to solve the b-matching problem. Anstee used the symmetric matrix transformation to find the optimal fractional b-matching solution and then round it to an integer solution that is not a perfect b-matching, and then apply an algorithm by Pulleyblank [19] that uses this solution to find efficiently the desired perfect b-matching. A similar use of the procedure is reported in [5,11]. Our procedure can be viewed as an extension of that procedure.

The problem of bidirected flows was shown by Edmonds [7] to be equivalent to the b-matching problem, and thus polynomially solvable. In the capacitated case (with lower and upper bounds on the variables) the transformation amounts to a factor of 4 increase in the number of nodes and the number of edges increases from m to $m+n$. Using the transformation of [10] on the resulting b-matching leads to a representation of the bidirected flow problem as a transportation problem with $8n$ nodes and $2(m+n)$ edges. Note that by comparison our approach has several advantages: it is direct and simpler; it leads to a smaller expansion in the size of the graph; it clarifies the properties of the polytope which are obscured by the known procedures; and it allows for identifying a maximal bipartite subgraph which in practice decreases substantially the size of the minimum cost network flow problem (see Section 2.1).

Cohen and Megiddo [4] discuss bidirected generalized flows in which they represent nonmonotone columns as two arcs one of which carries a negative gain. This allows the use of generalized flow algorithm for nonmonotone LP2 problems, and is thus implicitly a monotonization procedure as well. The procedure we describe maps the LP2 problem to an ordinary generalized flow problem without the need to consider negative gains (a consideration which requires adjustments of existing algorithms).

Hochbaum et al. [13] discuss the monotonization of inequalities with up to two variables, and Hochbaum [12] extends this to monotonizing inequalities with up to three variables per inequality. In the first case this is a 2-factor transformation. In the second case it is a 2-factor transformation only if the "third" variable appears in one constraint only with a coefficient of 1 (otherwise the transformation still applies but the mapping is an $1/\alpha$ factor transformation with

$$\begin{array}{r} \\ i \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ j' \\ \\ \\ \\ \end{array}\begin{bmatrix} 0 \\ a \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ - - - \\ 0 \\ \vdots \\ 0 \\ b \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{array}{r} \\ \\ \\ j \\ \\ \\ \\ \\ \\ i' \\ \\ \\ \\ \\ \\ \end{array}\begin{bmatrix} 0 \\ \vdots \\ 0 \\ b \\ 0 \\ \vdots \\ 0 \\ - - - \\ 0 \\ a \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad\qquad \begin{array}{r} \\ i \\ \\ \\ j \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{bmatrix} 0 \\ a \\ \vdots \\ 0 \\ b \\ 0 \\ \vdots \\ 0 \\ - - - \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{array}{r} \\ \\ \\ \\ \\ \\ \\ \\ \\ i' \\ \\ j' \\ \\ \\ \\ \end{array}\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ - - - \\ 0 \\ a \\ \vdots \\ b \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(a) (b)

Fig. 2. The doubling of column (i,j) of type (a) $(+,+)$ or $(-,-)$ or, (b) $(+,-)$.

$\alpha > 2$). These results were employed in order to obtain 2-approximation algorithms (and in some cases α-approximations).

2. Monotonization algorithm

We show here how to transform an LP2 problem to a column monotone LP2 problem, LP2M, which, after the appropriate scaling, is a generalized flow problem. The transformation involves a factor of $\frac{1}{2}$ in the inverse map of the solution of the monotone problem to the solution of the nonmonotone problem. If the nonzero entries are all 1 or -1, then the procedure reduces the problem to a minimum cost network flow problem, which can be solved by efficient and strongly polynomial techniques, e.g. Orlin's [18].

First we assume without loss of generality that the constraints of a linear program with up to two nonzeroes per column are all equality constraints: Observe that if the constraints of LP2 are inequality constraints then one adds slacks and surplus variables in order to convert the set of inequalities to equalities. This does not affect the structure of the matrix with at most two nonzeroes per column, as slacks or surpluses are columns with one 1 or one -1 per column. Therefore, without loss of generality the constraints (other than lower and upper bound constraints) may be assumed to be equality constraints.

Let $V=\{1,\dots,n\}$ be the set of rows of the matrix $\mathbf{A}$ of the equality constraints. We double the set of rows V and have a copy V' where for each $j \in V$ there is a copy row index $j' \in V'$.

Each column of $\mathbf{A}$ is characterized by the pair of rows that contain the nonzero entries in that column. For each such pair we are only interested in the sign pattern of the coefficients, which is either $(+,+)$, $(-,-)$ or $(-,+)$ (which is same as $(+,-)$. For convenience of exposition we refer to a column with nonzero entries in positions i and j as (i,j). This is even though (i,j) is not a unique identification of a column since the same (i,j) can refer to different columns with different entries in the same pair of rows.

Consider a (i,j) column with either $(+,+)$ or $(-,-)$ pattern then we double the column to (i,j') and (i',j). That means that we replace the column with a in the i position and b in the j position by two columns with the same entries in the respective rows, as in Fig. 2. Let c_{ij} be the coefficient of x_{ij} in the objective function. Then we let $c_{ij'} = c_{ij}$ and $c_{i'j} = c_{ij}$.

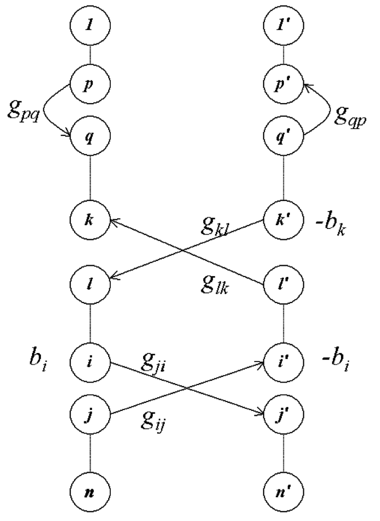

Fig. 3. Network representing the generalized flow problem for a problem with (i,j) a $(+,+)$ column; (k,l) a $(-,-)$ column; and (p,q) a $(+,-)$ column.

If (i,j) is $(+,-)$ then we replace it by two columns (i,j) and (i',j'). We set $c_{i'j'}=c_{ij}$.

We now multiply the set of rows V' by -1 each. This has the effect of reversing the signs of one of the rows for columns with two identical signs and retaining the columns with opposite signs with that property. Notice that for mixed signs columns $(+,-)$ or $(-,+)$ since both entries are either in V or in V' the opposite signs are preserved after multiplying V' by -1.

The matrix after the transformation has twice the number of columns and twice the number of constraints as the original matrix **A**. The signs of the nonzero elements in each column are opposite. We now subdivide (that is, scale) each column with entries $a, -b$ by the value of the positive coefficient, a. Each column is now of the form $1, -\alpha$. After this scaling the formulation is that of the generalized flow network problem. Rows that have positive right-hand sides are interpreted to be supply nodes and those that have negative right-hand sides are demand nodes.

The generalized flow problem can be described on a graph with the set of nodes corresponding to V on one side, and the set of nodes corresponding to V' on the other. Fig. 3 illustrates generalized flow network resulting from the transformation for three types of columns. (i,j) is a $(+,+)$ column, (k,ℓ) is a $(-,-)$ column and (p,q) is a $(+,-)$ column. The quantities g_{ij} are the gain factors which for a monotone column with a in position i and $-b$ in position j is $-b/a$.

Let the column monotone linear programming problem resulting from the transformation be denoted by LP2M.

Lemma 2.1. *If* **x** *is a feasible solution vector for LP2 then $x_{i'j}=x_{ij'}=x_{ij}$ is feasible for LP2M. If $\tilde{\mathbf{x}}$ is a feasible solution to LP2M, then $x_{ij}=(\tilde{x}_{ij'}+\tilde{x}_{i'j})/2$ is a feasible solution to LP2.*

Proof. Let a row i constraint in LP2 be,

$$\sum_{\{k,\ell\}\in A_i} a_{k,\ell}x_{k,\ell}=b_i,$$

where A_i is the set of columns that have nonzero entry in row i. Since **x** is feasible it satisfies this equation as well as the equation corresponding to i' in LP2M,

$$\sum_{\{k,\ell\}\in A_i} -a_{k,\ell}x_{k,\ell}=-b_i=b_{i'}.$$

Hence, the setting $x_{i'j}=x_{ij'}=x_{ij}$ is feasible for LP2M. Note that $\{k,\ell\}\in A_i$ means that either k or ℓ is equal to i.

Now let $\tilde{\mathbf{x}}$ be a feasible solution to LP2M, then,

$$\sum_{\{k,\ell\}\in A_i} a_{k,\ell}\tilde{x}_{k,\ell'}=b_i,$$

$$\sum_{\{k,\ell\}\in A_{i'}} -a_{k,\ell}\tilde{x}_{k',\ell}=-b_i.$$

Subtracting the second equation from the first and noting that $A_i=A_{i'}$, we get,

$$\sum_{\{k,\ell\}\in A_i} a_{k,\ell}(\tilde{x}_{k,\ell'}+\tilde{x}_{k',\ell})=2b_i.$$

It follows that setting $x_{ij}=(\tilde{x}_{ij'}+\tilde{x}_{i'j})/2$ solves the equation $\sum_{\{k,\ell\}\in A_i} a_{k,\ell}x_{k,\ell}=b_i$. □

Corollary 2.1. *The objective value of LP2M and LP2 are equal under the transformation. In particular, it follows that the transformation is approximation preserving.*

2.1. A bipartization improvement

Let a matrix **A** have at most two nozeros per column. The concept of *bipartization* for the matrix **A** is a partition of the set of nodes (rows) V into two subsets that are *consistent* with all edges and arcs (columns).

Definition 1. A bipartition of V into $V_1 \cup V_2$ is *consistent* with the set of columns of a matrix $\mathbf{A}$, if

1. for each $(+,+)$ or $(-,-)$ column (an edge) (i,j), i and j belong to different sets in the bipartition: if $i \in V_1$ then $j \in V_2$, and vice versa.
2. for each $(+,-)$ column (i,j), either $i,j \in V_1$ or $i,j \in V_2$.

With a consistent bipartition we can multiply the set of equalities in V_2 by -1 each and achieve a matrix where each column has at most one positive nonzero and at most one negative nonzero.

A simple greedy algorithm delivers a consistent bipartition, if one exists, or reports that there is no consistent bipartition. This greedy algorithm is fashioned after the greedy algorithm of Even et al. [8] for finding a truth assignment, if one exists, to a 2-satisfiability boolean formula, Conjunctive Normal Form (2CNF), [8]. A sketch of the idea, in terms of our case, is to assign a boolean value (V_1 or V_2) to an unassigned node (row index) v and then "propagate" the implications until there is either a conflict, or there is no further propagation possible. A propagation is to scan all columns in which a node is newly assigned and to assign the other node (if there is another nonzero entry) as per the type of column in the consistency rules. A conflict occurs if there is already an assignment for one node in a column (i,j) and the other node needs to be assigned a value violating the consistency rules. In the latter case, the procedure backtracks to node v and assigns it the opposite value. If there is still another conflict, then there is no consistent bipartition.

We modify this greedy to be used to find a *maximal* consistent bipartition. Note that finding a *maximum* number of nodes consistent bipartition is an NP-hard problem [23]. A maximal bipartition is one that is not a subset of another bipartition which assigns a strictly larger set of nodes. The adjustment of the greedy for *maximality* is to simply leave unassigned all the nodes in the propagated sequence starting from v which leads to a conflict in both possible assignments of v.

At the end of the greedy process there is a partial assignment of a subset of nodes to V_1 and V_2. The remaining columns of the matrix are doubled as in the original procedure, and when done all rows in V_2 are multiplied by -1. This procedure tends to result in smaller size of matrices after the transformation.

3. Application to solving *w*-matching in strongly polynomial time

One well-known class of LP2 problems is the w-matching problem also known as the weighted edge packing problem which is LP2 with both entries of each column are 1. In this problem each node i has an integer weight w_i associated with it and each edge $\{i,j\}$ has a value c_{ij}. The problem is to find integer weights assigned to each edge so that the total weight of edges adjacent to a node does not exceed w_i and the weighted value of all the edges selected is maximized. When the weights w_i are all 1, this is the (nonbipartite) maximum weight matching problem. When the weights are "small" (of polynomial size in the number of nodes) then we refer to the problem as the b-matching problem and it can be solved in (strongly) polynomial time by reducing it to the 1-matching problem. The question whether the w-matching problem can be solved in strongly polynomial time (independent of the values of the weights w) was settled first by Anstee [2].

We let the variable x_{ij} be the weight assigned to edge $\{i,j\}$. The formulation is

$$\text{(w-matching)}\qquad \begin{aligned} \max \quad & \sum_{\{i,j\}\in E} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{j|\{i,j\}\in E} x_{ij} \leqslant w_i, \quad i \in V \\ & x_{ij} \geqslant 0 \text{ integer}, \quad \{i,j\} \in E. \end{aligned}$$

The constraint matrix here has two 1's per column. With the monotonization transformation the problem becomes a bipartite minimum cost flow problem, or the *transportation problem*. That problem is solved in strongly polynomial time, e.g. by the algorithm of Orlin [17] in time $O(n \log n(m + n \log n))$. The mapping back yields a superoptimal solution to the w-matching problem each component of which is an integer multiple of $\frac{1}{2}$.

In the case of w-matching however we have a stronger result. Firstly, the subdeterminants of an LP2 $\{0,1,-1\}$ matrix are "small": A matrix is said to be *separable* if there is a partition of the columns and rows to two subsets (or more) C_1, C_2 and R_1, R_2 such that all nonzero entries in every row and column appear only in the submatrices defined by the sets $C_1 \times R_1$ and $C_2 \times R_2$. The following lemma of Hochbaum et al. applies only to *nonseparable* matrices, since one can construct a separable matrix with an arbitrary number, K, of nonseparable ones on its diagonal, each of determinant 2, thus achieving a matrix with a determinant that is 2^K.

Lemma 3.1 (Hochbaum et al. [13], Lemma 6.1). *The determinants of all nonseparable submatrices of a* $\{0, 1, -1\}$ *LP2 matrix have absolute value at most* 2.

A consequence of Lemma 3.1 is that every basic solution has an integer denominator of absolute value at most 2. (To see that recall that a basic solution is an inverse of a submatrix of **A** multiplied by the right-hand side vector, and apply Cramer's rule for the inverse, which is an integer matrix divided by the scalar 2). Therefore, if the right-hand sides are all even, then any basic solution, and in particular an optimal solution is integer. One can then solve the problem with even right-hand sides with linear programming to obtain an optimal integer solution, or better yet, with the monotonization procedure, we can solve the problem with even right-hand sides in integers in strongly polynomial time using minimum cost transportation problem. Let the right-hand side vector be $\mathbf{w}$ and $\mathbf{w}' = 2\lfloor \mathbf{w}/2 \rfloor$. Then according to the *proximity theorem* of Cook et al. [6], the optimal solution $\mathbf{x}^*$ and the optimal solution $\mathbf{x}'$ to the system with $\mathbf{w}'$ as right-hand sides satisfy

$$\|\mathbf{x}^* - \mathbf{x}'\|_\infty \leqslant n\Delta \|\mathbf{w} - \mathbf{w}'\|_\infty.$$

Since the right-hand side $\|\mathbf{w} - \mathbf{w}'\|_\infty$ is at most 1 and $\Delta \leqslant 2$ this implies,

$$\|\mathbf{x}^* - \mathbf{x}'\|_\infty \leqslant 2n.$$

Therefore, we use the monotonizing to solve the problem for $\mathbf{w}'$ in the complexity of minimum cost network flow on a bipartite network. The solution $\mathbf{x}' - 2n\mathbf{e}$ is then a lower bound on the optimal solution. It thus remains to solve a problem where the weight of each node is at most $2n$. This is an instance of the b-matching (which is in 0-1 variables) that is solved in integers in strongly polynomial time by reducing it to a 1-matching problem [19].

4. Application for solving large maximum cut problems

The maximum cut problem is to partition the nodes of an edge weighted graph into two subsets so that the total weight of the edges with endpoints in different sets is maximum. Although the minimum cut version of the problem is polynomially solvable the maximum cut is a known NP-hard problem. We refer to this problem as max-cut. The description in this section is due to Rinaldi [20].

A linear programming relaxation of max-cut (MC) is the following (for simplicity we assume that the graph is complete; the extension to general graphs is quite simple): For each triple of distinct nodes i, j, and k of the graph, take:

$$x_{ij} + x_{ik} + x_{jk} \leqslant 2,$$

$$x_{ij} - x_{ik} - x_{jk} \leqslant 0,$$

$$-x_{ij} + x_{ik} - x_{jk} \leqslant 0,$$

$$-x_{ij} - x_{ik} + x_{jk} \leqslant 0.$$

These inequalities, also called the *triangle inequalities*, describe the so-called *semimetric polytope*. Each of them defines a facet of the MC polytope; moreover, the trivial inequalities $x_{ij} \geqslant 0$ and $x_{ij} \leqslant 1$ are implied by triangle inequalities thus are redundant and can be dropped from the formulation. So, solving the relaxation over the semimetric polytope provides a bound for the problem.

Fortunately, the separation problem for all these inequalities is trivial as they are $O(n^3)$. So, optimizing over the semimetric polytope provides a polynomially computable bound for MC.

When the graph is large, say, more than 100 nodes, then the corresponding linear programming (LP) is too large to be explicitly represented in an LP solver, thus we need to optimize with a cutting plane algorithm. However, such an algorithm takes usually *many* iterations, each LP may get quite hard to solve (even with state-of-the-art commercial LP codes), and, as a result, it is out of the question to optimize even for graphs of 60 nodes in a decent amount of time (using the Simplex Dual). One can go a bit higher (say, 100 nodes) with a Barrier code. But also this algorithm gets into troubles for larger graphs, basically because of excessive memory requirements.

A different approach is proposed in [9]: all the triangle inequalities are dualized and the Lagrangian problem is solved using the bundle method. The only constraints left are the upper and the lower bounds. The results are quite good: one can optimize in much shorter time than with Simplex/Barrier based approaches (with comparable primal/dual error tolerances) and therefore address larger problems. The Lagrangian problem however needs to be solved numerous times in the process.

A better Lagrangian problem can be obtained if, instead of relaxing all the triangle inequalities, some of them are left explicitly expressed. In particular, the inequalities that are not dualized are given by all the triangles that share a selected root node r:

$$\begin{aligned} &x_{ri} + x_{rj} + x_{ij} \leqslant 2,\\ &x_{ri} - x_{rj} - x_{ij} \leqslant 0,\\ &-x_{ri} + x_{rj} - x_{ij} \leqslant 0,\\ &-x_{ri} - x_{rj} + x_{ij} \leqslant 0. \end{aligned} \tag{1}$$

These inequalities define the so-called *rooted semimetric polytope*. We are interested in optimizing a linear function over this polytope with a very fast algorithm (that will have to be run over and over during the execution of the bundle method).

By applying a variable substitution to (1), taking its dual and applying another variable substitution to it, one obtains the following linear program:

$$\begin{aligned} \min \quad & \sum u_{rj} + \sum v_{ij} \\ \text{s.t.} \quad & \\ & u_{rj} + \sum v_{ij} + \sum_{k<j} w_{kj} - \sum_{k>j} w_{jk} \geqslant d_{rj} \\ & v_{ij} \leqslant d_{ij} \quad \text{for } c_{ij} > 0, \\ & w_{ij} \leqslant d_{ij} \quad \text{for } c_{ij} \leqslant 0, \\ & u, v, w \geqslant 0, \end{aligned}$$

where the c_{ij}'s are the MC objective function coefficients and the d_{ij}'s depend only on the c_{ij}'s.

The u variables are associated with the edges adjacent to the root node r of the graph; the v and the w variables are associated with the remaining edges of the graph with positive and nonpositive c_{ij}, respectively.

This problem can be seen as a minimum cost flow problem in a network with edges (v) and arcs (w). In other words, this is a bidirected cost flow problem. The monotonization transformation maps this problem into a regular minimum cost flow problem. As reported in [9], solving it as a minimum cost network improves the run time of the Lagrangian and bundle method substantially and makes it possible to solve considerably larger instances.

5. Two approximation results

5.1. *Strongly polynomial approximation scheme for LP2*

Once the problem is reduced to a generalized flow problem, algorithms for solving such problems can be used to solve LP2M. Thus Wayne's $O(m^3n^2 \log m \log B)$ combinatorial algorithm can be used to solve the problem optimally. Wayne [22] also devised a strongly polynomial approximation scheme for the generalized flow problem. A solution within ε of the optimum is obtained in time $O(m^2n^2 \log m \log \frac{1}{\varepsilon})$. This approximation scheme is directly applicable to LP2 since the transformation is approximation preserving.

5.2. *An approximation algorithm for nonnegative LP2*

Here we show how an approximation result for generalized assignment is translated into a corresponding result for nonnegative LP2 that has the matrix $\mathbf{A}$ and the vector $\mathbf{b}$ all with nonnegative entries.

Consider the generalized assignment problem. This is a generalized minimum cost flow problem defined on a bipartite network with one side having supply nodes and the other demand nodes. Shmoys and Tardos [21] proposed an algorithm that either establishes that there is no feasible solution of cost C or else it finds a solution of cost at most C which violates the supply restrictions by a factor 2 at most.

Consider now an LP2 problem in integers with all right-hand sides $\mathbf{b}$ nonnegative and all entries of $\mathbf{A}$

nonnegative. This means that if the constraints are inequality constraints then they must be of packing type so that setting it up as equalities requires adding slacks, with coefficients 1, and it preserves the nonnegativity of **A**. For nonnegative problems of this type the application of the monotonization algorithm results in an LP2M problem that has all constraint coefficients of the rows in V' nonpositive. We scale each column of LP2M by the positive coefficient in the row of V. Let the largest such coefficient be a.

Next we apply the algorithm of Shmoys and Tardos which solves the scaled LP2M problem in integers. When we map back the solution we get a solution of value not exceeding C but possibly violating the constraints by a factor up to 2. The values of the variables in that solution are rationals with largest denominator equal to $2a$. Notice that a linear programming relaxation of the problem would give an optimal solution with largest denominator as large as the largest subdeterminant which can be exponentially larger than a (as large as $O(a^n)$). Therefore, the solution provided by the procedure delivers a tighter bound for the integer problem permitting the factor 2 violation of the right-hand sides.

Therefore, there is a polynomial time algorithm for solving the LP2 problem with nonnegative coefficients in rationals with largest denominator a and within a factor of 2 of the right-hand sides.

References

[1] I. Adler, S. Cosares, A strongly polynomial algorithm for a special class of linear programs, Oper. Res. 39 (6) (1991) 955–960.

[2] R.P. Anstee, A polynomial algorithm for *b*-matching: an alternative approach, Inform. Process. Lett. 24 (1987) 153–157.

[3] E. Cohen, N. Megiddo, Improved algorithms for linear inequalities with two variables per inequality, SIAM J. Comput. 23 (1994) 1313–1347.

[4] E. Cohen, N. Megiddo, New algorithms for generalized network flows, Math. Programming A 64 (3) (1994) 325–336.

[5] W. Cook, W. Cunningham, W. Pulleyblank, L. Schrijver, Combinatorial Optimization, Wiley, New York, 1998, pp. 185–187.

[6] W. Cook, A.M.H. Gerards, A. Schrijver, È. Tardos, Sensitivity theorems in integer linear programming, Math. Programming 34 (1986) 251–264.

[7] J. Edmonds, An Introduction to Matching: Lecture Notes, University of Michigan, Ann Arbor, 1967.

[8] S. Even, A. Itai, A. Shamir, On the complexity of timetable and multicommodity flow problems, SIAM J. Comput. 5 (4) (1976) 691–703.

[9] A. Frangioni, A. Lodi, G. Rinaldi, New approaches for optimizing over the semimetric polytope, IASI-CNR, Rome, 2003, in preparation.

[10] D.R. Fulkerson, A.J. Hoffman, M.H. McAndrew, Some properties of graphs with multiple edges, Can. J. Math. 17 (1965) 166–177.

[11] A.M.H. Gerards, Matching. Vol. 7 of Handbooks in Operations Research and Management Science, North-Holland, Amsterdam, 1995, pp. 135–224 (Chapter 3).

[12] D.S. Hochbaum, Solving integer programs over monotone inequalities in three variables: a framework for half integrality and good approximations, Eur. J. Oper. Res. 140 (2) (2002) 291–321.

[13] D.S. Hochbaum, N. Megiddo, J. Naor, A. Tamir, Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality, Math. Programming 62 (1993) 69–83.

[14] D.S. Hochbaum, J. Naor, Simple and fast algorithms for linear and integer programs with two variables per inequality, SIAM J. Comput. 23 (6) (1994) 1179–1192.

[15] J.C. Lagarias, The computational complexity of simultaneous diophantine approximation problems, SIAM J. Comput. 14 (1985) 196–209.

[16] E.L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston, New York, 1976.

[17] J.D. Oldham, Combinatorial approximation algorithms for generalized flow problems, J. Algorithms 38 (2001) 135–169.

[18] J.B. Orlin, A faster strongly polynomial algorithm for the minimum cost flow problem, Oper. Res. 41 (1993) 338–350.

[19] W.R. Pulleyblank, Faces of matching polyhedra, Ph.D. Thesis, Department of Combinatorics and Optimization, University of Wateloo, 1973.

[20] G. Rinaldi, Oberwolfach, 2002, private communication.

[21] D.B. Shmoys, È. Tardos, An approximation algorithm for the generalized assignment problem, Math. Programming 62 (1993) 461–474.

[22] K.D. Wayne, A polynomial combinatorial algorithm for generalized minimum cost flow, Math. Oper. Res. 27 (3) (2002) 445–459.

[23] M. Yannakakis, Node-deletion problems on bipartite graphs, SIAM J. Comput. 10 (1981) 310–327.