Mathematical Programming 63 (1994) 23–45
North-Holland

A technique for speeding up the solution of the Lagrangean dual

Dimitris Bertsimas and James B. Orlin

Sloan School of Management, MIT, Cambridge, MA, USA

Received 29 April 1991
Revised manuscript received 8 January 1993

We propose techniques for the solution of the LP relaxation and the Lagrangean dual in combinatorial optimization and nonlinear programming problems. Our techniques find the optimal solution value and the optimal dual multipliers of the LP relaxation and the Lagrangean dual in polynomial time using as a subroutine either the Ellipsoid algorithm or the recent algorithm of Vaidya. Moreover, in problems of a certain structure our techniques find not only the optimal solution value, but the solution as well. Our techniques lead to significant improvements in the theoretical running time compared with previously known methods (interior point methods, Ellipsoid algorithm, Vaidya's algorithm). We use our method to the solution of the LP relaxation and the Langrangean dual of several classical combinatorial problems, like the traveling salesman problem, the vehicle routing problem, the Steiner tree problem, the k-connected problem, multicommodity flows, network design problems, network flow problems with side constraints, facility location problems, K-polymatroid intersection, multiple item capacitated lot sizing problem, and stochastic programming. In all these problems our techniques significantly improve the theoretical running time and yield the fastest way to solve them.

Key words: Lagrangean relaxation, linear programming relaxation, polynomial algorithm.

1. Introduction

During the last two decades combinatorial optimization has been one of the fastest growing areas in mathematical programming. One major success has been computational: researchers have been able to solve large-scale instances of some NP-hard combinatorial optimization problems. The successful solution approaches typically rely on solving, either approximately or exactly, the linear programming (LP) relaxation or the Lagrangean dual of an integer programming formulation of the problem.

The successes in using linear programming methodology (LP relaxations, polyhedral combinatorics) have been very impressive. Starting with the seminal computational research conducted by Crowder and Padberg [9] and Padberg and Hong [28], researchers have now been able to solve to optimality a number of applications including traveling salesman problems with up to 2000 nodes (Padberg and Rinaldi [29]), a variety of large scale (up to about 2000 integer variables), real world zero–one business planning problems (Crowder,

Correspondence to: Prof. Dimitris Bertsimas, Sloan School of Management, E53-359, MIT, 50 Memorial Drive, Cambridge, MA 02139, USA.

SSDI 0025-5610(93)E0011-3

Johnson and Padberg [8]), input–output matrices that arise in economic planning (Grötschel, Jünger and Reinelt [15]), multiple item capacitated lot size problems that arise in production planning (Barany, Van Roy and Wolsey [5], Eppen and Martin [11], Leung, Magnanti and Vachani [23]), strategic planning problems (Johnson, Kostreva and Suhl [20]), production planning problems with changeover costs (Magnanti and Vachani [25]), and certain spin glass problems that arise in physics (Barahona et al. [4]).

These successes show that linear programming and Lagrangean relaxations can play an important role in solving many problems met in practice. The landmark in the development of Lagrangean relaxation (see for example, Geoffrion [13] or Fisher [12]) for combinatorial optimization problems were the two papers for the traveling salesman problem (TSP) by Held and Karp [17], [18]. In the first paper, Held and Karp [17] proposed a Lagrangean relaxation based on the notion of 1-tree for the TSP. Using a complete characterization of the 1-tree polytope, they showed that this Lagrangean relaxation gives the same bound as the LP relaxation of a classical formulation of the TSP. In the second paper, Held and Karp [18] introduced a method, which is now known under the name of subgradient optimization (see also Held, Wolfe and Crowder [19]), to solve the Lagrangean dual. Despite its success in computational experiments, subgradient optimization is not known to be a polynomial method.

In this paper we propose techniques for the solution of the LP relaxation and the Lagrangean dual in combinatorial optimization problems. We also extend our methods to solve the Lagrangean dual of general nonlinear programming problems. Our techniques find the optimal solution value and the optimal dual multipliers of the LP relaxation and the Lagrangean dual in polynomial time using as a subroutine either the Ellipsoid algorithm [22] or the recent algorithm of Vaidya [33]. Moreover, in problems of a certain structure our techniques find not only the optimal solution value, but the solution as well. The approach of using the ellipsoid algorithm to solve the Lagrangean dual was independently developed by Chandru and Trick [6]. Their paper has two primary foci. First, they used the ellipsoid algorithm for solving the Lagrangean dual in order to prove that certain related problems were solvable in polynomial time. Second, they implemented their approach for solving the Held and Karp bound for the traveling salesman problem. In this paper we focus on problems already known to be solvable in polynomial time, but where ellipsoid like methods lead to significant computational speedups. Surprisingly, our method is significantly faster than interior point methods and ellipsoid like methods directly applied to the problem, and the current methods do lead to significant speedups in the worst case complexity to solve the Lagrangean dual.

We apply our techniques to the solution of the LP relaxation and the Lagrangean dual of several classical combinatorial problems, like the traveling salesman problem, the vehicle routing problem, the Steiner tree problem, the k-connected problem, network design problems, network flow problems with side constraints, facility location problems, K-polymatroid intersection, multiple item capacitated lot sizing problem, etc. We also include an application to stochastic programming. In all these applications our techniques significantly improve the running time and yield the fastest way to solve them. Our technique can also

be used to speed up the solution of the dual of multicommodity flow problems in certain cases.

The paper is structured as follows: In Section 2 we introduce our method and its variations. In Section 3 we apply it to a wide variety of classical combinatorial optimization problems and stochastic programming. The final section contains some concluding remarks.

2. The theoretical development of new techniques

In this section we develop the algorithm to solve linear programming problems. The method is particularly useful for problems of special structure as those appearing in the solution of LP relaxations of combinatorial optimization problems.

2.1. An algorithm for linear programs of special structure

Consider the following LP:

$$(P_1) \quad z_1 = \text{Min } cx$$

$$\text{subject to} \quad Ax = b, \tag{1}$$

$$x \in S,$$

where S is a polyhedron containing no lines. We will assume throughout the paper that all the polyhedra we consider have no lines and thus have at least one extreme point. Let x^k, $k \in J$, be the extreme points of S, and similarly let w^k, $k \in R$, be the extreme rays of S. Then applying the resolution theorem for the polyhedron S we obtain that any point $x \in S$ can be written as

$$x = \sum_{k \in J} \lambda_k x^k + \sum_{k \in R} \theta_k w^k, \quad \lambda_k, \theta_k \geqslant 0, \sum_{k \in J} \lambda_k = 1.$$

Substituting in (1) we obtain that problem (P_1) is equivalent with

$$(P_1') \quad \text{Min } z_1 = \sum_{k \in J} \lambda_k (cx^k) + \sum_{k \in R} \theta_k (cw^k)$$

$$\text{subject to} \sum_{k \in J} \lambda_k (Ax^k) + \sum_{k \in R} \theta_k (Aw^k) = b,$$

$$\sum_{k \in J} \lambda_k = 1,$$

$$\lambda_k, \theta_k \geqslant 0.$$

The dual of the above problem is

$$(D_1) \quad z_1 = \text{Max } yb + \sigma$$

$$\text{subject to} \quad y(Ax^k) + \sigma \leqslant cx^k, \quad k \in J,$$

$$y(Aw^k) \leqslant cw^k, \quad k \in R.$$

Because of strong duality, problem (D_1) has exactly the same value as the original problem (P_1). Problem (D_1), however, has a potentially exponential number of constraints. In order to find z_1 we will solve problem (D_1) using either the ellipsoid method or the recent algorithm of Vaidya [33].

Given a solution y_0, σ_0 for the dual problem, we need to solve the separation problem:

$$\text{(SEP)} \quad u = \text{Min } (c - y_0 A)x$$
$$\text{subject to} \quad x \in S.$$

If the solution of this problem is bounded and $u \geqslant \sigma_0$ then the solution (y_0, σ_0) is feasible in (D_1) and thus we can start Vaidya's sliding objective algorithm (see [33]) or the Sliding objective Ellipsoid algorithm (see for example Nemhauser and Wolsey [27, p. 155]) with the same separation problem. If $u < \sigma_0$, a new extreme point is generated, which is added to (D_1). If, on the other hand, the subproblem is unbounded, then a new extreme ray of S is generated, which is also added to (D_1). In the last two cases the ellipsoid method or Vaidya's algorithm will continue with one extra constraint. The algorithm will compute the optimal value z_1^* of (D_1) (which is also equal to the optimal value of (P_1)), as well as the optimal dual solution y^* and σ^*.

In general, Vaidya's [33] algorithm for an LP with k variables takes $O(kL)$ iterations with a running time

$$O(TkL + M(k)kL), \tag{2}$$

where T is the number of arithmetic operations to solve the separation problem (SEP), L is the input size of problem (P_1) and $M(k)$ is the number of arithmetic operations for multiplying two $k \times k$ matrices. (It is known that $M(k) = O(k^{2.38})$ [7].) For comparison, the number of iterations of the ellipsoid algorithm is $O(k^2L)$ and its running time is $O(Tk^2L + k^4L)$. Note that the Ellipsoid algorithm does not benefit from fast matrix multiplication. So, overall Vaidya's algorithm has a better worst-case time complexity than the ellipsoid method. For this reason we will use Vaidya's algorithm in the rest of the paper.

For problem (P_1) let n be the number of variables, let m be the number of constraints $Ax = b$ and let $T(n)$ be the number of arithmetic operations to solve the separation problem (SEP). Then, the number of variables in (D_1) is $k = m + 1$ and thus from (2) the overall time complexity to find z_1^* and the optimal dual solution y^*, σ^* of (D_1) is $O(T(n)mL + M(m)mL)$. We summarize the previous developments in the following theorem.

Theorem 1. *The optimal solution value z_1^* of problem (P_1) and the optimal dual variables y^*, σ^* can be found in $O(T(n)mL + M(m)mL)$ arithmetic operations.* □

A natural question is whether we can also find the optimal primal solution x^* of (P_1). For example, since our algorithm takes $O(mL)$ calls to the separation problem (SEP), at most $O(mL)$ constraints from (D_1) will have been generated and correspondingly $O(mL)$ variables λ_k, θ_k in (P_1'). Applying an interior point algorithm in (P_1') we can find the optimal λ_k^*, θ_k^* and x^* in $O((mL)^3L)$, to be contrasted with applying an interior point

algorithm directly to (P_1), which leads to an $O(n^3L)$ running time. Unfortunately, this running time is not attractive except for problems where $L=O(1)$. Moreover, if the optimal solutions of problems (D_1), (P_1') are unique we can apply the complementary slackness property and thus we can find the optimal x^* in $O(mL)$, which does not change the overall complexity of the method. In applications, however, we often want to find the optimal solution value rather than the solution of the LP or the Lagrangean relaxation, since the solution value can be later used in a branch and bound algorithm.

2.2. LPs with more than one subproblem

We now generalize the technique of the previous subsection to handle problems of the form:

$$(P_2) \quad z_2 = \text{Min} \sum_{r=1}^{N} c_r x_r$$

$$\text{subject to} \quad \sum_{r=1}^{N} A_r x_r = b, \tag{3}$$

$$x_r \in S_r, \quad r=1, \ldots, N,$$

where each S_r is a polyhedron with the property that optimizing over S_r is easy.

Let x_r^k, $k \in J_r$ be the extreme points of S_r, and similarly let w_r^k, $k \in R_r$ be the extreme rays of S_r. Using the same technique as before we find that problem (P_2) is equivalent to problem

$$(D_2) \quad z_2 = \text{Max } yb + \sum_{r=1}^{N} \sigma_r$$

$$\begin{aligned}
\text{subject to} \quad & y(A_1 x_1^k) + \sigma_1 \leqslant c_1 x_1^k, \quad k \in J_1, \\
& \vdots \\
& y(A_N x_N^k) + \sigma_N \leqslant c_N x_N^k, \quad k \in J_N, \\
& y(A_1 w_1^k) \leqslant c_1 w_1^k, \quad k \in R_1, \\
& \vdots \\
& y(A_N w_N^k) \leqslant c_N w_N^k, \quad k \in R_N.
\end{aligned}$$

In order to apply Vaidya's algorithm to (D_2) we should be able to solve efficiently the separation problem, which in this case decomposes to the N subproblems:

$$(SEP_r) \quad \text{Min } (c_r - yA_r)x_r$$

$$\text{subject to} \quad x_r \in S_r.$$

As a result, Vaidya's algorithm applied to (D_2) with separation problems (SEP_r), $r=1, \ldots, N$, computes the optimal value z_2^* of (D_2) (which is also equal to the optimal solution value of (P_2)) and the optimal dual solution y^*, σ_r^*, $r=1, \ldots, N$.

Let n_r be the number of variables in S_r, let m be the number of constraints $\sum_{r=1}^{N} A_r x_r = b$

and let $T_r(n_r)$ be the number of arithmetic operations to solve the separation problem (SEP_r). Therefore, the total number of arithmetic operations to solve the entire separation problem is $\sum_{r=1}^{N} T_r(n_r)$. Since the number of variables in (D_2) is $k=m+N$, we obtain from (2) that the overall time complexity to solve (D_2) is

$$O\left(\left[\sum_{r=1}^{N} T_r(n_r)\right](m+N)L+M(m+N)(m+N)L\right).$$

Therefore:

Theorem 2. *The optimal solution value z_2^* of problem (P_2) and the optimal dual variables y^*, σ_r^*, $r=1, \ldots, N$, can be found in $O([\sum_{r=1}^{N} T_r(n_r)](m+N)L+M(m+N)(m+N)L)$ arithmetic operations.* □

2.3. Cost splitting

We now consider LPs, in which the feasible region is the intersection of K polyhedra. Examples in this category are the K-matroid and K-polymatroid intersection problems, LP relaxation problems of combinatorial problems with this property, etc. In order to speed up algorithms for the solution of such problems we combine our technique with cost splitting or Lagrangean decomposition (Jornsten and Nasberg [21], see also Nemhauser and Wolsey [27, p. 333]), a method which has been applied to strengthen the bounds given by Lagrangean relaxation. Consider the LP

$$\begin{aligned} (\mathrm{P}_3) \quad z_3 &= \text{Min } cx \\ &\text{subject to } x \in S_1 \cap S_2 \cap \cdots \cap S_K, \end{aligned} \tag{4}$$

where $S_1, \ldots, S_K$ are polyhedra, over which we can optimize easily. We rewrite the problem as follows:

$$\begin{aligned} (\mathrm{P}_3') \quad z_3 &= \text{Min } cx_1 \\ &\text{subject to } x_1 - x_2 = 0, \\ &\qquad\quad x_2 - x_3 = 0, \\ &\qquad\quad \vdots \\ &\qquad\quad x_{k-1} - x_k = 0, \\ &\qquad\quad x_1 \in S_1,\ x_2 \in S_2,\ \ldots,\ x_K \in S_K. \end{aligned} \tag{5}$$

Let x_r^k, $k \in J_r$ and w_r^k, $k \in R_r$ be the extreme points and extreme rays of S_r $(r=1, 2, \ldots, K)$. Applying the resolution theorem to the polyhedra S_r $(r=1, 2, \ldots, K)$ we rewrite (P_3') in terms of x_r^k and w_r^k. Taking the dual of (P_3') we obtain

$$(D_3) \quad z_3 = \text{Max } \sigma_1 + \sigma_2 + \ldots + \sigma_K$$

$$\text{subject to} \quad y_1 x_1^k + \sigma_1 \leqslant c x_1^k, \quad k \in J_1,$$

$$(y_r - y_{r-1}) x_r^k + \sigma_r \leqslant 0, \quad k \in J_r, \ r = 2, \ldots, K-1,$$

$$-y_K x_K^k + \sigma_K \leqslant 0, \quad k \in J_K,$$

$$y_1 w_1^k \leqslant c w_1^k, \quad k \in R_1,$$

$$(y_r - y_{r-1}) w_r^k \leqslant 0, \quad k \in R_r, \ r = 2, \ldots, K-1,$$

$$y_K w_K^k \leqslant 0, \quad k \in R_K.$$

In order to apply Vaidya's algorithm to (D_3) we should be able to solve efficiently the separation problem, which in this case decomposes to the K subproblems:

(SEP_r) Min $v_r x_r$

$$\text{subject to} \quad x_r \in S_r,$$

with $v_1 = c - y_1$, $v_K = y_K$ and $v_r = y_{r-1} - y_r$ for $r = 2, \ldots, K-1$. As a result, Vaidya's algorithm applied to (D_3) with separation problems (SEP_r), $r = 1, \ldots, K$, computes the optimal value z_3^* of (D_3) (which is also equal to the optimal solution value of (P_3)) and the optimal dual solution y_r^*, σ_r^*, $r = 1, \ldots, K$. In order to analyze the running time of the algorithm let n be the number of variables, and let $T_r(n)$ be the number of arithmetic operations to solve the separation problem (SEP_r) respectively. From (2) with $k = Kn$ we obtain:

Theorem 3. *The optimal solution value z_3^* of problem (P_3) and the optimal dual variables y^*, σ_1^*, σ_2^*, ..., σ_K^*, can be found in $O([\sum_{r=1}^{K} T_r(n)]KnL + M(Kn)KnL)$ arithmetic operations.* □

The cost splitting approach will be superior to applying Vaidya's method directly to (P_3) whenever the separation problem over S_r is more difficult than the optimization problem over S_r. Examples in this category include polymatroid polytopes. As a result, we will see that the cost splitting approach leads to significant improvements in the K-polymatroid intersection problem.

2.4. Applications to Lagrangean relaxation

Lagrangean relaxation is a primary method used in practice to find good bounds for combinatorial optimization problems. Consider the integer programming problem:

$$z_{IP} = \text{Min } cx$$

$$\text{subject to} \quad Ax \leqslant b, \qquad (6)$$

$$x \in S = \{x \in \mathbb{Z}_+^n : A_1 x \leqslant b_1\}.$$

Suppose we want to solve the Lagrangean dual

$$(P_4) \quad z_{LD} = \underset{\lambda \geqslant 0}{\text{Max}} \ \underset{x \in S}{\text{Min}} \ [cx + \lambda(Ax - b)].$$

It is well known that $z_{LD} = \text{Min}\{cx: Ax \leqslant b, x \in \text{conv}(S)\}$ and also $z_{LP} \leqslant z_{LD} \leqslant z_{IP}$, i.e., the Lagrangean dual gives bounds at least as good as the LP relaxation (Held and Karp [17], Geoffrion [13]).
In order to find z_{LD} we rewrite (P_4) as follows:

$$(P_4') \quad z_{LD} = \text{Max } w$$

$$\text{subject to} \quad w \leqslant cx + \lambda(b - Ax) \quad \text{for all } x \in S,$$

$$\lambda \geqslant 0.$$

In order to apply Vaidya's algorithm to (P_4') we need to solve the following separation problem: Given (w, λ), with $\lambda \geqslant 0$,

$$(\text{SEP}) \quad \text{Min } (c - \lambda A)x \qquad (7)$$

$$\text{subject to} \quad x \in S.$$

If $T(n)$ is the number of arithmetic operations to solve the separation problem (7) and there m constraints $Ax \leqslant b$, then the application of Vaidya's algorithm to the reformulation (P_4') leads to:

Theorem 4. *The optimal solution value of the Lagrangean dual z_{LD} of problem (P_4) and the optimal Lagrange multipliers λ^*, can be found in $O(T(n)mL + M(m)mL)$ arithmetic operations.* □

Remark. After $O(mL)$ iterations our algorithm has generated $O(mL)$ constraints of (P_4') of the form $w \leqslant cx_i + \lambda(b - Ax_i)$, where $x_i \in S$ is the solution of some separation problem (SEP) that was generated during the algorithm. At least one of these constraints is satisfied with equality at the optimal solution (w^*, λ^*), otherwise we could further decrease z_{LD}. The corresponding x_i for this constraint is an optimal primal solution. Clearly there might be alternate optimal solutions (all x_i's that satisfy their corresponding constraint with equality at the optimal solution (w^*, λ^*)). Therefore, the algorithm constructs an optimal solution x^* of (P_4) without increasing the running time.

2.5. Variable relaxation

We will now consider LPs where instead of complicating constraints we have complicating variables. Our goal is again to speed up the computation. Consider the LP

$$(P_5) \quad z_5 = \text{Min } cx + dy$$

$$\text{subject to} \quad Ax + By \geqslant b, \qquad (8)$$

$$x \geqslant 0, \quad y \geqslant 0.$$

Our development here is similar in spirit to Benders decomposition. Suppose that the problem is easy to solve whenever y is fixed. Examples in this category are LP relaxations of fixed charge network design problems. We write problem (P_5) as follows:

$$z_5 = \operatorname*{Min}_{y \geqslant 0} \left[dy + \operatorname*{Min}_{x:\ Ax \geqslant b - By,\ x \geqslant 0} cx \right].$$

Taking the dual in the inner minimization we obtain

$$z_5 = \operatorname*{Min}_{y \geqslant 0} \left[dy + \operatorname*{Max}_{\pi A \leqslant c,\ \pi \geqslant 0} \pi(b - By) \right]$$

which can be written as follows:

$$\begin{aligned} z_5 = \operatorname{Min}\ & dy + \sigma \\ \text{subject to}\quad & \pi(b - By) \leqslant \sigma \quad \text{for all } \pi \in \{\pi:\ \pi A \leqslant c,\ \pi \geqslant 0\}. \end{aligned} \tag{9}$$

We will solve problem (9) using Vaidya's algorithm. Given a (y, σ), the separation problem is

$$\begin{aligned} (\text{SEP})\quad \operatorname{Max}\ & \pi(b - By) \\ \text{subject to}\quad & \pi A \leqslant c, \\ & \pi \geqslant 0, \end{aligned}$$

which by taking the dual is equivalent to

$$\begin{aligned} (\text{SEP}')\quad \operatorname{Min}\ & cx \\ \text{subject to}\quad & Ax \geqslant b - By, \\ & x \geqslant 0. \end{aligned}$$

If in the original problem (P_5) $x \in \mathbb{R}^n_+$ and $y \in \mathbb{R}^m_+$ and $T(n)$ is the number of arithmetic operations to solve both separation problems (SEP) and (SEP′) we obtain from (2) that Vaidya's algorithm takes $O(T(n)mL + M(m)mL)$. Note that in this case the algorithm not only produces the optimal solution value, but in addition it finds the optimal y^*. Given the optimal y^*, we can solve problem (SEP′) in $T(n)$ arithmetic operations to find the optimal solution x^* as well. Therefore, in this case we can derive the optimal solution value as well as the optimal solution. Therefore:

Theorem 5. *The optimal solution value z_5^* of problem (P_5) and the optimal solution x^*, y^* can be found in $O(T(n)mL + M(m)mL)$ arithmetic operations.* □

2.6. Extensions to nonlinear programming

In this section we extend the approach of Section 2.4 to solve the Lagrangean dual of an arbitrary nonlinear programming problem. For convex optimization problems

$$\text{Min } f(x)$$
$$\text{subject to} \quad g_i(x) \leqslant 0, \quad i = 1, \ldots, m,$$

with k variables. Vaidya's method takes $O(k \log(1/\varepsilon))$ iterations to find an approximation with $\varepsilon > 0$ of the optimal solution value. The running time is $O([T + M(k)]k \log(1/\varepsilon))$, where T is the time to solve the separation problem, i.e., the time to find whether a given x_0 satisfies $g_i(x_0) \leqslant 0$, $i = 1, \ldots, m$, and if not to find a violated constraint.

Consider the nonlinear programming problem

$$z_{\text{NLP}} = \text{Min } f(x)$$
$$\text{subject to} \quad g_i(x) \leqslant 0, \quad i = 1, \ldots, m, \tag{10}$$
$$x \in X \subseteq \mathbb{R}^n.$$

Let $g(x) = [g_1(x), \ldots, g_m(x)]$. Note that the functions $f(x)$, $g_i(x)$ are arbitrary (not necessarily convex). Suppose we want to solve the Lagrangean dual

$$(\text{P}_6) \quad z_{\text{LD}} = \underset{\lambda \geqslant 0}{\text{Max}} \; \underset{x \in X}{\text{Min}} \; L(x, \lambda),$$

with $L(x, \lambda) = f(x) + \lambda g(x)$.

It is well known (Shapiro [31, p. 145]) that $z_{\text{LD}} \leqslant z_{\text{NLP}}$ (weak duality), and $z_{\text{LD}} = z_{\text{NLP}}$ (strong duality) if the functions $f(x)$, $g_i(x)$ are convex and X is a convex set. Moreover, problem (P_6) is a concave maximization problem, since $L(\lambda) = \text{Min}_{x \in X} L(x, \lambda)$ is a concave function, i.e., the Lagrangean dual of an arbitrary mathematical programming problem is a concave maximization problem.

In order to find z_{LD} we rewrite (P_6) as follows:

$$(\text{P}_6') \quad z_{\text{LD}} = \text{Max } w$$
$$\text{subject to} \quad w \leqslant f(x) + \lambda g(x) \quad \text{for all } x \in X,$$
$$\lambda \geqslant 0.$$

In order to apply Vaidya's algorithm to (P_6') we need to solve the following separation problem: Given (w, λ), with $\lambda \geqslant 0$,

$$(\text{SEP}) \quad \text{Min } f(x) + \lambda g(x) \tag{11}$$
$$\text{subject to} \quad x \in X.$$

If $T(n)$ is the number of arithmetic operations to solve the separation problem (11) and there m constraints $g_i(x)$, $i = 1, \ldots, m$, then the application of Vaidya's algorithm to the reformulation (P_6') leads to:

Theorem 6. *Given an $\varepsilon > 0$, an approximation within ε of the optimal solution value of the Lagrangean dual z_{LD} of problem (P_6) and the optimal Lagrange multipliers λ^* can be found in $O([T(n) + M(m)]m \log(1/\varepsilon))$ arithmetic operations.* □

Remarks. 1. In the case of nonlinear programming problems the optimal solution might have irrational components and therefore we can only speak about approximate solutions in the Turing machine model. In the case of linear programming it is enough to find a solution within $\varepsilon = 2^{-L}$, since given that we have an approximate solution we can use rounding as the optimal solution has rational components.

2. As we remarked in the end of Section 2.4 we can find (using exactly the same method) an optimal solution x^* of (P_6) without increasing the running time.

2.7. Summary of algorithms

In this section we summarize our findings in order to facilitate the reading and for future reference. In Table 1 we summarize the problem type we considered and the separation algorithm we need to solve. Table 2 includes the running times. $T(n)$ always refers to the time to solve the separation problem in Table 1 and $M(n)$ is the number of arithmetic operations to multiply two $n \times n$ matrices.

Note that our methods always produce the optimal solution value and the optimal dual variables. In problems (P_4), (P_5) and (P_6) the algorithm produces primal optimal solutions as well without increasing further the running time. In problems (P_1), (P_2) and (P_3) the method can produce primal optimal solutions without increasing the running time if the optimal solution is unique; otherwise it can produce primal optimal solutions at the expense of increasing the running time.

Table 1
Problem type and its separation problem

Problem		Separation problem
(P_1)	Min cx, s.t. $Ax=b, x\in S$	Min cx, s.t. $x\in S$
(P_2)	Min $\sum_{r=1}^{N} c_r x_r$, s.t. $\sum_{r=1}^{N} A_r x_r = b, x_r \in S_r$	Min $c_r x_r$, s.t. $x_r \in s_r$
(P_3)	Min cx, s.t. $x\in S_1\cap S_2\cap\cdots\cap S_K$	Min cx, s.t. $x\in S_r$
(P_4)	$\mathrm{Max}_{\lambda\geqslant 0}\,\mathrm{Min}_{x\in S}\,[cx+\lambda(b-Ax)]$	Min cx, s.t. $x\in S$
(P_5)	Min $cx+dy$, s.t. $Ax+By\geqslant b, x, y\geqslant 0$	Min cx, s.t. $Ax\geqslant b-By, x\geqslant 0$
(P_6)	$\mathrm{Max}_{\lambda\geqslant 0}\,\mathrm{Min}_{x\in X}\,[f(x)+\lambda g(x)]$	Min $f(x)+\lambda g(x)$ s.t. $x\in X$

Table 2
Running times

Problem		Running time
(P_1)	Min cx, s.t. $Ax=b, x\in S$	$O([T(n)+M(m)]mL)$
(P_2)	Min $\sum_{r=1}^{N} c_r x_r$, s.t. $\sum_{r=1}^{N} A_r x_r = b, x_r \in S_r$	$O([\sum_{r=1}^{N} T_r(n_r)+M(m+N)](m+N)L)$
(P_3)	Min cx, s.t. $x\in S_1\cap S_2\cap\cdots\cap S_K$	$O([\sum_{r=1}^{K} T_r(n)+M(Kn)]KnL)$
(P_4)	$\mathrm{Max}_{\lambda\geqslant 0}\,\mathrm{Min}_{x\in S}\,[cx+\lambda(b-Ax)]$	$O([T(n)+M(m)]mL)$
(P_5)	Min $cx+dy$, s.t. $Ax+By\geqslant b, x, y\geqslant 0$	$O([T(n)+M(m)]mL)$
(P_6)	$\mathrm{Max}_{\lambda\geqslant 0}\,\mathrm{Min}_{x\in X}\,[f(x)+\lambda g(x)]$	$O([T(n)+M(m)]m\log(1/\varepsilon))$

3. Applications

In this section we apply the algorithms of the previous sections to solve LP and Lagrangean relaxations of several classical combinatorial optimization problems. Our intention is not to exhaust all the possible applications of our methods. It is rather to illustrate the significant computational savings that can result from our approach to some of the classical problems in combinatorial optimization. We start our investigation with the traveling salesman problem (TSP).

3.1. *The Held and Karp lower bound for the TSP*

Held and Karp [17] proposed a Lagrangean relaxation based on the notion of 1-tree for the TSP. Using a complete characterization of the 1-tree polytope, they showed that this Lagrangean relaxation gives the same bound as the LP relaxation of the following classical formulation of the TSP ($x_{ij}=x_{ji}$):

$$\text{Min} \sum_{i\in V} \sum_{j\in V,\, j>i} c_{ij}x_{ij} \tag{12}$$

$$\text{subject to} \quad \sum_{j\in V,\, j>i} x_{ij} + \sum_{j\in V,\, j<i} x_{ji} = 2 \quad \forall i\in V, \tag{13}$$

$$\sum_{i\in S} \sum_{j\in S,\, j>i} x_{ij} \leqslant |S| - 1 \quad \forall \emptyset \neq S \subset V, \tag{14}$$

$$0 \leqslant x_{ij} \leqslant 1 \quad \forall i, j\in V, j>i, \tag{15}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j\in V,\ j>i. \tag{16}$$

In the above formulation x_{ij} indicates whether cities i and j are adjacent in the optimal tour; c_{ij} represents the cost of traveling from city i to city j or, by symmetry, from city j to city i. One can compute the Held–Karp lower bound z_{HK} in polynomial time using the Ellipsoid or Vaidya's algorithm directly, since the separation problem corresponding to the polytope (13)–(15) reduces to a max-flow problem. Using this approach Vaidya's algorithm, which is the fastest of the two, will take $O(n^3n^2L+M(n^2)n^2L)$ arithmetic operations, since from (2) there are $k=n^2$ variables and $T(n)=O(n^3)$ is the time to solve a max-flow problem on a possibly complete graph. Thus this approach takes $O(n^{6.76}L)$ where we used the bound that $M(k)=O(k^{2.38})$. Note that the ellipsoid algorithm will be even worse taking $O(n^8L)$.

The HK polytope (13)–(15) is exactly in the form of problem (P_1), where S is the spanning tree polytope described by the constraints (14), (15). Applying Theorem 1 we obtain that using our approach we can compute z_{HK} in $O(n^2nL+M(n)nL)=O(n^{3.38}L)$ arithmetic operations, since the separation problem is a minimum spanning tree computation, i.e., $T(n)=O(n^2)$ and the number of extra constraints is $m=n$. As a result, our approach leads to the fastest known algorithm for HK. Moreover, if one does not use fast matrix multiplication (i.e., $M(k)=k^3$), then our approach leads to $O(n^4L)$ time complexity

while Vaidya's or the Ellipsoid method, applied to the LP relaxation of (12), take $O(n^8L)$, a savings of $O(n^4)$.

Recently, Plotkin, Shmoys and Tardos [30] developed an algorithm that finds an "approximately feasible" element of a set $S=\{Ax\leqslant b,\ x\in P\}$ for some convex set P, assuming that optimization of a linear function over P is efficiently solvable. If $A,\ b\geqslant 0$ then for any specified ε, their algorithm either produces a solution $x\in P$ such that $Ax\leqslant(1+\varepsilon)b$ or else it proves that the set S is empty. Their algorithm is not a polynomial time algorithm, since it grows linearly with the width ρ (which could be exponential) and with ε^{-2}. In certain interesting applications, however, they transformed the problem to an equivalent problem with smaller ρ. As a result, their running time could become exponential in L. For this reason we do not contrast their running times with those of our algorithm.

3.2. The Steiner tree and the 2-connected problem

Goemans and Bertsimas [14] prove that under the triangle inequality, the cost of a standard LP relaxation of the Steiner tree problem z_{Steiner} and the 2-connected problem $z_{k\text{-conn}}$ are related in the following way:

$$z_{\text{Steiner}}=\tfrac{1}{2}z_{\text{HK}},\qquad z_{2\text{-conn}}=z_{\text{HK}},$$

where z_{HK} is the cost of the Held–Karp lower bound for the TSP. Therefore, if the cost satisfy the triangle inequality, we can apply the algorithm of the previous subsection to compute the value of the LP relaxation of the Steiner tree problem and the k-connected problem.

3.3. The vehicle routing problem

Consider the following classical vehicle routing problem: There is a set A of K vehicles, located at a single depot 0, such that the kth vehicle has capacity Q_k and is allowed to travel a distance of at most d_k. These vehicles serve a set V of n customers. Customer i has demand p_i, while c_{ij}^k is the cost of vehicle k traveling from i to j and d_{ij} is the distance from i to j. The goal is to route the vehicles at minimum cost such that all constraints are satisfied. We formulate the problem as follows: Let x_{ij}^k be 1 if vehicle k travels from i to j and 0 otherwise. Let S_k be the following polytope:

$$S_k=\left\{x_{ij}^k\ \middle|\ \sum_{i\in S}\sum_{j\in S,\ j>i}x_{ij}^k\leqslant|S|-1\ \forall\emptyset\neq S\subset V,\ \sum_{i\in V}x_{0i}^k+\sum_{j\in V}x_{j0}^k=2\right\}.$$

The polytope S_k is the intersection of the spanning tree polytope on V (note that V does not include the depot 0) and an additional constraint that 2 additional arcs are incident to the depot. For fixed k we denote all the x_{ij}^k's as the vector x^k. We are interested to compute the LP relaxation of the following formulation of VRP:

$$\text{(VRP) Min} \sum_{i,\,j,\,k} c_{ij}^k x_{ij}^k \tag{17}$$

$$\text{subject to} \quad \sum_{i\in V} \sum_{k\in A} x_{ij}^k = 1 \quad \forall j \neq 0, \tag{18}$$

$$\sum_{j\in V} \sum_{k\in A} x_{ij}^k = 1 \quad \forall i \neq 0, \tag{19}$$

$$\sum_{i,\,j\in V} x_{ij}^k d_{ij} \leqslant d_k \quad \forall k \in A, \tag{20}$$

$$\sum_{i,\,j\in V} x_{ij}^k p_i \leqslant Q_k \quad \forall k \in A, \tag{21}$$

$$x^k \in S_k, \tag{22}$$

$$0 \leqslant x_{ij}^k \leqslant 1, \tag{23}$$

$$x_{ij}^k \in \{0, 1\}. \tag{24}$$

In order to solve the LP relaxation of the above problem we will apply our approach of Section 2.2. The above formulation is of the type of problems (P_2), where there are $N=K$ subproblems S_k, and the number of additional constraints (18)–(21) is $m=2(n+K)$. Since we can optimize over S_k using the greedy algorithm, the time to solve the separation problem is $O(n^2)$. Thus, applying Theorem 2, we can solve the LP relaxation of (VRP) in

$$O(Kn^2(n+K)L+M(n+K)(n+K)L) = O((n^3K+n^{3.38})L),$$

since we can assume $K \leqslant n$.

For comparison if we applied Vaidya's algorithm directly to (17) it would lead to an

$$O(Kn^3(Kn^2)L+M(Kn^2)(Kn^2)L) = O(K^{3.38}n^{6.76}L)$$

algorithm, since there are Kn^2 variables and the separation problem reduces to K max-flow problems.

3.4. Multicommodity flows

Consider the classical multicommodity flow problem: Given a network $G=(V, E)$ ($|V|=n$, $|E|=m$), a set C of K commodities and a supply (or demand) b_i^k of commodity k at node $i \in V$. Arc (i, j) has capacity u_{ij} and the cost of sending one unit of flow of commodity k across arc (i, j) is c_{ij}^k. The goal is to decide the amount of flow x_{ij}^k from commodity k to send across arc (i, j), so as to satisfy supply-demand and capacity constraints at minimum cost. The classical formulation of the problem is:

$$\text{Min} \sum_{i,\, j\in V,\ k\in C} c_{ij}^k x_{ij}^k \tag{25}$$

$$\text{subject to} \quad \sum_{k\in C} x_{ij}^k \leqslant u_{ij}, \quad (i,j)\in E, \tag{26}$$

$$\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = b_i^k, \quad i\in V,\ k\in C, \tag{27}$$

$$x_{ij}^k \geqslant 0. \tag{28}$$

The above formulation is of the type (P_2) in Section 2.2, with

$$S_k = \left\{ x_{ij}^k \;\middle|\; x_{ij}^k \geqslant 0,\ \sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = b_i^k,\ i\in V \right\}$$

and with m global constraints (26). In this case the separation problem is a min-cost flow problem. Under the similarity assumption, the time to solve the separation problem is $T = O(nm \log^2 n)$. For a more refined definition of T see, for example, Ahuja et al. [1].

As a result, applying Theorem 2 we can solve the multicommodity flow problem in

$$O([Kmn \log^2 n + M(m+K)](m+K)L) = O([Knm^2 \log^2 n + m^{3.38}]L).$$

The comparison for multicommodity flows is more complex because of another algorithm of Vaidya [32]. He considered the problem in which each commodity consists of a single source and a single sink. The resulting running time is $O(K^{2.5} n^2 \sqrt{m}\, L)$. This running time dominates ours in some cases and is dominated by ours in others. For example, for $k = n^2$ Vaidya's running time is $O(n^7 \sqrt{m}\, L)$. However, our algorithm runs in $O([n^2 m^2 \log^2 n + m^{3.38}]L)$ time in this case, since the problem can be converted into a n-commodity flow problem. In this case, our algorithm dominates Vaidya's for all values of m, and is increasingly better as the graph becomes sparser.

3.5. Network flows with side constraints

Typical problems met in practice involve network flow problems with some additional side constraints. Consider for example a min-cost flow problem with K additional constraints. Using these K constraints as the global constraints in the formulation of problem (P_1) in Section 2.1 and the network flow polytope

$$\left\{ x_{ij} \;\middle|\; x_{ij} \geqslant 0,\ \sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = b_i,\ i\in V \right\}$$

as the polytope S we can apply Theorem 1 to solve the problem in $O([mn \log^2 n + M(K)]KL)$, where $T = mn \log^2 n$ is an upper bound on the complexity to solve the network flow problem under the similarity assumption (see [1]). For K constant we see that the complexity of solving the problem *with K side constraints* is exactly the same as the complexity of solving the problem *with only one side constraint*, i.e., $O((mn \log^2 n)L)$.

If one applied an interior point algorithm for this problem, it would lead to an $O(m^3L)$ running time.

3.6. The network design problem

The fixed charge network design problem, which is a fundamental discrete choice design problem, is useful for a variety of applications in transportation, distribution, communication, and several other problem settings that make basic cost tradeoffs between operating costs and fixed costs for providing network facilities (see Magnanti and Wong [26]). The problem is described as follows. We are given a set of nodes N, a set of uncapacitated arcs A and a set K of commodities. For each $k \in K$, one unit of flow of commodity k must be sent from its origin $O(k)$ to its destination $D(k)$. Each arc has two types of cost: a per unit flow cost depending on the commodity and a fixed charge for using the arc. The problem is to select a subset of arcs that minimizes the sum of the routing costs and fixed charge costs.

The importance of the network design problem stems from its wide applicability and flexibility. As noted in Magnanti and Wong [26], it contains a number of well-known network optimization problems as special cases including the shortest path, minimum spanning tree, uncapacitated plant location, traveling salesman and Steiner tree problems.

There are a number of IP formulations for the problem. For a review, see Magnanti and Wong [26]. Balakrishnan et al. [3] propose the following multicommodity flow formulation, which contains two types of variables, one modeling discrete design choices and the other continuous flow decisions. Let y_{ij} be a binary variable that indicates whether ($y_{ij}=1$) or not ($y_{ij}=0$) arc $\{i, j\}$ is chosen as part of the network's design. Let x_{ij}^k denote the flow of commodity k on the directed arc (i, j). Note that (i, j) and (j, i) denote directed arcs with opposite orientations corresponding to the undirected arc $\{i, j\}$. Even though arcs in the formulation are undirected, we refer to the directed arcs (i, j) and (j, i) because the flows are directed. The formulation is the following.

$$\text{Min} \sum_{k \in K} \sum_{\{i, j\} \in A} (c_{ij}^k x_{ij}^k + c_{ji}^k x_{ji}^k) + \sum_{\{i, j\} \in A} F_{ij} y_{ij} \tag{29}$$

$$\text{subject to} \sum_{j \in V:\ (j, i) \in A} x_{ji}^k - \sum_{j \in V:\ (i, j) \in A} x_{ij}^k = \begin{cases} -1 & i = O(k), \\ 1 & i = D(k), \\ 0 & \text{otherwise}, \end{cases} \tag{30}$$

$$(\text{DP}_1)\quad x_{ij}^k \leqslant y_{ij}, \quad x_{ji}^k \leqslant y_{ij}, \qquad \{i, j\} \in A,\ k \in K,$$

$$x_{ij}^k \geqslant 0, \quad \{i, j\} \in A,\ k \in K, \tag{31}$$

$$y_{ij} \in \{0, 1\}, \quad (i, j) \in A.$$

In this formulation each arc $\{i, j\}$ has a nonnegative fixed design cost F_{ij} and c_{ij}^k is the nonnegative cost for routing commodity k on the directed arc (i, j). Constraints (30) imposed upon each commodity k are the usual network flow conservation equations. The "forcing" constraints (31) state that if $y_{ij}=0$, i.e., arc $\{i, j\}$ is not included in the design,

then the flow of every commodity k on this arc must be zero in both directions, and if arc $\{i,j\}$ is included in the design, i.e., $y_{ij}=1$, the arc flow is unlimited. Directed network design problems are formulated in a very similar manner.

Although the network design problem is a hard discrete optimization problem, in the last decade researchers have proposed several computationally successful approaches for solving it. Magnanti et al. [24] propose a Benders decomposition approach, and Balakrishnan et al. [3] propose a dual ascent heuristic which has solved large instances of network design problems to within 2%–5% of optimality. In both these cases the authors judge the algorithm's effectiveness by comparing solutions generated to the LP relaxation of their formulations. It is therefore important to solve the LP relaxation efficiently.

We treat the forcing variables y_{ij} in the network design formulation (29) as the complicating variables in the sense of Section 2.5. If the y_{ij} are known then the problem decomposes to K shortest paths problems. Therefore, applying Theorem 5, the LP relaxation of (29) can be solved in

$$O([K(m+n\log n)+M(m)]mL),$$

where m is the number of complicating variables y_{ij}, and $O(m+n\log n)$ is the time to solve a single shortest path problem. Note that for the problems considered in [3] $K=n^2$ and $m=O(n)$ and thus our algorithm takes $O([n^2(m+n\log n)]mL)$.

For purposes of comparison if one solves the LP relaxation of (29) using an interior point algorithm, it will lead to an $O(K^3m^3L)$ running time.

3.7. Facility location problems

We consider the well known p-median problem in facility location (see for example [27]). We are interested in solving the LP relaxation of the following formulation of the problem:

$$\text{Min} \sum_{i\in V} c_{ij}x_{ij} + \sum_{j\in V} y_j \tag{32}$$

$$\text{subject to} \quad \sum_{j\in V} y_j = p, \tag{33}$$

$$\sum_{j\in V} x_{ij} = 1, \quad i\in V, \tag{34}$$

$$x_{ij} \leqslant y_j, \quad i,j\in V, \tag{35}$$

$$x_{ij} \geqslant 0, \qquad 0\leqslant y_j \leqslant 1, \tag{36}$$

$$y_j \in \{0,1\}. \tag{37}$$

The interpretation is that y_j is 1 if node j is assigned to a facility and 0 otherwise and x_{ij} is 1 if node i is assigned to a facility j and 0 otherwise. The LP relaxation of (32) has been found to be very close to the IP solution. For this reason almost all algorithmic approaches to the problem compute first the LP relaxation.

In order to solve the LP relaxation we observe that the p-median problem is of the type

of problem (P_1) with constraints (33) and (34) being the global constraints and (35) and (36) being the polytope S. We can solve the separation problem over S in $T(n)=O(n^2)$ ($|V|=n$), since we can solve it in closed form in one pass. Therefore, applying Theorem 1 we can solve the LP relaxation of the p-median problem in $O(n^2nL+M(n)nL)=O(n^{3.38}L)$. For comparison purposes, if one applied an interior point algorithm directly to solve the LP relaxation of (32), it would take $O(n^6L)$ iterations since there are $O(n^2)$ variables.

For uncapacitated location problems (see for example [27]) exactly the same approach as in the case of the p-median problem leads to an $O(n^{3.38}L)$ algorithm for the solution of the LP relaxation.

3.8. The K-polymatroid intersection problem

Consider K polymatroids $M_i=(N, f_i)$ where f_i is a submodular set function. For example, if f_i is the rank function of a matroid, the problem reduces to the K-matroid intersection problem. Given costs c_j for all $j\in N$, the weighted K-polymatroid intersection problem is described by the mathematical programming problem:

$$\text{Max} \sum_{j\in N} c_j x_j \tag{38}$$

$$\text{subject to} \quad \sum_{j\in S} x_j \leqslant f_1(S) \quad \forall S\subset N, \tag{39}$$

$$\vdots$$

$$\sum_{j\in S} x_j \leqslant f_K(S) \quad \forall S\subset N, \tag{40}$$

$$x_j \geqslant 0.$$

Classical problems in combinatorial optimization can be modelled in that way. For example the maximum spanning tree ($K=1$ and $f_1(S)=|S|-1$), maximum bipartite matching ($K=2$ and $f_i(S)$ are the rank functions of two partitioned matroids). Our goal is to solve (38) using the techniques of Section 2.3.

Let $S_i=\{x>0 \mid \sum_{j\in S} x_j \leqslant f_i(S)\ \forall S\subset N\}$. Using the cost splitting method of Section 2.3 and applying Theorem 3 we can solve problem (38) in $O([KT(n)+M(Kn)]KnL)$, where $T(n)$ is the number of arithmetic operations to solve the optimization problem over one polymatroid, which, as it is well known (see for example Nemhauser and Wolsey [27, p. 689]), can be solved by the greedy algorithm as follows:

Step 1. Sort $c_1\geqslant c_2\geqslant\cdots\geqslant c_k>0\geqslant c_{k+1}\geqslant\cdots\geqslant c_n$.
Step 2. Let $S^j=\{1, \ldots, j\}$ with $S^0=\emptyset$.
Step 3. $x_j=f(S^j)-f(S^{j-1})$ for $j\leqslant k$ and $x_j=0$ for $j>k$.

For purposes of comparison, an alternative approach is to apply Vaidya's algorithm to (38) directly. The separation problem is to decide whether a given $x_0 \in S_i$. Such an approach leads to a $O([KA(n)+M(n)]nL)$ arithmetic operations where $A(n)$ is the number of arithmetic operations to solve the separation problem $x_0 \in S_i$. Indeed, $A(n) = O(nL[T(n)+M(n)])$ if we use Vaidya's algorithm. Overall this approach leads to $O(K(nL)^2[T(n)+M(n)])$ arithmetic operations. Alternatively in order to solve the separation problem, one could use the strongly polynomial algorithm of Cunningham [10], but unfortunately the running time would not be as good.

Overall, we expect that our approach will work better, in problems in which the separation problem is much harder than the optimization problem.

3.9. Network flow problems on graphs with almost special structure

Consider a network flow problem on the network $G=(N, A \cup B)$ such that $G_A=(N, A)$ is a graph of special structure (for example planar, tree, unbalanced bipartite, etc.) for which the related network flow problem can be solved faster than in a general graph. For example, for network flow problems on unbalanced bipartite graphs see [2]. Assume that $|B|=K$. The network flow problem can be formulated as follows:

$$z = \text{Min } c_A x_A + c_B x_B \tag{41}$$

$$\text{subject to} \quad N_A x_A + N_B x_B = b, \tag{42}$$

$$0 \leqslant x_A \leqslant u_A, \tag{43}$$

$$0 \leqslant x_B \leqslant u_B, \tag{44}$$

where N_A, N_B is the arc incidence matrix corresponding to the arcs in A and in B respectively.

We first observe that problem (41) is of the type of problem (P_5) in Section 2.5. Applying the variable splitting algorithm of Section 2.5, we obtain that problem (41) can be solved in $O([T_A(n,m)+M(K)]KL)$ where $T_A(n,m)$ is the time to solve the network flow problem on $G_A=(N, A)$, which is a graph of special structure. Because of the special structure of G_A, $T_A(n, m)$ will be smaller than the time to solve the problem on general graphs. For K constant the running time becomes $O(T_A(n,m)L)$. For comparison purposes we will denote the running time on general graphs as $O(T_G(n, m))$.

3.10. The multiple item capacitated lot sizing problem

Consider the multiple item capacitated lot sizing problem, where x_{jt}, y_{jt} and s_{jt} represent the production, setup and storage variables for item j in period t, d_{jt}, c_{jt}, f_{jt}, h_{jt} are the demand, production cost, setup cost and storage cost for item j in period t and Q_t represents the amount of the resource available in period t:

$$\text{Min} \sum_{j=1}^{K} \sum_{t=1}^{T} [c_{jt}x_{jt} + f_{jt}y_{jt} + h_{jt}s_{jt}] \tag{45}$$

$$\text{subject to} \quad x_{jt} + s_{j,\,t-1} = d_{jt} + s_{jt}, \quad j=1, \ldots, K, \ t=1, \ldots, T, \tag{46}$$

$$s_{j0} = s_{jT} = 0, \quad j=1, \ldots, K, \tag{47}$$

$$x_{jt} \leqslant \left(\sum_{r=t}^{T} d_{jr} \right) y_{jt}, \quad j=1, \ldots, K, \ t=1, \ldots, T, \tag{48}$$

$$\sum_{j=1}^{K} x_{jt} \leqslant Q_t, \quad t=1, \ldots, T, \tag{49}$$

$$x_{jt}, s_{jt} \geqslant 0, \qquad y_{jt} \in \{0, 1\}, \tag{50}$$

The multiple item capacitated lot sizing problem is N P-hard. If we relax constraints (49) the problem decomposes into K single item capacitated lot sizing problems, each of which can be solved by a dynamic programming algorithm, that has $O(T)$ running time. Applying the algorithm of Section 2.4 we can find the value of the Lagrangean dual in

$$O([KT + M(T)]TL) = O([KT^2 + T^{3.38}]L).$$

Note that the value of the Lagrangean dual can be strictly better that the value of the LP relaxation, since the subproblem does not have the integrality property. To the best of our knowledge we do not know any other polynomial time approach for the problem. For comparison, the solution of the LP relaxation of (45), which gives a weaker bound than the Lagrangean dual, takes $O(K^3T^3L)$ using an interior point approach.

3.11. Stochastic programming

In two-stage stochastic programming there are two kinds of decision variables. Decisions represented by variables x_1, which need to be taken now and decisions represented by variables x_2, which are taken after the occurrence of a random event. The two-stage stochastic programming problem is often defined as follows:

$$z_{SP} = \text{Min } c_1 x_1 + E[f(x_1, \xi)]$$
$$\text{subject to} \quad x_1 \in X = \{A_1 x_1 = b_1, \ x_1 \in \mathbb{R}^{n_1}_+\},$$

where

$$f(x_1, \xi) = \text{Min } c_2 x_2$$
$$\text{subject to} \quad B_1 x_1 + B_2 x_2 = \xi,$$
$$x_2 \in \mathbb{R}^{n_2}_+.$$

Here, we further assume that the rhs ξ is a random vector which has a discrete probability distribution, i.e. $\Pr\{\xi = \xi_r\} = p_r, r = 1, \ldots, K$, so $E[f(x_1, \xi)] = \sum_{r=1}^{K} p_r f(x_1, \xi_r)$. The problem can then be reformulated as follows.

[10] W. Cunningham, "Testing membership in matroid polyhedra," *Journal of Combinatorial Theory Series B* 36 (1984) 161–188.
[11] G.D. Eppen and R.K. Martin, "Solving multi-item capacitated lot sizing problems using variable redefinition," *Operations Research* 35 (1987) 832–848.
[12] M. Fisher, "The Lagrangean relaxation method for solving integer programming problems," *Management Science* 27 (1981) 1–18.
[13] A. Geoffrion, "Lagrangean relaxation and its uses in integer programming," *Mathematical Programming Study* 2 (1974) 82–114.
[14] M. Goemans and D. Bertsimas, "Survivable networks, linear programming relaxations and the parsimonious property," *Mathematical Programming* 60 (1993) 145–166.
[15] M. Grötschel, M. Jünger and G. Reinelt, "A cutting plane algorithm for the linear ordering problem," *Operations Research* 32 (1984) 1195–1220.
[16] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization* (Springer, Berlin, 1988).
[17] M. Held and R.M. Karp, "The traveling-salesman problem and minimum spanning trees," *Operations Research* 18 (1970) 1138–1162.
[18] M. Held and R.M. Karp, "The traveling-salesman problem and minimum spanning trees: Part II," *Mathematical Programming* 1 (1971) 6–25.
[19] M. Held, P. Wolfe and H. Crowder, "Validation of subgradient optimization," *Mathematical Programming* 6 (1974) 62–88.
[20] E.L. Johnson, M.M. Kostreva and H.H. Suhl, "Solving 0–1 integer programming problems arising from large scale planning models," *Operations Research* 33 (1985) 802–820.
[21] K. Jornsten and M. Nasberg, "A new Lagrangean relaxation approach to the generalized assignment problem," *European Journal of Operations Research* 27 (1986) 313–323.
[22] L.G. Khachian, "A polynomial algorithm for linear programming," *Soviet Mathematics Doklady* 20 (1979) 191–194.
[23] J. Leung, T.L. Magnanti and R. Vachani, "Facets and algorithms for capacitated lot sizing," *Mathematical Programming* 45 (1989) 331–359.
[24] T.L. Magnanti, P. Mireault and R.T. Wong, "Tailoring Benders decomposition for uncapacitated network design," *Mathematical Programming Study* 26 (1986) 112–154.
[25] T.L. Magnanti and R. Vachani, "A strong cutting plane algorithm for production scheduling with changeover costs," working paper OR173-87, Operations Research Center (1989).
[26] T.L. Magnanti and R.T. Wong, "Network design and transportation planning: Models and algorithms," *Transportation Science* 18 (1984) 1–56.
[27] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization* (Wiley, New York, 1985).
[28] M.W. Padberg and S. Hong, "On the symmetric traveling salesman problem: a computational study," *Mathematical Programming Study* 12 (1980) 78–107.
[29] M.W. Padberg and G. Rinaldi, "Optimization of a 532-city symmetric traveling salesman problem by branch and cut," *Operations Research Letters* 6 (1987) 1–7.
[30] S. Plotkin, D. Shmoys and E. Tardos, "Fast approximation algorithms for fractional packing and covering problems," extended abstract.
[31] J. Shapiro, *Mathematical Programming, Structures and Algorithms* (Wiley, New York, 1979).
[32] P. Vaidya, "Speeding-up linear programming using fast matrix multiplication," *Proceedings of the 30th Symposium on the Foundations of Computer Science* (1989) 332–337.
[33] P. Vaidya, "A new algorithm for minimizing convex functions over convex sets," AT&T Bell Laboratories (Murray Hill, NJ, 1990).