SIAM J. OPTIM.
Vol. 16, No. 3, pp. 726–750

SOLVING LIFT-AND-PROJECT RELAXATIONS OF BINARY INTEGER PROGRAMS*

SAMUEL BURER† AND DIETER VANDENBUSSCHE‡

Abstract. We propose a method for optimizing the lift-and-project relaxations of binary integer programs introduced by Lovász and Schrijver. In particular, we study both linear and semidefinite relaxations. The key idea is a restructuring of the relaxations, which isolates the complicating constraints and allows for a Lagrangian approach. We detail an enhanced subgradient method and discuss its efficient implementation. Computational results illustrate that our algorithm produces tight bounds more quickly than state-of-the-art linear and semidefinite solvers.

Key words. integer programming, lift-and-project relaxations, semidefinite programming, augmented Lagrangian

AMS subject classifications. 90C10, 90C22, 90C27, 90C30

DOI. 10.1137/040609574

1. Introduction.

In the field of optimization, binary integer programs have proven to be an excellent source of challenging problems, and the successful solution of larger and larger problems over the past few decades has required significant theoretical and computational advances. One of the fundamental issues is how to obtain a "good" description of the convex hull of integer solutions, and many specific classes of integer programs have been solved by finding problem-specific ways to address this issue.

Researchers have also developed techniques for approximating the convex hull of integer solutions without any specific knowledge of the problem, i.e., techniques that apply to arbitrary binary integer programs. Some of the earliest work done in this direction was by Gomory [19] in generating linear inequalities that tighten the basic linear relaxation. A different idea, which has been advocated by several authors, is to approximate the convex hull as the projection of some polyhedron lying in a space of higher dimension. We refer the reader to [3, 38, 32, 4, 24, 7]. Connections between these works are explored in [27, 26].

Although these so-called *lift-and-project* methods are quite powerful theoretically, they present great computational challenges because one typically must optimize in the space of the lifting, i.e., the space of higher dimension. Computational issues are detailed in [4, 39, 11, 22, 14].

In this paper, we focus on the techniques proposed by Lovász and Schrijver (LS), including both linear and semidefinite relaxations. In particular, our main goal is to present improved computational methods for optimizing over the first-level LS relaxations. We are aware of only one study (by Dash [14]), which investigates the strength of these relaxations computationally. This shortage of computational experience is due to the dramatic size of these relaxations. For example, one specific semidefinite

*Received by the editors June 7, 2004; accepted for publication (in revised form) June 17, 2005; published electronically January 6, 2006.
http://www.siam.org/journals/siopt/16-3/60957.html
†Department of Management Sciences, University of Iowa, Iowa City, IA 52242-1000 (samuel-burer@uiowa.edu). This author was supported in part by NSF grant CCR-0203426.
‡Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (dieterv@uiuc.edu).

relaxation that has been considered by Dash (and which we also consider in this paper) has well over 1.7 million constraints.

Since it is unlikely that these relaxations can be solved using direct algorithms, we instead adopt the paradigm of decomposition, which is common in large-scale optimization methods. (Dash [14] also considers a decomposition approach.) The main idea here is a clever decomposition of the LS relaxations, which allows for a Lagrangian approach. Instead of using the common subgradient algorithm, however, we propose to use an augmented Lagrangian algorithm, which is, in some sense, an enhanced subgradient method and which also has connections with the bundle method for convex optimization. We provide a theoretical explanation of the benefits of the augmented Lagrangian algorithm and give a detailed explanation of our implementation, which is demonstrated to outperform state-of-the-art subgradient, linear, and semidefinite solvers on certain classes of problems.

We remark that, while the idea of using the augmented Lagrangian method for linear programs is not new (see [35, 40]), little work has been done on the computational aspects of such a method. In this paper, we fill this gap concerning large-scale linear programs and also present an augmented Lagrangian method for semidefinite programs for the first time.

The paper is organized as follows. In section 2, we give background on the Lovász-Schrijver lift-and-project relaxations as well as propose the decomposition technique that will become the basis of our augmented Lagrangian algorithm. Then, in section 3, we discuss the augmented Lagrangian algorithm, including its theoretical benefits and specialization in the current context. Next, in section 4, we present the details of our implementation and computational results. We also discuss the strength of the LS relaxations on various problem classes, with one highlight being that, in practice, the LS semidefinite relaxation provides the strongest known bounds for a collection of problems in the Quadratic Assignment Problem Library [36]. Finally, we conclude with a few final remarks and suggestions for future research in section 5.

1.1. Notation and terminology. In this section, we introduce some of the notation that will be used throughout the paper. $\mathbb{R}^n$ will refer to n-dimensional Euclidean space. The norm of a vector $x \in \mathbb{R}^n$ is denoted by $\|x\| := \sqrt{x^T x}$. We let $e_i \in \mathbb{R}^n$ represent the ith unit vector, and e is the vector of all ones. $\mathbb{R}^{n\times n}$ is the set of real $n \times n$ matrices, $\mathcal{S}^n$ is the set of symmetric matrices in $\mathbb{R}^{n\times n}$, while $\mathcal{S}^n_+$ is the set of positive semidefinite symmetric matrices. The special notation $\mathbb{R}^{1+n}$ and $\mathcal{S}^{1+n}$ is used to denote the spaces $\mathbb{R}^n$ and $\mathcal{S}^n$ with an additional "zeroth" entry prefixed or an additional zeroth row and zeroth column prefixed, respectively. The inner product of two matrices $A, B \in \mathbb{R}^{n\times n}$ is defined as $A \bullet B := \operatorname{tr}(A^T B)$, where $\operatorname{tr}(\cdot)$ denotes the sum of the diagonal entries of a matrix. The Frobenius norm of a matrix $A \in \mathbb{R}^{n\times n}$ is defined as $\|A\|_F := \sqrt{A \bullet A}$. $\operatorname{diag}(A)$ is defined as the vector with the diagonal of A as its entries.

2. The lift-and-project operators of Lovász and Schrijver. When solving a 0-1 integer program of the form

$$\min \left\{ c^T x \mid Ax \le b,\ x \in \{0,1\}^n \right\}, \tag{IP}$$

we are often interested in relaxations of the convex hull of integer solutions

$$P := \operatorname{conv} \{ x \in \{0,1\}^n \mid Ax \le b \}.$$

Optimization over such relaxations provides lower bounds that can be used within branch-and-bound methods or allow one to assess the quality of feasible solutions

of (IP). The trivial linear programming (LP) relaxation is obtained by replacing $x \in \{0,1\}^n$ with $x \in [0,1]^n$. In an effort to develop relaxations that are stronger than the LP relaxation, Lovász and Schrijver [32] introduced the lifted matrix variable

$$Y = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^T = \begin{pmatrix} 1 & x^T \\ x & xx^T \end{pmatrix}.$$

Given this definition relating Y and $x \in \{0,1\}^n$, we can observe a number of interesting properties of Y:

1. Y is symmetric and positive semidefinite, i.e., $Y \in \mathcal{S}^{1+n}_+$;
2. the diagonal of Y equals the zeroth column of Y, i.e., $\operatorname{diag}(Y) = Ye_0$;
3. if we multiply the constraints $Ax \le b$ of P by some x_i, we obtain the set of nonlinear inequalities $bx_i - Axx_i \ge 0$, which are valid for P; these inequalities can be written in terms of Y as

$$\big(b\big| - A\big)\, Ye_i \ge 0 \quad \forall\ i = 1, \ldots, n;$$

4. analogously, multiplying $Ax \le b$ by $1 - x_i$ yields

$$\big(b\big| - A\big)\, Y(e_0 - e_i) \ge 0 \quad \forall\ i = 1, \ldots, n.$$

Lovász and Schrijver [32] observed that these properties could be used to obtain relaxations of P. In particular, they homogenized the standard LP relaxation of (IP) by defining

$$K := \left\{ \begin{pmatrix} x_0 \\ x \end{pmatrix} \in \mathbb{R}^{1+n} \mid Ax \le x_0 b,\ 0 \le x \le x_0 e \right\}.$$

We remark that enforcing $x_0 = 1$ in K yields the LP relaxation and that the third and fourth properties above can be written as $Ye_i \in K$ and $Y(e_0 - e_i) \in K$. They then proposed the following sets:

$$M(K) := \big\{ Y \in \mathcal{S}^{1+n} \mid Ye_0 = \operatorname{diag}(Y), \quad Ye_i \in K,\ Y(e_0 - e_i) \in K \ \ \forall\ i = 1, \ldots, n \big\}$$
$$M_+(K) := \big\{ Y \in \mathcal{S}^{1+n}_+ \mid Y \in M(K) \big\}.$$

Note that $M_+(K)$ differs from $M(K)$ only in that positive semidefiniteness is enforced on the Y variable. $M(K)$ now leads to a linear relaxation of P via the projected set

$$N(K) := \left\{ x \in \mathbb{R}^n \mid \begin{pmatrix} 1 \\ x \end{pmatrix} = \operatorname{diag}(Y) \ \text{ for some } \ Y \in M(K) \right\},$$

and $M_+(K)$ leads to an analogous semidefinite relaxation $N_+(K)$ of P. In particular, Lovász and Schrijver [32] showed that $P \subseteq N_+(K) \subseteq N(K)$ and that $N(K)$ is contained in the LP relaxation of (IP). Further, they showed that applying these relaxation procedures iteratively n times yields P exactly.

We remark that our definitions of $N(K)$ and $N_+(K)$ are actually slices (at $Y_{00} = 1$ and $x_0 = 1$) of the cones originally defined by Lovász and Schrijver [32].

Applying these ideas to the stable set problem, Lovász and Schrijver proved that some classes of inequalities for the stable set polytope are satisfied by all the points in $N(K)$, while other classes are only valid for $N_+(K)$. In turn, these results have significant implications for the complexity of finding maximum stable sets in various classes of graphs. Further, theoretical results concerning the strength of $N(K)$ and

$N_+(K)$, as well as the higher-order liftings, have also been established; see [41, 12, 18, 25, 30].

To compute lower bounds available from the relaxations $N(K)$ and $N_+(K)$, one must solve the LP

$$\min \left\{ c^T x \mid x \in N(K) \right\} \tag{1}$$

or the semidefinite program (SDP)

$$\min \left\{ c^T x \mid x \in N_+(K) \right\}. \tag{2}$$

Defining $\tilde{c} := \begin{pmatrix} 0 \\ c \end{pmatrix}$ and using the above definitions, (1) can be written explicitly as

$$\begin{aligned}
&\min && \tilde{c}^T Y e_0 && && (3)\\
&\text{s.t.} && Y = Y^T, && && (4)\\
& && Y e_0 = \operatorname{diag}(Y), && && (5)\\
& && Y e_i \in K && \forall\ i = 1, \ldots, n, && (6)\\
& && Y(e_0 - e_i) \in K && \forall\ i = 1, \ldots, n, && (7)\\
& && Y_{00} = 1. && && (8)
\end{aligned}$$

Likewise, (2) can be written as

$$\begin{aligned}
&\min && \tilde{c}^T Y e_0\\
&\text{s.t.} && Y \in \mathcal{S}_+^{1+n}\\
& && (4)\text{–}(8).
\end{aligned}$$

A couple of comments concerning (1) and (2) are in order. First, the constraints (6) and (7) imply $Ye_0 \in K$. Combined with (8), this in turn implies that each component of the zeroth column of Y is in $[0, 1]$. By (4), the same holds for the zeroth row of Y, which implies by (6) that all other components of Y are in $[0, 1]$. Hence, we may replace K with $\hat{K} := K \cap [0, 1]^{1+n}$ without affecting the optimal solution sets of (1) and (2).

Second, if K is defined by m constraints, including upper and lower bounds, then the LP described by (3)–(8) has $\mathcal{O}(n^2 + nm)$ constraints and $\mathcal{O}(n^2)$ variables. Consequently, solving this LP using, say, the simplex method or interior-point methods becomes too cumbersome even for problems with moderate n and m. This situation is further exacerbated when solving (2). In fact, using standard SDP methods, (2) will only be solvable for very small n and m. As a result, very little research has been done on actually solving (1) and (2). Work involving (1) is discussed in [39] and some computations of (2) for various 0-1 polytopes can be found in [14].

This second observation motivates us to investigate new optimization techniques for solving (1) and (2), and, in particular, we are interested in applying decomposition methods for large-scale optimization. We first show how this can be done for (1). Unfortunately, all the constraints (4)–(8) are tightly linked, and so the problem does not immediately lend itself to decomposition. To partially overcome this obstacle, however, we introduce the matrix variable

$$Z = YQ \in \mathbb{R}^{(1+n)\times n}, \qquad \text{where} \quad Q := \left(e_0 - e_1 \middle| e_0 - e_2 \middle| \cdots \middle| e_0 - e_n \right)$$

and reformulate (3)–(8) as

$$\begin{aligned} &\min && \tilde{c}^T Y e_0 \\ &\text{s.t.} && Y = Y^T, \quad Y e_0 = \operatorname{diag}(Y), \quad Z = YQ \qquad (9) \\ & && Y e_i \in \hat{K}, \quad Z e_i \in \hat{K} \quad \forall\, i = 1, \ldots, n \qquad (10) \\ & && Y_{00} = 1. \qquad (11) \end{aligned}$$

Note that K has been replaced with $\hat{K}$ in accordance with the first comment above. Furthermore, it is clear that the constraints (10) are separable over the columns of Y and Z but that these same columns are linked via the constraints (9).

A reasonable idea is to apply Lagrangian relaxation to the constraints (9), and in order to simplify notation, we denote (9) by the collection of linear equations $h(Y, Z) = 0$. Letting λ denote the vector of unconstrained dual multipliers for $h(Y, Z) = 0$, we obtain the Lagrangian relaxation

$$\begin{aligned} L(\lambda) := \min \quad & \tilde{c}^T Y e_0 + \lambda^T h(Z, Y) \\ \text{s.t.} \quad & Y e_i \in \hat{K} \quad \forall\, i = 0, 1, \ldots, n \qquad (12) \\ & Z e_i \in \hat{K} \quad \forall\, i = 1, \ldots, n \qquad (13) \\ & Y_{00} = 1. \qquad (14) \end{aligned}$$

Note that we have added the constraint $Y e_0 \in K$, which is redundant for (9) and (10) but is included here in order to properly constrain the zeroth column of Y. It is now clear that $L(\lambda)$ is separable over the columns of Y and Z, and so to evaluate $L(\lambda)$ for any λ, we can simply solve $2n + 1$ separate linear optimizations over $\hat{K}$ (while respecting the simple constraint $Y_{00} = 1$). Furthermore, from standard LP theory, we know that the optimal value of

$$\max_{\lambda} \quad L(\lambda) \qquad (15)$$

equals the optimal value of (1).

The semidefinite optimization (2) can be approached in a similar fashion, i.e., by introducing the auxiliary variable Z and then relaxing the linking constraints. However, we must also introduce a dual multiplier $S \in \mathcal{S}_+^{1+n}$ for the constraint that keeps Y positive semidefinite, which modifies the Lagrangian relaxation to read

$$\begin{aligned} L(\lambda, S) := \min \quad & \tilde{c}^T Y e_0 + \lambda^T h(Z, Y) - S \bullet Y \\ \text{s.t.} \quad & (12)\text{–}(14), \end{aligned}$$

so that the resulting Lagrangian optimization is

$$\sup_{\lambda, S} \left\{ L(\lambda, S) \mid S \in \mathcal{S}_+^{1+n} \right\}. \qquad (16)$$

It is well known that the dual SDP of (2) has an interior-point, i.e., it satisfies Slater's condition, which implies that there is no duality gap and that optimality is attained in (2)—although optimality may not be attained in the dual of (2). As a result, it is not difficult to see that the optimal value of (16) equals that of (2).

Theoretically, one can solve both (15) and (16) using a subgradient method. Our initial experiments, however, indicated that convergence of subgradient methods was sometimes slow. This motivated us to examine an augmented Lagrangian method, which overall proved to be more robust while still allowing us to exploit the structure inherent in (1) and (2). We discuss these issues in detail in sections 3 and 4.

3. The augmented Lagrangian method for linear conic programs. In this section, we discuss the specialization of the augmented Lagrangian method—a standard tool of nonlinear programming (NLP)—to the case of linear optimization over linear and conic constraints, of which problems (1) and (2) are particular examples. More specifically, let $C \subseteq \mathbb{R}^q$ be a closed, convex cone, and let

$$X := \{y \in \mathbb{R}^q \mid Ey = f,\ y \in C\}. \tag{17}$$

We consider the following generic problem throughout this section:

$$\min\left\{c^T y \mid Ay = b,\ y \in X\right\}. \tag{18}$$

Here, $y \in \mathbb{R}^q$ is the optimization variable and $c \in \mathbb{R}^q$, $A \in \mathbb{R}^{p\times q}$, and $b \in \mathbb{R}^p$ are the data. We assume that an optimal solution exists and denote the optimal value by v^*.

We acknowledge that initially it may seem counterintuitive to apply a NLP algorithm to linear conic problems. In fact, we are aware of only two studies [35, 40], which consider the augmented Lagrangian method in such a context, in particular for $C = \mathbb{R}^q_+$. In this section, however, besides laying the groundwork for the augmented Lagrangian algorithm, we also hope to highlight the advantages of the method, which when combined with several computational ideas discussed in section 4 make it a good choice for optimizing (1) and (2).

3.1. The augmented Lagrangian method. The augmented Lagrangian method can be seen as a combination of the standard subgradient and quadratic penalty methods. It is based on the following function, which is specified for fixed $\lambda \in \mathbb{R}^p$ and $\sigma > 0$:

$$L_{\lambda,\sigma}(y) = c^T y + \lambda^T(b - Ay) + \frac{\sigma}{2}\|b - Ay\|^2. \tag{19}$$

The augmented Lagrangian method is then stated as Algorithm 1. Roughly speaking, the augmented Lagrangian method runs the subgradient and quadratic penalty methods at the same time by alternating between the update of λ and σ (typically using some predefined update strategy, such as performing the nontrivial update of σ every 10 iterations). Some of the main advantages of the augmented Lagrangian algorithm over subgradient and penalty methods are that it yields both primal and dual solutions, as well as dual bounds, for (18).

Algorithm 1 Augmented Lagrangian algorithm

Set $\lambda^1 = 0$, $\sigma_1 = 1$
for $k = 1, 2, 3, \ldots,$ **do**
  Calculate some $y^k \in \operatorname{Argmin}\left\{L_{\lambda^k,\sigma_k}(y) : y \in X\right\}$;
  Calculate the subgradient $d^k = b - Ay^k$;
  Choose either ($\alpha_k = \sigma_k$ and $\eta_k = 1$) or ($\alpha_k = 0$ and $\eta_k \gg 1$);
  Calculate $\lambda^{k+1} = \lambda^k + \alpha_k d^k$ and $\sigma_{k+1} = \eta_k \sigma_k$.
end for

There are also some additional, less obvious advantages of the augmented Lagrangian method. First, an important feature of the augmented Lagrangian, in contrast with the subgradient method, is that there is a definitive choice of step-size α_k in each iteration. This choice is dictated by convergence results for augmented Lagrangian methods (see [6]) and also seems to work well in practice. Second, it is well

known in the study of NLP that the introduction of explicit dual variables into the quadratic penalty method tends to lessen the ill-conditioning encountered in penalty methods. In particular, it can be proved that if the iterates λ^k are sufficiently close to an optimal dual solution, then there is a finite value of σ that still guarantees convergence [6].

In fact, in the specific case of (18), where X is given by (17), it is possible to show a much stronger result, namely that Algorithm 1 converges even if σ_k is held constant at an arbitrary initial value $\sigma_1 > 0$. In order to state the result, we point out that the dual of (18) can be written explicitly as

$$\max \left\{ b^T\lambda + f^T\eta \mid A^T\lambda + E^T\eta + s = c,\ s \in C^* \right\}, \tag{20}$$

where

$$C^* := \left\{ s \in \mathbb{R}^q : s^T y \geq 0 \ \ \forall\, y \in C \right\}$$

is the dual cone of C, $\eta \in \mathbb{R}^m$ is the dual variable associated with the constraint $Ey = f$, and $s \in \mathbb{R}^q$ is the dual variable associated with the constraint $x \in C$. We also make the following assumptions: A has full row rank, and both (18) and (20) have Slater points, i.e., feasible points in the interior of C and C^*, respectively. In addition, we let η^k and s^k denote the optimal multipliers associated with $Ey = f$ and $y \in C$ gotten from the solution of the kth augmented Lagrangian subproblem. The result is the following theorem.

THEOREM 3.1. *Let X be given by* (17)*, and suppose that Algorithm* 1 *is executed so that $\sigma_k = \sigma_1$ for all $k \geq 1$, i.e., the dual multiplier is updated nontrivially in each iteration. Then any accumulation point of the combined sequence $\{(y^k, \lambda^{k+1}, \eta^k, s^k)\}$ constitutes a primal-dual optimal pair of* (18) *and* (20)*.*

This result is proven by Poljak and Tret′jakov [35] for the specific case of $C = \mathbb{R}^q_+$, and the essence of their proof carries over to general C. Although it is not difficult to extend the result based on their ideas, we include a proof in section 3.4 for completeness.

We finish this subsection with a few observations. First, Theorem 3.1 shows that (1) and (2) can theoretically be solved without ever increasing σ, and computational experiments support this. In practice, however, it still may be advantageous to increase σ to moderate levels in order to facilitate convergence of the method, as we demonstrate in section 4.

Second, even with the theoretical and practical benefits of augmented Lagrangian methods, it is still important to keep in mind that the inner optimization over $y \in X$ in each iteration of the algorithm utilizes a convex quadratic objective, instead of a linear one as in the subgradient method. In some applications, this is a disadvantage of the augmented Lagrangian method that may preclude its use, but we will show that, in the case of (1) and (2), the use of the augmented Lagrangian method is indeed beneficial.

Third, in practice it is rarely the case that the inner minimization of Algorithm 1 is computed exactly. Nonetheless, convergence is typically observed in practice even if y^k is a "nearly" optimal solution [6]. This behavior will guide some of our implementation decisions, which we describe in section 4.

3.2. Variations on the augmented Lagrangian method. Typically the augmented Lagrangian method is stated in terms of handling difficult equality constraints, such as the constraints $Ay = b$ in (18). Although there exist variations which can

handle inequality constraints directly (see [34]), a standard approach for handling inequalities is to simply add slack variables and then to revert to the equality case. For example, consider the problem

$$\min\left\{c^T y \mid Ay \le b,\ y \in X\right\}.$$

By introducing the variable $z \in \mathbb{R}^p_+$, we have the equivalent problem

$$\min\left\{c^T y \mid Ay + z = b,\ (y,z) \in X \times \mathbb{R}^p_+\right\}.$$

The augmented Lagrangian can now be applied directly to the second formulation. Of course, the inner optimization of the augmented Lagrangian algorithm is now slightly more complicated due to the addition of slack variables, but this complication is usually worth the trouble.

Further extensions of this idea can also be considered. If $Z \subseteq \mathbb{R}^p$ is an arbitrary set, a problem of the form

$$\min\left\{c^T y \mid b - Ay \in Z,\ y \in X\right\}$$

can then be converted to

$$\min\left\{c^T y : Ay + z = b,\ (y,z) \in X \times Z\right\},$$

after which the augmented Lagrangian method can be applied. Here again, the simplicity of the inner optimization over $(y,z) \in X \times Z$ is the key to the overall efficiency of the augmented Lagrangian algorithm. In particular, convergence will be theoretically and practically more reliable if Z is convex.

3.3. Relationship with the bundle method. In section 3.1, we have presented the augmented Lagrangian algorithm as an alternative to the standard subgradient algorithm. When the set X is convex, another well-known alternative to the subgradient algorithm is the bundle method (see [29, 23]). Like the subgradient method, the bundle method uses subgradient information to produce a sequence $\{\lambda^k\}$ of dual multipliers whose corresponding Lagrangian objective values converge to v^*. However, the bundle method differs from the subgradient method in the precise way that the subgradient information is used.

The bundle method is initialized with $\lambda^1 = 0$ and, at the kth iteration, the basic idea is to calculate λ^{k+1} by solving an approximation of the Lagrangian dual optimization

$$\sup_{\lambda \in \mathbb{R}^p} \min\left\{c^T y + \lambda^T(b - Ay) \mid y \in X\right\}. \tag{21}$$

More specifically, the bundle method assumes that a current best point $\bar{\lambda}$ has been calculated (note that $\bar{\lambda}$ does not necessarily equal λ^k) and that a finite collection of dual points $\{\lambda^j \mid j \in J^k\}$ is available, for some finite index set J^k. For example, one may take $J^k = \{1, \ldots, k\}$ so that the dual points correspond to the dual solutions λ^j already produced by the algorithm in the first $k-1$ iterations, though other choices are possible. Defining $\tilde{X} := \operatorname{conv}\{y^j : j \in J^k\}$, the approximation of (21) is then given as

$$\sup_{\lambda \in \mathbb{R}^p} \min\left\{c^T y + \lambda^T(b - Ay) \mid y \in \tilde{X}\right\}. \tag{22}$$

Generally speaking, the inner minimization of (22) (viewed as a function of λ) is only considered a reliable approximation of the inner minimization of (21) for those λ relatively close to $\bar{\lambda}$. Hence, the next iterate λ^{k+1} is not actually chosen as an optimal solution of (22), but rather as an optimal solution of

$$\max_{\lambda \in \mathbb{R}^p} \min \left\{ c^T y + \lambda^T (b - Ay) \mid y \in \tilde{X} \right\} - \frac{\rho}{2} \left\| \lambda - \bar{\lambda} \right\|^2, \tag{23}$$

where $\rho > 0$ is a proximity parameter. In other words, (23) is similar to (22) except that it penalizes points that are too far from the current best iterate $\bar{\lambda}$, and the parameter ρ controls the precise amount of penalization. Once λ^{k+1} has been calculated, the bundle method calculates the value of the Lagrangian function at λ^{k+1} and decides whether or not λ^{k+1} should become the new best iterate $\bar{\lambda}$.

With this description of the bundle method, it is not difficult to see that the bundle method's procedure for computing λ^{k+1} is similar to that of the augmented Lagrangian algorithm. Indeed, (23) can be rearranged as

$$\min \left\{ \max_{\lambda \in \mathbb{R}^p} \; c^T y + \lambda^T (b - Ay) - \frac{\rho}{2} \left\| \lambda - \bar{\lambda} \right\|^2 \mid y \in \tilde{X} \right\}. \tag{24}$$

Since the inner optimization of (24) is a concave maximization over $\lambda \in \mathbb{R}^p$, its optimal solution is given by $\bar{\lambda} + \rho^{-1}(b - Ay)$, which further simplifies (24) to

$$\min \left\{ c^T y + \bar{\lambda}^T (b - Ay) + \frac{1}{2\rho} \| b - Ay \|^2 \mid y \in \tilde{X} \right\}. \tag{25}$$

Letting $\sigma = \rho^{-1}$, we now easily see that (25) is similar in form to the kth augmented Lagrangian subproblem except that (25) approximates X by $\tilde{X}$. Next, once the bundle method calculates an optimal solution y^k of (25), λ^{k+1} is calculated by the formula

$$\lambda^{k+1} := \lambda^k + \rho^{-1} \left(b - Ay^k \right),$$

which matches the update formula used by the augmented Lagrangian algorithm.

As described above, in addition to the approximation $\tilde{X}$ of X, the bundle method differs from the augmented Lagrangian method in that it selectively keeps a current best iterate $\bar{\lambda}$ (which affects each stage of the algorithm), whereas the augmented Lagrangian algorithm simply generates each λ^k in succession. It is further interesting to note that the bundle method is known to converge for fixed ρ.

3.4. Proof of Theorem 3.1. In this subsection, we give the proof of Theorem 3.1, which has been stated in section 3.1. We remark that our proof is an extension of the proof given in [35] for the case $C = \mathbb{R}^q_+$.

The main idea of the theorem is that, when X is given by (17), the augmented Lagrangian algorithm converges without ever increasing the penalty parameter σ. For this, we consider the value $\sigma > 0$ to be fixed throughout the execution of Algorithm 1, i.e., $\sigma_k = \sigma$ for all $k \geq 1$. Also recall the following assumptions: A has full row rank, and (18) and (20) have Slater points. We will investigate three sequences produced by the algorithm:

(i) the sequence $\{y^k\}$ of primal estimates;
(ii) the shifted sequence $\{\lambda^{k+1}\}$ of dual multipliers; note that λ^{k+1} is calculated as a result of the kth augmented Lagrangian subproblem;
(iii) the sequence $\{(\eta^k, s^k)\}$ of optimal multipliers for the constraints $Ey = f$ and $y \in C$ in the sequence of augmented Lagrangian subproblems.

Because (18) and (20) each have a Slater point, strong duality holds and there exists a primal-dual solution (y, λ, η, s) that satisfies $s^T y = 0$. The kth augmented Lagrangian problem (with fixed σ) is

$$\min \left\{ c^T y + (\lambda^k)^T (b - Ay) + \frac{\sigma}{2} \|b - Ay\|^2 \mid Ey = f,\ y \in C \right\}, \tag{26}$$

and its dual (see [16] for QP duality in the case of $\mathbb{R}^q_+$) can be stated as

$$\begin{aligned} \max \quad & b^T \lambda^k + f^T \eta + \frac{\sigma}{2} \left(b^T b - v^T A^T A v \right) \\ \text{s.t.} \quad & A^T \left(\lambda^k + \sigma(b - Av) \right) + E^T \eta + s = c \\ & s \in C^*. \end{aligned} \tag{27}$$

Since (18) has a Slater point, so does (26). Furthermore, since A has full row rank, we can use a Slater point from (20) to construct such a point for (27). As a result, strong duality also holds between (26) and (27), and there exists a primal-dual solution (y, v, η, s) such that $v = y$ and $s^T y = 0$.

We first show that $Ay^k \to b$ via two lemmas and a proposition.

LEMMA 3.2. *Let $\bar{\lambda}$ and $\hat{\lambda}$ be arbitrary multipliers for $Ax = b$, and let $\bar{y}$ and $\hat{y}$ be optimal solutions of the corresponding augmented Lagrangian subproblems. Then*

$$\left(\bar{\lambda} - \hat{\lambda} \right)^T (A\bar{y} - A\hat{y}) \geq \sigma \|A\bar{y} - A\hat{y}\|^2.$$

Proof. Optimality of $\bar{y}$ with respect to $\bar{\lambda}$ implies

$$(c - A^T(\bar{\lambda} + \sigma(b - A\bar{y})))^T (y - \bar{y}) \geq 0$$

for all y such that $Ey = f, y \in C$. Likewise, for $\hat{y}$ and $\hat{\lambda}$:

$$(c - A^T(\hat{\lambda} + \sigma(b - A\hat{y})))^T (y - \hat{y}) \geq 0.$$

Applying these results with $y = \hat{y}$ and $y = \bar{y}$, respectively, summing the two resultant inequalities, and rearranging terms, we achieve the result. ∎

LEMMA 3.3. *Let (λ^*, η^*, s^*) be an optimal solution of* (20). *Then any $y \in X$ is optimal for the augmented Lagrangian subproblem corresponding to λ^* if and only if y is optimal for* (18).

Proof. Using dual feasibility, we have that

$$c^T y + (\lambda^*)^T (b - Ay) + \frac{\sigma}{2} \|b - Ay\|^2 = (\lambda^*)^T b + f^T \eta^* + (s^*)^T y + \frac{\sigma}{2} \|b - Ay\|^2 .$$

Hence, ignoring the constant terms $b^T \lambda^* + f^T \eta^*$, the minimum value attainable by the augmented Lagrangian function is clearly bounded below by 0. Moreover, 0 is attained if and only if $(s^*)^T y = 0$ and $Ay = b$, which proves the result. ∎

PROPOSITION 3.4. *The sequence $\{Ay^k\}$ converges to b.*

Proof. Let (λ^*, η^*, s^*) be any optimal solution of (20), and let y^* be any optimal solution of (18). For all $k \geq 1$,

$$\begin{aligned} \|\lambda^{k+1} - \lambda^*\|^2 &= \|\lambda^k - \lambda^*\|^2 + 2\sigma(b - Ay^k)^T (\lambda^k - \lambda^*) + \sigma^2 \|b - Ay^k\|^2 \\ &\leq \|\lambda^k - \lambda^*\|^2 - 2\sigma^2 \|Ay^* - Ay^k\|^2 + \sigma^2 \|b - Ay^k\|^2 \\ &= \|\lambda^k - \lambda^*\|^2 - \sigma^2 \|b - Ay^k\|^2 \text{ (by Lemmas 3.2 and 3.3).} \end{aligned}$$

For arbitrary N, summing this inequality for $k = 1, \ldots, N$, we have

$$\begin{aligned} 0 &\leq \sum_{k=1}^{N} \left(\|\lambda^k - \lambda^*\|^2 - \sigma^2 \|b - Ay^k\|^2 - \|\lambda^{k+1} - \lambda^*\|^2 \right) \\ &= \|\lambda^1 - \lambda^*\|^2 - \|\lambda^{N+1} - \lambda^*\|^2 - \sigma^2 \sum_{k=1}^{N} \|b - Ay^k\|^2, \end{aligned}$$

which implies $\sigma^2 \sum_{k=1}^{N} \|b - Ay^k\|^2 \leq \|\lambda^1 - \lambda^*\|^2$. Hence, because N is arbitrary, Ay^k must converge to b. ∎

Now, with Proposition 3.4 and an additional lemma, we prove Theorem 3.1.

LEMMA 3.5. *For all $k \geq 1$, y^k and $(\lambda^{k+1}, \eta^k, s^k)$ are primal-dual optimal solutions of*

$$\min \left\{ c^T y \mid Ay = Ay^k,\ Ey = f,\ y \in C \right\}, \tag{28}$$

$$\max \left\{ (Ay^k)^T \lambda + f^T \eta \mid A^T \lambda + E^T \eta + s = c,\ s \in C^* \right\}. \tag{29}$$

Proof. Clearly, y^k is feasible for (28). Moreover, strong duality between (26) and (27) implies that (λ^k, η^k, s^k) is feasible for (27) such that $(s^k)^T y^k = 0$. Combining this with the definition $\lambda^{k+1} := \lambda^k + \sigma(b - Ay^k)$, we see that $(\lambda^{k+1}, \eta^k, s^k)$ is feasible for (29) and that strong duality holds between (28) and (29). This proves the result. ∎

Proof of Theorem 3.1. By Lemma 3.5, for each $k \geq 1$, $(y^k, \lambda^{k+1}, \eta^k, s^k)$ is a solution of the nonlinear system

$$\begin{aligned} &Ay = Ay^k,\ Ey = f,\ y \in C \\ &A^T \lambda + E^T \eta + s = c,\ s \in C^* \\ &y^T s = 0. \end{aligned}$$

By continuity, any accumulation point $(\bar{y}, \bar{\lambda}, \bar{\eta}, \bar{s})$ of $\{(y^k, \lambda^{k+1}, \eta^k, s^k)\}$ satisfies the above system with Ay^k replaced by its limit b (due to Proposition 3.4). In other words, $(\bar{y}, \bar{\lambda}, \bar{\eta}, \bar{s})$ is a primal-dual optimal solution for (18) and (20). ∎

4. Computational issues and results. In this section, we discuss the implementation details of the augmented Lagrangian algorithm used to solve (1) and (2). We then demonstrate the effectiveness of this approach on various problem classes and also illustrate advantages of the augmented Lagrangian approach over other methods.

4.1. Optimizing over $N(K)$ and $N_+(K)$. We have suggested in section 2 that one could consider a purely Lagrangian approach for solving the linear relaxation (1) since the calculation of $L(\lambda)$ is separable into $2n + 1$ LPs over the columns of the variables Y and Z. We have argued in section 3, however, that the augmented Lagrangian method has several advantages over the Lagrangian approach. In the case of (1), the kth augmented Lagrangian subproblem can be stated as

$$\min\ \tilde{c}^T Y e_0 + (\lambda^k)^T h(Z, Y) + \frac{\sigma_k}{2} \|h(Z, Y)\|^2 \tag{30}$$

$$\text{s.t.}\quad Ye_i \in \hat{K} \quad \forall\ i = 0, 1, \ldots, n \tag{31}$$

$$Ze_i \in \hat{K} \quad \forall\ i = 1, \ldots, n \tag{32}$$

$$Y_{00} = 1. \tag{33}$$

An important observation is that, in contrast with $L(\lambda)$, (30)–(33) is a nonseparable convex QP. More precisely, the quadratic term in the objective (30) is the sole cause of the nonseparability.

Even with this complication, we still advocate the use of the augmented Lagrangian method. To exploit the structure inherent in the constraints (31)–(33), we propose to employ block coordinate descent for solving the subproblem, iteratively taking a descent step over a particular column of Y or Z, while keeping all other columns fixed. Block coordinate descent is known to be a convergent method; see Proposition 2.7.1 in [6].

When the amount of coupling between the blocks is small, block coordinate descent can be expected to converge quickly. One can observe that, because of the particular structure of $h(Y, Z)$, the amount of coupling between the columns of Y and Z is relatively small. In particular, the greatest amount of coupling is between Ye_i, Ze_i, and Ye_0 for $i = 1, \ldots, n$. As a result, we expect block coordinate descent to be a good choice for optimizing (30)–(33).

For optimizing over $N_+(K)$, we also advocate the augmented Lagrangian method, but at first glance, it is not clear how to handle the constraint that Y be positive semidefinite. This constraint is difficult not only because it involves positive semidefiniteness but also because it links the columns of Y. To handle this constraint, we follow the suggestions laid out in section 3.2. In particular, we introduce an "excess" variable U, which is required to be symmetric positive semidefinite and must also satisfy $U = Y$, and simultaneously drop the positive semidefiniteness constraint on Y. After introducing a symmetric matrix S of Lagrange multipliers for the constraint $U = Y$, the resulting kth augmented Lagrangian subproblem becomes

$$\begin{aligned} \min \quad & \tilde{c}^T Y e_0 + (\lambda^k)^T h(Z, Y) + \frac{\sigma_k}{2}\|h(Y, Z)\|^2 + S^k \bullet (U - Y) + \frac{\sigma_k}{2}\|U - Y\|_F^2 \\ \text{s.t.} \quad & (31)\text{–}(33), \ U \succeq 0. \end{aligned}$$

Again we propose to solve this problem using block coordinate descent. For example, we first fix U and solve a series of QPs as we did in the case (30)–(33), one for each column of Y and Z. We then proceed by fixing Y and Z and solving the subproblem over U, which is equivalent to

$$\min \left\{ 2\,\sigma_k^{-1}\, S^k \bullet (U - Y) + \|U - Y\|_F^2 \ \mid \ U \succeq 0 \right\}.$$

By completing the square, we see that

$$2\,\sigma_k^{-1}\, S^k \bullet (U - Y) + \|U - Y\|_F^2 = \|\sigma_k^{-1} S^k + (U - Y)\|_F^2 - \sigma_k^{-2}\, S^k \bullet S^k,$$

so that the above minimization is equivalent to solving

$$\min \left\{ \left\|\sigma_k^{-1} S^k + (U - Y)\right\|_F^2 \ \mid \ U \succeq 0 \right\},$$

which in turn is solved explicitly by projecting $Y - \sigma_k^{-1} S^k$ onto the cone of symmetric positive semidefinite matrices.

It is well known that calculating the projection M_+ of a symmetric matrix M onto the cone of symmetric positive semidefinite matrices can be done by calculating the spectral decomposition $M = QDQ^T$ and then forming the matrix $M_+ = QD_+Q^T$, where D_+ is derived from D by replacing all negative diagonal entries with 0. However, in our case, the matrix that we project, $M = Y - \sigma_k^{-1} S^k$, is not symmetric.

TABLE 1
Number of variables and constraints in the descriptions of $N(K)$ and $N_+(K)$ that serve as the basis of the augmented Lagrangian algorithm. Note that m represents the number of linear constraints (including lower and upper bounds) present in $\hat{K}$.

	Variables		Constraints		
	Y, Z	U	Linear enforced	Linear relaxed	SDP
$N(K)$	$(2n+1)(n+1)$	0	$1+(2n+1)m$	$n\left(\frac{3}{2}n+\frac{5}{2}\right)$	0
$N_+(K)$	$(2n+1)(n+1)$	$\frac{1}{2}(n+1)(n+2)$	$1+(2n+1)m$	$n\left(\frac{3}{2}n+\frac{5}{2}\right)+(n+1)^2$	1

Nevertheless, by using the identity

$$\begin{aligned}\|U-M\|_F^2 &= \left\|U-\frac{1}{2}(M+M^T)-\frac{1}{2}(M-M^T)\right\|_F^2 \\ &= \left\|U-\frac{1}{2}(M+M^T)\right\|_F^2 - \left(U-\frac{1}{2}(M+M^T)\right)\bullet\left(M-M^T\right)+\frac{1}{4}\left\|M-M^T\right\|_F^2 \\ &= \left\|U-\frac{1}{2}(M+M^T)\right\|_F^2 + \frac{1}{4}\left\|M-M^T\right\|_F^2,\end{aligned}$$

where the final equality follows from the fact that the dot product of a symmetric matrix and a skew-symmetric matrix is zero, we can easily see that the projection of M is equal to the projection of $(M+M^T)/2$, which is itself symmetric.

Note that since S is the dual multiplier for the equality $U=Y$, it is unrestricted. However, from basic duality, it is not difficult to see that S will be constrained to be positive semidefinite in the dual problem. An illustration of this is as follows. Consider the generic SDP

$$\min\left\{C\bullet Y \mid \mathcal{A}(Y)=b,\ Y-U=0,\ U\succeq 0\right\},$$

which has the dual SDP

$$\max\left\{b^T y \mid \mathcal{A}^*(y)+S=C,\ -S\preceq 0\right\};$$

here, $\mathcal{A}$ is a generic linear operator and $\mathcal{A}^*$ its adjoint. Thus, without loss of generality, we may restrict each S^k to be positive semidefinite, enforcing this by projection after each dual update.

Before moving onto the description of the specifics of the implementation in the next subsection, we would like to give some sense of the actual size of the LPs and SDPs that we are proposing to solve with the augmented Lagrangian method. This is given in Table 1. In the table, the quantity m represents the number of linear constraints (including lower and upper bounds) present in $\hat{K}$.

4.2. Implementation details. The augmented Lagrangian algorithm for (1) and (2) has been implemented in ANSI C under the Linux operating system on a Pentium 4 having a 2.4 GHz processor and 1 GB of RAM. The pivoting algorithm of CPLEX 8.1 for convex QP [21] has been employed for solving the $2n+1$ quadratic subproblems encountered during block coordinate descent, and LAPACK [1] has been utilized for the spectral decompositions required when projecting onto the positive semidefinite cone.

The choice of a pivoting algorithm for the quadratic subproblems—as opposed to an interior-point algorithm—was motivated by the warm-start capabilities of pivoting

algorithms. In particular, it is not difficult to see that the objective functions of the $2n+1$ quadratic subproblems change only slightly between loops of block coordinate descent or between iterations of the augmented Lagrangian algorithm. As a result, the ability to warm-start from an advance basis has proven to be invaluable for speeding up the overall algorithm.

Although Theorem 3.1 indicates that it is theoretically not necessary to increase the penalty parameter σ during the course of the algorithm, we have found that it is indeed advantageous to increase σ in order to enhance convergence. Our update rule is as follows:

> *Penalty update rule.* Every 500 iterations, σ is increased by a factor of 10.

We consider this to be a fairly conservative update rule.

Practically speaking, during the course of the algorithm, one can expect the norm of the constraint violation—$\|h(Y^k, Z^k)\|$ in the case of (1) and $(\|h(Y^k, Z^k)\|^2 + \|Y^k - U^k\|_F^2)^{1/2}$ in the case of (2)—to decrease towards 0, which is an indication that the algorithm is converging. As a result, we implement the following overall stopping criterion:

> *Stopping criterion.* The augmented Lagrangian algorithm is terminated once primal iterates are calculated such that the corresponding constraint violation is less than 10^{-6}.

We remark that, during early experiments on a handful of instances, this criterion was not achieved due to numerical difficulties caused by a large value of σ—typically around 10^8 or higher. As a result, we have also implemented the following:

> *Alternate stopping criterion.* The augmented Lagrangian algorithm is terminated once σ grows larger than 10^8.

Another implementation detail is how accurately the augmented subproblem is solved in each iteration. Recall from section 3 that it is not theoretically necessary to solve each subproblem exactly, and in fact, we have found in practice that it often suffices to solve them fairly loosely. In the computational results presented in section 4.4, our goal is to highlight the quality and speed of dual bounds provided by (1) and (2), rather than to calculate optimal solutions of high precision. In light of this goal, we decided to "solve" each augmented Lagrangian subproblem by performing exactly one cycle of block coordinate descent.

4.3. Problems. We have chosen four classes of 0-1 integer programs to illustrate the performance of the augmented Lagrangian algorithm.

4.3.1. Maximum stable set. Given an undirected, simple graph G with vertex set $V = \{1, \ldots, n\}$ and edge set $E \subseteq V \times V$, the (unweighted) maximum stable set problem is

$$\max \left\{ e^T x \mid x_i + x_j \leq 1,\ (i,j) \in E,\ x \in \{0,1\}^n \right\}.$$

As mentioned in section 2, the maximum stable set problem has been studied extensively by Lovász and Schrijver, and a number of theoretical results are known which illustrate the strength of (1) and (2).

We have collected a total of 26 graphs for testing; a basic description of these problems can be seen in Table 4. All graphs were obtained from the Center for Discrete Mathematics and Theoretical Computer Science [15] and originated as test instances for the maximum clique problem in the Second DIMACS Implementation Challenge. As such, we actually use the complement graphs as instances of the maximum stable set problem.

4.3.2. Problem of Erdös and Turán. We consider a 0-1 integer programming formulation of a problem studied by Erdös and Turán: calculate the maximum size of a subset of numbers in $\{1,\dots,n\}$ such that no three numbers are in arithmetic progression. This number is the optimal value of

$$\max\left\{e^T x \mid x_i + x_j + x_k \le 2,\ i+k=2j,\ i<j<k,\ x\in\{0,1\}^n\right\}.$$

For a full discussion, we refer the reader to [14], which includes background on the problem as well as some active set approaches for approximately optimizing (2) in this case. In the computational results, we consider 10 instances for $n = 60, 70, \dots, 150$. It is interesting to note that the number of constraints in $N(K)$ and $N_+(K)$ for $n = 150$ is approximately 1.7 million.

4.3.3. Market share. The market share instances from MIPLIB [33], markshare1 and markshare2, have proven to be very small yet challenging instances of mixed integer programs. They are not pure binary integer programs, so one cannot apply the lift-and-project operator directly. It is easy, however, to generate very similar problems using only 0-1 variables. We first generate an $m \times n$ matrix, A, exactly as is done for the market share instances (see [13]). We also define the vector b by $b_i = \lfloor \frac{1}{2}\sum_{j=1}^n A_{ij} \rfloor$ for each $i = 1,\dots,m$. The resulting IP is

$$\begin{aligned}\min\ & \sum_{i=1}^m \left(b_i - \sum_{j=1}^n A_{ij}x_j \right) \\ & Ax \le b \\ & x \in \{0,1\}^n.\end{aligned}$$

We generated two instances of these problems of the same sizes as markshare1 (6×50) and markshare2 (7×60).

4.3.4. Quadratic assignment. Besides 0-1 linear integer programs, the lift-and-project relaxations provided by Lovász and Schrijver can easily be applied to 0-1 QPs with linear constraints. A quadratic objective $x^T Q x$ in the original problem becomes the linear objective

$$\begin{pmatrix} 0 & 0 \\ 0 & Q \end{pmatrix} \bullet Y$$

in the lifted problem. Using this technique, we can also consider the quadratic assignment problem (QAP), which is a problem of this type arising in location theory.

Because of its difficulty, QAP has attracted a large amount of attention and study; see [36, 10] and the recent survey by Anstreicher [2]. One of the most common forms of the QAP is the Koopmans–Beckmann form: given an integer p and matrices

$A, B \in \mathbb{R}^{p\times p}$, the QAP is

$$\begin{aligned}
\min \ & \sum_{i=1}^{p}\sum_{j=1}^{p}\sum_{k=1}^{p}\sum_{l=1}^{p} a_{ij}b_{kl}x_{ik}x_{jl} \\
\text{s.t.} \ & \sum_{i=1}^{p} x_{ik} = 1 \quad \forall\ k = 1,\dots,p \\
& \sum_{k=1}^{p} x_{ik} = 1 \quad \forall\ i = 1,\dots,p \\
& x_{ik} \in \{0,1\} \quad \forall\ i,k = 1,\dots,p.
\end{aligned}$$

We have taken 91 test instances from QAPLIB, and all instances in QAPLIB are in Koopmans–Beckmann form.

In the effort to solve QAP to optimality, a variety of dual bounds have been developed for QAP. In particular, we will compare with three bounds from the literature: (i) the Gilmore–Lawler bound (denoted GLB) [17, 28]; the LP bound (denoted KCCEB) found in [22]; and the semidefinite programming bound (denoted RSB) of Rendl and Sotirov [37]. It is known that KCCEB is stronger than GLB, and it has been observed that RSB is stronger than KCCEB. As one might expect, however, RSB requires the most time to compute, while GLB takes the least.

The bound KCCEB is based on the first-level reformulation-linearization technique of Sherali and Adams [38] applied to QAP. It is not difficult to see that this relaxation is equivalent to (1), and so the bound provided by (1) and KCCEB are theoretically equal, although slight differences are observed in practice due to computational differences such as the level of precision to which the relaxation is solved.

The derivation of RSB is based on similar lift-and-project ideas as (2). However, RSB selectively includes certain constraints that are implied by $N_+(K)$, while adding in additional constraints that are not implied by $N_+(K)$ (at least not implied explicitly). It is currently unclear whether one bound is theoretically stronger than the other, although our computational results in the next subsection show that the Lovász–Schrijver bound is stronger than RSB on all test instances.

One additional comment regarding QAP is in order. Since QAP has equality constraints and upper bounds on the variables are redundant, it is possible to show that the constraints (7) of (1) and (2) are implied by (6). As a result, in this instance it is unnecessary to introduce the variable Z, which has the benefit of reducing the number of quadratic subproblems in the block coordinate descent from $2n+1$ to $n+1$. We have implemented these savings in the code and remark that the calculation of KCCEB has taken this into account as well.

4.4. Results. We first provide a comparison of our implementation with existing methods on a few problems selected from our test instances. We feel that these comparisons provide a fair indication of the advantages of our method in terms of both bound quality and computation time. Next, we provide detailed information on the performance of our method on the four problem classes discussed above.

4.4.1. Comparison with linear and semidefinite solvers. We directly solved some instances of (1), i.e., optimization over $N(K)$, using the dual simplex LP algorithm of CPLEX 8.1 [21] (with both default pricing and steepest-edge pricing), enforcing a time limit of 15,000 seconds. We show these results in Table 2 together with the running times and bounds obtained by the augmented Lagrangian approach

TABLE 2

Comparisons of bounds achieved for optimization over $N(K)$ by CPLEX 8.1 dual simplex (with default and steepest-edge pricing) and the augmented Lagrangian method. Timings (in seconds) are also given. When prefixed to a timing, the symbol ($\star$) indicates that the algorithm did not terminate within the 15,000 seconds allotted.

	Bound			Time(s)		
	CPLEX		Auglag	CPLEX		Auglag
Name	Default	Steep		Default	Steep	
MANN_a9	18.0000	18.0000	18.0000	3	5	5
johnson8-2-4	9.3333	9.3333	9.3333	0	0	6
johnson8-4-4	23.3333	23.3333	23.3333	88	85	90
hamming6-2	32.0000	32.0000	32.0000	11	10	11
hamming6-4	21.3333	21.3333	21.3334	12	12	134
johnson16-2-4	40.0000	40.0000	40.0001	1,505	1,521	763
keller4	57.0000	57.0000	57.0001	4,745	5,059	4,910
hamming8-2	128.9971	128.9971	128.0000	$\star$ 15,000	$\star$ 15,000	640
san200_09_1	86.3989	82.3446	70.1613	$\star$ 15,000	$\star$ 15,000	4,840
san200_09_2	83.7750	87.7951	66.6667	$\star$ 15,000	$\star$ 15,000	3,484
san200_09_3	80.4699	83.8000	66.6678	$\star$ 15,000	$\star$ 15,000	2,538
Erdös–Turán ($n = 60$)	34.2857	34.2857	34.6688	4,126	2,421	354
Erdös–Turán ($n = 70$)	40.0000	40.0000	40.7446	$\star$ 15,000	14,119	778
Erdös–Turán ($n = 80$)	46.5922	46.4502	46.7352	$\star$ 15,000	$\star$ 15,000	1,686
Erdös–Turán ($n = 90$)	54.5405	54.1181	53.0720	$\star$ 15,000	$\star$ 15,000	2,550
Erdös–Turán ($n = 100$)	63.8962	63.4383	59.6474	$\star$ 15,000	$\star$ 15,000	2,810

applied to (1). Note that we include both stable set and Erdös–Turán instances and that the instances are ordered roughly in terms of increasing size. We were unable to solve larger, denser instances with CPLEX as these required more than the available memory. For very large problems, these results clearly indicate the ability of our method to obtain bounds for $N(K)$ in much less time than is required by standard LP solvers.

We also attempted to carry out some optimizations of $N_+(K)$ using standard semidefinite solvers, such as CSDP developed by Borchers [8], but found that all but the smallest of instances would require more than the 1 GB of available memory on our computer.

4.4.2. Comparison with subgradient methods. We implemented a subgradient approach for optimizing over $N(K)$ using the volume algorithm developed in [5], for which an open source framework is available from the COIN-OR repository [31]. We found that this subgradient implementation in the current context was sensitive to the choice of initial dual multipliers. Consequently, some experimentation was required to settle upon appropriate starting duals. In particular, we found that starting duals of all -1's performed much better than all 0's.

In Table 3, we compare the subgradient algorithm and our augmented Lagrangian algorithm initialized with the same starting duals. We report not only the final bound and total running time for both algorithms but also the time at which one algorithm surpasses the best bound of the other. For example, on the *brock*200_1 instance, in 2,052 seconds the augmented Lagrangian algorithm achieved the same bound that the subgradient algorithm achieved after 8,603 seconds. A quick summary of Table 3 is that the subgradient algorithm outperforms augmented Lagrangian on the Erdös–Turán instances, whereas augmented Lagrangian outperforms subgradient on the stable set instances.

We found that increasing the number of cycles of coordinate descent (recall that we use only one cycle in all computations in this paper) improved the convergence

TABLE 3

Comparisons of bounds achieved for optimization over $N(K)$ by the subgradient method and the augmented Lagrangian method, along with timings (in seconds). Also shown is the time at which one algorithm surpassed the best bound of the other.

	Bound		Time(s)		Surpass Time(s)	
Instance	Subgrad	Auglag	Subgrad	Auglag	Subgrad	Auglag
MANN_a9	18.0098	18.0000	11	8		8
brock200_1	69.3607	66.6668	8,603	6,175		2,052
brock200_2	68.1062	66.6667	18,672	13,951		5,607
brock200_3	68.8632	66.6667	13,420	9,773		3,801
brock200_4	68.0411	66.6668	9,136	8,378		3,596
c-fat200-1	66.7608	66.6668	22,995	26,652		14,295
c-fat200-2	66.8576	66.6668	27,681	23,859		12,090
c-fat200-5	67.5605	66.6672	28,285	15,937		7,103
hamming6-2	32.0119	32.0000	31	20		20
hamming6-4	21.3543	21.3334	212	155		103
hamming8-2	128.0129	128.0000	2,407	1,118		1,065
hamming8-4	86.8582	85.3333	41,376	22,398		8,219
johnson16-2-4	40.0239	40.0003	972	684		522
johnson8-2-4	9.3875	9.3333	12	7		5
johnson8-4-4	23.3814	23.3333	177	98		69
keller4	57.8079	57.0001	5,275	4,960		1,865
p_hat300-1	102.1317	100.0013	198,263	98,646		35,030
p_hat300-2	102.7619	100.0036	129,663	61,445		17,471
p_hat300-3	102.8840	100.0016	64,609	26,660		7,005
san200_07_1	72.2302	66.6667	12,224	7,550		2,054
san200_07_2	66.9081	66.6670	72,026	7,306		3,846
san200_09_1	70.0103	70.2033	14,733	4,746	11,193	
san200_09_2	66.9371	66.6667	33,923	3,462		1,363
san200_09_3	66.9232	66.6668	31,469	2,528		1,353
sanr200_0_7	69.7920	66.6667	14,061	7,920		2,695
sanr200_0_9	66.8795	66.6667	32,368	3,459		1,410
Erdös–Turán ($n = 60$)	34.3553	37.5214	508	863	123	
Erdös–Turán ($n = 70$)	40.1841	43.9156	589	1,591	202	
Erdös–Turán ($n = 80$)	45.9178	49.3888	1,468	3,052	488	
Erdös–Turán ($n = 90$)	51.5557	57.4332	2,169	3,927	714	
Erdös–Turán ($n = 100$)	57.3410	62.7492	3,273	5,458	1,312	
Erdös–Turán ($n = 110$)	63.1194	67.4054	5,858	9,752	2,337	
Erdös–Turán ($n = 120$)	68.7601	71.9690	7,066	16,852	3,629	
Erdös–Turán ($n = 130$)	74.5256	79.6999	14,909	19,356	6,113	
Erdös–Turán ($n = 140$)	80.1107	88.7548	18,311	35,119	6,833	
Erdös–Turán ($n = 150$)	85.9341	96.5941	24,155	85,492	8,078	

of the augmented Lagrangian algorithm on the Erdös–Turán instances. However, the resultant timings were still not competitive with the subgradient algorithm on these instances. We remark that such convergence issues were much less prevalent in the semidefinite computation using augmented Lagrangian, as Table 5 will demonstrate.

We also experimented with our own implementation of a subgradient method to optimize over $N_+(K)$ but found that standard stepsize strategies for subgradient methods did not seem appropriate for the positive semidefinite multiplier S. Consequently, convergence was difficult to achieve in this case.

4.4.3. Maximum stable set. In Table 4, we give the dual bounds obtained by the augmented Lagrangian algorithm for the maximum stable set instances. The bounds and times (in seconds) to achieve those bounds are listed under N for (1) and N_+ for (2). When prefixed to the bound, the symbol (‡) indicates that the alternate stopping criterion for our method was enforced, i.e., σ had grown too large before primal feasibility had been obtained.

TABLE 4

Results on the maximum stable set problem, comparing bounds achieved by the N and N_+ procedures. Timings (in seconds) are also given. When prefixed to the bound, the symbol (‡) indicates that the alternate stopping criterion was enforced.

						Bound		Time(s)	
Name	$\|V\|$	$\|E\|$	α	ϑ_+	ϑ	N	N_+	N	N_+
brock200_1	200	5066	21	27.2	27.5	66.6668	27.9874	5,119	28,590
brock200_2	200	10024	12		14.2	66.6671	‡ 17.0805	11,174	67,302
brock200_3	200	7852	15		18.8	66.6670	‡ 20.7928	8,674	51,665
brock200_4	200	6811	17	21.1	21.3	66.6681	22.8004	6,765	43,433
c-fat200-1	200	18366	12	12.0	12.0	66.6667	‡ 14.9735	18,125	126,103
c-fat200-2	200	16665	24	24.0	24.0	66.6686	24.0877	13,861	83,691
c-fat200-5	200	11427	58		60.3	66.6671	58.1798	16,774	44,483
hamming6-2	64	192	32	32.0	32.0	32.0000	32.0000	11	15
hamming6-4	64	1312	4	4.0	5.3	21.3334	‡ 4.5460	134	1,416
hamming8-2	256	1024	128	128.0	128.0	128.0000	128.0001	640	728
hamming8-4	256	11776	16	16.0	16.0	85.3340	‡ 20.5442	21,723	90,169
johnson8-2-4	28	168	4	4.0	4.0	9.3333	4.0052	6	59
johnson8-4-4	70	560	14	14.0	14.0	23.3333	14.0076	90	479
johnson16-2-4	120	1680	8	8.0	8.0	40.0001	‡ 10.2637	763	3,140
keller4	171	5100	11	13.5	14.0	57.0001	‡ 15.4119	4,910	19,319
MANN-a9	45	72	16		17.5	18.0000	17.1790	5	50
p_hat300-1	300	33917	8	10.0	10.1	100.0003	‡ 18.6697	129,437	322,287
p_hat300-2	300	22922	25		27.0	100.0004	‡ 30.1066	83,142	244,428
p_hat300-3	300	11460	36		41.2	100.0008	43.3282	33,554	101,995
san200_07_1	200	5970	30	30.0	30.0	66.6672	30.7071	7,995	31,049
san200_07_2	200	5970	18	18.0	18.0	66.6670	‡ 20.0176	7,710	37,102
san200_09_1	200	1990	70	70.0	70.0	70.1613	70.5464	4,840	6,947
san200_09_2	200	1990	60	60.0	60.0	66.6667	60.7250	3,484	6,977
san200_09_3	200	1990	44	44.0	44.0	66.6678	44.4080	2,538	12,281
sanr200_07	200	6032	18	23.6	23.8	66.6672	24.9716	7,946	36,576
sanr200_09	200	2037	42		49.3	66.6667	49.3156	3,335	9,428

In order to gauge the quality of the bounds in Table 4, we also include the size α of the maximum stable set (either obtained from the literature or computed using the IP solver of CPLEX) as well as the Lovász ϑ number of the graph (obtained by the algorithm of Burer and Monteiro [9]) and Schrijver's strengthening ϑ_+ of ϑ (obtained from Kim Toh (personal communication)). Note that the value of ϑ_+ was not available for all instances. The numbers ϑ and ϑ_+ are polynomial-time computable upper bounds on α, which are obtained by solving two related semidefinite programs. Theoretically, the N_+ bound is at least as strong as ϑ_+, which is at least as strong as ϑ, but computationally, ϑ takes less time to compute than ϑ_+, which in turn takes much less time than the N_+ bound (a reasonable estimate is roughly one order of magnitude less).

The results indicate that, at least on the majority of problems in this sample of graphs, the computed N_+ bound is significantly tighter than the computed N bound. Moreover, it is a real challenge for the augmented Lagrangian algorithm to optimize the N_+ bound fully, which is evidenced by the fact that the computed value for the N_+ bound is actually higher than ϑ on most instances. This demonstrates the possibility for further improvement of our optimization technique, perhaps by more sophisticated guidelines for choosing the number of cycles of block coordinate descent or for updating σ.

Nevertheless, we stress that our overall intention is to show that (1) and (2) can be (approximately) solved for general 0-1 integer programs. Accordingly, Table 4 serves to demonstrate that, given a specific problem (such as the stable set problem),

TABLE 5

Results on the problem of Erdös and Turán, comparing bounds achieved by the N and N_+ procedures. Timings (in seconds) are also given. Lower and upper bounds (LB and UB) were achieved by running the IP solver of CPLEX 8.1 for at most 15,000 seconds.

			Bound		Time(s)	
n	LB	UB	N	N_+	N	N_+
60	19	19	34.6688	31.4183	354	653
70	20	20	40.7446	36.6120	778	1,062
80	22	27	46.7352	41.6060	1,686	1,676
90	24	32	53.0720	46.5339	2,550	3,164
100	25	37	59.6474	51.8541	2,810	4,295
110	27	44	65.8540	57.0635	5,387	6,773
120	30	48	71.1935	62.1836	10,932	9,659
130	32	55	77.2897	67.2867	11,023	13,459
140	32	60	82.9586	72.3553	21,164	18,239
150	32	66	89.5872	77.2238	34,180	23,450

our method allows one to compute the N and N_+ bounds and hence to evaluate their quality.

4.4.4. Problem of Erdös and Turán. Table 5 lists the results of our algorithm on the Erdös–Turán instances, and the details of the table are the same as for Table 4. In order to assess the quality of the computed N and N_+ bounds, we ran CPLEX's IP solver on each instance for at most 15,000 seconds and report the best lower bound (LB) and best upper bound (UB) achieved.

Three things are interesting to note. First, the N_+ bounds are a significant improvement over the bounds reported by Dash [14] (for example, Dash gives a bound of 87.6 for $n = 150$). Second, the times for computing the N and N_+ bounds are not dramatically different from one another, and in fact, on some of the largest problems, the N_+ bound actually takes less time to compute. Third, the upper bound calculated by CPLEX's IP solver (after 15,000 seconds) is significantly tighter than the computed N and N_+ bounds, which puts into perspective the time dedicated to calculating N and N_+. Even still, as with the stable set instances in the previous subsection, the results demonstrate that the augmented Lagrangian greatly improves our capabilities for actually computing the Lovász–Schrijver bounds.

4.4.5. Market share. Besides solving both the N and N_+ relaxations of our market share instances, we also ran the default CPLEX IP solver for 15,000 on each instance in order to obtain lower and upper bounds. The results appear in the following table (recall that these are minimization problems).

			Bound		Time(s)	
Instance	lb	ub	N	N_+	N	N_+
Markshare1 (6×50)	0	3	0.0000	0.0000	20	17
Markshare2 (7×60)	0	11	0.0000	0.0000	37	31

Since the problems are quite small, optimizing over N and N_+ is done very quickly. However, neither relaxation improves on the basic LP bound of 0. This demonstrates that lift-and-project operators are not always able to strengthen the LP relaxation of an IP.

4.4.6. Quadratic assignment. Tables 6 and 7 list our results on the quadratic assignment instances, and the details of the table are similar to Table 4, except for

TABLE 6

Results on the quadratic assignment problem (I)*, comparing gaps achieved by three previous bounding techniques and the two techniques of this paper, N and N_+. Timings (in seconds) for N and N_+ are also given. Gaps are calculated with respect to the feasible value listed, which is known to be optimal unless the symbol (†) is prefixed. When prefixed to the gap, the symbol (‡) indicates that the alternate stopping criterion was enforced.*

		Gap (%)					Time(s)	
Name	Feas Val	GLB	KCCEB	RSB	N	N_+	N	N_+
bur026a	5,426,670	2.05	1.29		1.19	0.19	17,929	60,397
bur026b	3,817,852	2.70	1.69		1.57	0.22	18,248	53,749
bur026c	5,426,795	2.11	1.21		1.08	0.18	17,932	67,918
bur026d	3,821,225	2.87	1.64		1.51	0.21	18,064	69,804
bur026e	5,386,879	1.48	0.97		0.86	‡ 0.20	18,530	71,917
bur026f	3,782,044	1.99	1.27		1.14	0.10	18,195	78,748
bur026g	10,117,172	1.37	0.61		0.51	0.15	19,214	73,082
bur026h	7,098,658	1.77	0.75		0.63	0.09	18,385	70,939
chr012a	9,552	24.15	1.09		0.00	0.00	330	352
chr012b	9,742	26.65	0.00		0.00	0.00	293	363
chr012c	11,156	28.50	4.28		0.00	0.00	376	410
chr015a	9,896	43.16	11.65		4.35	0.14	1,415	1,461
chr015b	7,990	41.76	11.94		0.01	0.00	1,467	1,140
chr015c	9,504	35.13	3.80		0.00	0.00	901	1,188
chr018a	11,098	38.92	9.56		3.28	0.00	3,357	3,947
chr018b	1,534	0.00	0.00		0.00	0.13	1,277	6,239
chr020a	2,192	1.92	1.19		1.00	0.18	3,878	6,307
chr020b	2,298	4.44	1.70		0.87	0.13	4,030	9,778
chr020c	14,142	39.18	7.68		0.05	0.02	5,046	6,812
chr022a	6,156	3.77	0.58		0.24	0.03	6,762	12,711
chr022b	6,194	4.17	0.65		0.26	0.19	7,070	18,377
chr025a	3,796	27.16	4.98		0.53	0.37	13,901	33,402
els019	17,212,548	30.45	5.45		2.00	0.04	5,879	9,821
esc016a	68	44.12	39.71	13.24	29.41	5.88	452	1,195
esc016b	292	24.66	6.16	1.37	4.79	0.68	474	1,103
esc016c	160	48.13	43.13	11.25	26.25	3.75	538	1,981
esc016d	16	81.25	75.00	50.00	75.00	18.75	397	1,520
esc016e	28	57.14	57.14	17.86	50.00	3.57	349	1,318
esc016g	26	53.85	53.85	23.08	46.15	3.85	351	1,315
esc016h	996	37.25	29.32	2.61	29.32	1.91	618	1,369
esc016i	14	100.00	100.00	35.71	100.00	14.29	160	1,601
esc016j	8	87.50	75.00	12.50	75.00	0.00	345	1,331
esc032a	† 130	73.08			69.23	20.77	10,503	143,084
esc032b	† 168	42.86			42.86	21.43	5,308	130,393
esc032c	† 642	45.48			40.65	4.05	15,223	114,053
esc032d	† 200	47.00			44.00	4.50	10,767	117,556
esc032e	2	100.00			100.00	0.00	498	143,593
esc032f	2	100.00			100.00	0.00	496	144,820
esc032g	6	100.00			100.00	‡ 0.00	506	107,683
esc032h	† 438	41.32			33.79	3.20	15,031	140,406
had012	1,652	7.02	2.00	0.54	1.82	0.00	363	244
had014	2,724	8.52	2.31	0.33	2.13	0.00	818	1,092
had016	3,720	9.73	4.49	0.56	4.30	0.13	1,396	2,551
had018	5,358	10.86	5.23	0.77	5.06	0.11	2,518	4,966
had020	6,922	10.92	5.13	0.53	4.98	0.16	4,119	8,835

the following: (i) the best-known feasible value for the QAP is listed such that, if the symbol (†) is *not* prefixed, then the feasible value is actually optimal, while (†) is present when the value is not known to be optimal; (ii) instead of listing dual bounds, we give optimality gaps, i.e.,

$$\text{gap} = \frac{\text{feas val} - \text{bound}}{\text{feas val}} \times 100\%.$$

TABLE 7

Results on the quadratic assignment problem (II)*, comparing gaps achieved by three previous bounding techniques and the two techniques of this paper, N and N_+. Timings (in seconds) for N and N_+ are also given. Gaps are calculated with respect to the feasible value listed, which is known to be optimal unless the symbol (†) is prefixed. When prefixed to the gap, the symbol (‡) indicates that the alternate stopping criterion was enforced.*

		Gap (%)					Time(s)	
Name	Feas Val	GLB	KCCEB	RSB	N	N_+	N	N_+
kra030a	88,900	23.10	15.00	12.86	14.51	2.50	30,618	128,883
kra030b	91,420	24.45	16.61	11.23	16.10	4.07	30,648	136,187
kra032	88,700	24.02		10.19	16.06	3.51	40,543	225,053
lipa020a	3,683	0.00	0.00		0.00	0.00	1,337	3,699
lipa020b	27,076	0.00			0.00	0.00	1,556	4,276
lipa030a	13,178	0.00	0.00		0.01	0.02	20,584	121,748
lipa030b	151,426	0.00			0.00	0.00	10,783	77,868
nug012	578	14.71	9.86	3.63	9.52	1.73	315	469
nug014	1,014	15.98		2.17	8.97	0.39	837	1,093
nug015	1,150	16.26	10.17	2.43	9.48	0.78	1,043	1,543
nug016a	1,610	18.39	11.86	2.48	11.49	0.75	1,456	2,421
nug016b	1,240	17.58	12.74	4.19	12.26	1.69	1,338	2,143
nug017	1,732	19.86	13.51	3.64	13.05	1.44	1,932	3,379
nug018	1,930	19.48	14.20	4.04	13.89	1.92	2,393	4,932
nug020	2,570	19.96	15.45	4.63	15.14	2.49	3,772	7,577
nug021	2,438	24.82	17.64	4.72	17.10	2.46	5,150	13,791
nug022	3,596	30.95	21.19	4.34	20.88	2.34	6,110	18,074
nug024	3,488	23.28	18.09	5.10	17.75	2.61	8,581	25,887
nug025	3,744	23.37	18.16	5.58	17.76	3.29	10,457	31,082
nug027	5,234	29.29		5.14	21.63	2.33	14,406	69,574
nug028	5,166	26.71		5.13	21.58	2.92	16,501	81,564
nug030	6,124	25.39	21.86	5.24	21.60	3.10	21,762	127,011
rou012	235,528	14.12	5.09	5.03	4.79	0.11	566	759
rou015	354,210	15.71	8.64	5.91	8.30	1.13	1,377	2,027
rou020	725,522	17.31	11.59	8.50	11.36	4.19	5,112	10,997
scr012	31,410	11.31	5.96	6.65	5.07	0.00	489	496
scr015	51,140	12.52	5.07	4.51	3.70	0.00	1,403	1,197
scr020	110,030	30.23	14.12	13.66	13.58	3.88	5,182	10,564
ste036a	9,526	25.22	17.49		16.80	5.31	68,877	427,884
ste036b	15,852	45.41			30.65	7.61	73,431	358,981
ste036c	8,239,110	22.40			14.70	‡ 3.92	89,618	377,109
tai012a	224,416	12.70	1.61	0.73	1.02	0.00	563	414
tai012b	39,464,925	75.20	22.66		‡ 20.04	‡ 1.01	845	1,039
tai015a	388,214	15.64	9.34	6.04	9.12	2.86	1,402	2,022
tai015b	51,765,268	78.28	0.52		0.53	0.35	2,070	3,199
tai017a	491,812	16.08	10.23	8.23	10.01	3.11	2,521	4,414
tai020a	703,482	17.46	12.34	9.41	12.11	4.52	5,056	10,418
tai020b	122,455,319	88.40	24.44		‡ 23.06	‡ 4.11	7,981	15,591
tai025a	1,167,256	17.55	13.82	10.79	14.30	4.66	13,382	39,565
tai025b	344,355,646	86.15	56.93		‡ 55.82	‡ 11.70	16,855	65,262
tai030a	1,818,146	17.24	13.91	9.13	13.74	6.12	27,299	155,797
tai030b	† 637,117,113	93.57	78.46		‡ 78.46	‡ 18.47	31,333	247,114
tai035a	† 2,422,002	19.44	16.66		16.52	8.48	60,604	329,608
tai035b	† 283,315,445	88.49	64.05		‡ 66.40	‡ 15.42	63,888	430,914
tho030	149,936	39.59	33.40	9.26	32.84	4.75	28,180	99,265

A missing entry from the table (applicable only in the case of KCCEB and RSB) indicates that the gap was not available in the literature. In addition, note that the number contained in the names of the QAP instances is the size n of the QAP; for example, the instance *bur*026*a* has $n = 26$.

Though theory predicts that the KCCEB and N gaps should equal one another, we do see some discrepancies in Tables 6 and 7, probably because of numerical differences in the algorithms. Typically, N is slightly better than KCCEB, but in compar-

TABLE 8

Results on the quadratic assignment problem (III)*, comparing gaps achieved by the bounding technique of Hahn et al.* [20] *(denoted by H) and the semidefinite technique of this paper,* N_+*. The problem instances are a subset of those shown in Tables* 6 *and* 7*.*

	Gap (%)	
Name	H	N_+
had16	0.00	0.13
had18	0.00	0.11
had20	0.00	0.16
kra30a	2.98	2.50
kra30b	4.72	4.07
nug12	0.00	1.73
nug15	0.00	0.78
nug20	3.23	2.49

	Gap (%)	
Name	H	N_+
nug30	6.11	3.10
rou15	0.00	1.13
rou20	3.60	4.19
tai20a	3.93	4.52
tai25a	6.48	4.66
tai30a	7.25	6.12
tho30	8.82	4.75

TABLE 9

Average number of iterations of the augmented Lagrangian algorithm over all problems on three of the four problem classes, for both $N(K)$ *and* $N_+(K)$*.*

	Stable	Erdös–Turán	Market Share	QAP
$N(K)$	371	760	251	2,796
$N_+(K)$	1,648	1,197	205	2,812

ison with timings reported in [22], the calculation of N takes more time. We should point out, however, that the algorithm used to compute KCCEB exploits the structure of QAP to a great extent (more than just the reduction of $2n+1$ subproblems to $n+1$ mentioned previously) and does not appear to be generalizable to other 0-1 problems. On the other hand, the augmented Lagrangian method can be applied to any 0-1 problem.

The tables also indicate that the N_+ gap is significantly tighter than the RSB gap. In fact, on a number of relatively small problems, the N_+ gap is 0, indicating that (2) solves the QAP exactly. Previous to these results, RSB had provided the strongest known bounds for problems in QAPLIB. Only partial timing results are given by Rendl and Sotirov [37], and so we are unable to make precise timing comparisons with RSB.

After the initial appearance of this paper, Hahn (personal communication) announced bounds for several of the instances in Tables 6 and 7, which were calculated with an algorithm of Hahn et al. [20] (but were not specifically presented there). The bounds are obtained by solving the second-level Sherali–Adams linear program for the QAP. We present the bounds compared with our semidefinite bounds in Table 8.

4.4.7. Some further details. The tables of the previous subsections include bounds and timings for the augmented Lagrangian runs. Some additional aggregate information on the number of iterations is given in Table 9. Here, "number of iterations" refers to the number of outermost loops in the augmented Lagrangian algorithm—indexed by k in the statement of Algorithm 1.

For problems with m much larger than n, the most computationally intensive part of the augmented Lagrangian algorithm (applied to both $N(K)$ and $N_+(K)$) is using CPLEX to solve the convex QP subproblems corresponding to the columns of Y and Z. We consider all stable and Erdös–Turán instances, which have $n \leq 300$ and often $m \approx \mathcal{O}(n^2)$, to be of this type. On the other hand, in the case of $N_+(K)$, the $\mathcal{O}(n^3)$ eigenvalue decompositions required by the semidefinite projections (two per iteration) will constitute a large part of the computation, especially for large n and small m. For example, for the QAP instances, n ranges from 144 to 1,225 and

$m = 3n + 2$. This helps to explain the fact that, while the average number of QAP iterations for $N(K)$ and $N_+(K)$ shown in Table 9 are not significantly different, the corresponding timings in Tables 6 and 7 are quite different.

5. Conclusions. In this paper, we propose a novel method to apply dual decomposition to the lift-and-project relaxations of binary integer programs introduced by Lovász and Schrijver [32]. We believe that this is some of the first work that focuses on developing effective tools for solving these very large relaxations. Rather than using subgradient techniques to solve the dual, we show how to use an augmented Lagrangian technique to obtain bounds from these relaxations in both the LP and semidefinite case. We extend a result by Poljak and Tret′jakov [35] to show that in the case of linear, conic programs, the augmented Lagrangian approach can use a constant penalty parameter and still guarantee convergence. Through extensive computational testing, we demonstrate the ability of this technique to outperform standard LP, SDP, and subgradient methods for various classes of problems. For some instances, such as QAP, the bounds computed from these relaxations are the tightest known to date.

As part of our future work in this area, we will study the possibility of using special purpose algorithms to solve the QP subproblems, especially in cases such as QAP where the constraints of the subproblems are simply a homogenization of the assignment polytope. We also intend to examine how these techniques may be used to yield tight relaxations of problems with a mix of binary and continuous variables and of continuous nonconvex QP's.

In addition to introducing some of the first effective solution techniques for linear and positive semidefinite lift-and-project relaxations, the success of this approach also demonstrates the applicability of augmented Lagrangian techniques even for linear, conic problems. We believe it will be interesting to investigate how well this technique performs on other large-scale linear, conic problems with block-angular structure.

Acknowledgments. The authors are in debt to two anonymous referees for careful and thorough comments that have greatly improved the paper.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed., Society for Industrial and Applied Mathematics, Philadelphia, 1999.

[2] K. M. Anstreicher, *Recent advances in the solution of quadratic assignment problems*, Math. Program., 97 (2003), pp. 27–42.

[3] E. Balas, *Disjunctive programming*, Ann. Discrete Math., 5 (1979), pp. 3–51.

[4] E. Balas, S. Ceria, and G. Cornuéjols, *A lift-and-project cutting plan algorithm for mixed* 0–1 *programs*, Math. Program., 58 (1993), pp. 295–324.

[5] F. Barahona and R. Anbil, *The volume algorithm: Producing primal solutions with a subgradient method*, Math. Program., 87 (2000), pp. 385–399.

[6] D. P. Bertsekas, *Nonlinear Programming*, 1st ed., Athena Scientific, Belmont, MA, 1995.

[7] D. Bienstock and M. Zuckerberg, *Subset algebra lift operators for* 0–1 *integer programming*, SIAM J. Optim., 15 (2004), pp. 63–95.

[8] B. Borchers, *CSDP, a C library for semidefinite programming*, Optim. Methods Softw., 11/12 (1999), pp. 613–623.

[9] S. Burer and R. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Math. Program., 95 (2003), pp. 329–357.

[10] R. E. Burkard, S. Karisch, and F. Rendl, *QAPLIB—a quadratic assignment problem library*, J. Global Optim., 10 (1997), pp. 391–403.

[11] S. Ceria and G. Pataki, *Solving integer and disjunctive programs by lift-and-project*, in Proceedings of the Sixth International IPCO Conference, Lecture Notes in Comput. Sci. 1412, R. E. Bixby, E. A. Boyd, and R. Z. Rios-Mercato, eds., 1998, pp. 271–283.

[12] W. Cook and S. Dash, *On the matrix-cut rank of polyhedra*, Math. Oper. Res., 26 (2001), pp. 19–30.
[13] G. Cornuéjols and M. Dawande, *A class of hard small* 0–1 *programs*, INFORMS J. Comput., 11 (1999), pp. 205–210.
[14] S. Dash, *On the Matrix Cuts of Lovász and Schrijver and Their Use in Integer Programming*, Ph.D. thesis, Rice University, Houston, TX, 2001.
[15] See the website: http://dimacs.rutgers.edu/Challenges/.
[16] W. S. Dorn, *Duality in quadratic programming*, Quart. Appl. Math., 18 (1960/1961), pp. 155–162.
[17] P. C. Gilmore, *Optimal and suboptimal algorithms for the quadratic assignment problem*, J. Soc. Indust. Appl. Math., 10 (1962), pp. 305–313.
[18] M. X. Goemans and L. Tunçel, *When does the positive semidefiniteness constraint help in lifting procedures?*, Math. Oper. Res., 26 (2001), pp. 796–815.
[19] R. E. Gomory, *An algorithm for integer solutions to linear programs*, in Recent Advances in Mathematical Programming, R. Graves and P. Wolfe, eds., McGraw-Hill, New York, 1963, pp. 269–302.
[20] P. M. Hahn, W. L. Hightower, T. A. Johnson, M. Guignard-Spielberg, and C. Roucairol, *A level-2 reformulation-linearization techinque bound for the quadratic assignment problem*, manuscript, University of Pennsylvania, Philadelphia, PA, 2001.
[21] ILOG, Inc., *ILOG CPLEX* 8.1*, User's Manual*, 2002.
[22] S. E. Karisch, E. Çela, J. Clausen, and T. Espersen, *A dual framework for lower bounds of the quadratic assignment problem based on linearization*, Computing, 63 (1999), pp. 351–403.
[23] K. C. Kiwiel, *Methods of Descent for Nondifferentiable Optimization*, Springer, Berlin, 1985.
[24] J. B. Lasserre, *An explicit exact SDP relaxation for nonlinear* 0–1 *programs*, in Lecture Notes in Comput. Sci., 2081, K. Aardal and A. M. H. Gerards, eds., 2001, pp. 293–303.
[25] M. Laurent, *Tighter linear and semidefinite relaxations for max-cut based on the Lovász–Schrijver lift-and-project procedure*, SIAM J. Optim., 12 (2001/2002), pp. 345–375.
[26] M. Laurent, *A comparison of the Sherali–Adams, Lovász–Schijver, and Lasserre relaxation for* 0–1 *programming*, SIAM J. Optim., 28 (2003), pp. 470–496.
[27] M. Laurent and F. Rendl, *Semidefinite programming and integer programming*, Technical report PNA-R0210, CWI, Amsterdam, April 2002. To appear as chapter of the *Handbook on Discrete Optimization* edited by K. Aardal, G. Nemhauser and R. Weismantel.
[28] E. L. Lawler, *The quadratic assignment problem*, Management Sci., 9 (1962/1963), pp. 586–599.
[29] C. Lemaréchal, *Nonsmooth optimization and descent methods*, Technical report, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1977.
[30] L. Lipták and L. Tunçel, *The stable set problem and the lift-and-project rank of graphs*, Math. Program., 98 (2003), pp. 319–353.
[31] R. Lougee-Heimer, *The Common Optimization INterface for Operations Research*, IBM Journal of Research and Development, 47 (2003), pp. 57–66; also available online from www.coin-or.org.
[32] L. Lovász and A. Schrijver, *Cones of matrices and set-functions, and* 0–1 *optimization*, SIAM J. Optim., 1 (1991), pp. 166–190.
[33] MIPLIB, 2003. http://miplib.zib.de/.
[34] J. Nocedal and S. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[35] B. T. Poljak and N. V. Tret′jakov, *A certain iteration method of linear programming and its economic interpretation*, Èkonom. i Mat. Metody, 8 (1972), pp. 740–751.
[36] QAPLIB, http://www.seas.upenn.edu/qaplib/.
[37] F. Rendl and R. Sotirov, *Bounds for the quadratic assignment problem using the bundle method*, Technical report, Department of Mathematics, University of Klagenfurt, Klagenfurt, Austria, 2003.
[38] H. D. Sherali and W. P. Adams, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM J. Discrete Math., 3 (1990), pp. 411–430.
[39] H. D. Sherali and W. P. Adams, *A Reformulation-Linearization Technique (RLT) for Solving Discrete and Continuous Nonconvex Problems*, Kluwer, Dordrecht, The Netherlands, 1997.
[40] H. D. Sherali, B. Özdaryal, W. P. Adams, and N. Attia, *On using exterior penalty approaches for solving linear programming problems*, Comput. Oper. Res., 28 (2001), pp. 1049–1074.
[41] T. Stephen and L. Tunçel, *On a representation of the matching polytope via semidefinite liftings*, Math. Oper. Res., 24 (1999), pp. 1–7.