Chapter 1

Introduction

1.1 Motivation

The simplex method was invented by George B. Dantzig in 1947 for solving linear programming problems. Although, this method is still a viable tool for solving linear programming problems and has undergone substantial revisions and sophistication in implementation, it is still confronted with the exponential worst-case computational effort. The question of the existence of a polynomial time algorithm was answered affirmatively by Khachian [16] in 1979. While the theoretical aspects of this procedure were appealing, computational effort illustrated that the new approach was not going to be a competitive alternative to the simplex method. However, when Karmarkar [15] introduced a new low-order polynomial time interior point algorithm for solving linear programs in 1984, and demonstrated this method to be computationally superior to the simplex algorithm for large, sparse problems, an explosion of research effort in this area took place. Today, a variety of highly effective variants of interior point algorithms exist (see Terlaky [32]). On the other hand, while exterior penalty function approaches have been widely used in the context of nonlinear programming, their application to solve linear programming problems has been comparatively less explored. The motivation of this research effort is to study how several variants of exterior penalty function methods, suitably modified and fine-tuned, perform in comparison with each other, and to provide insights into their viability for solving linear programming problems.

Many experimental investigations have been conducted for studying in detail the simplex method and several interior point algorithms. Whereas these methods generally perform well and are sufficiently adequate in practice, it has been demonstrated that solving linear programming problems using a simplex based algorithm, or even an interior-point type of procedure, can be inadequately slow in the presence of complicating constraints, dense coefficient matrices, and ill-conditioning. Yet, algorithms based on interior point methods for solving linear programming problems have gained popularity due to some recent techniques which eliminate the ill-conditioning that is inherent with barrier function approaches (see Terlaky [32]). However, there is comparatively almost no evidence on the performance of exterior penalty function approaches for solving linear programs. Our aim is to shed some light on this gap in the literature in this thesis.

Consider the problem

$$\begin{aligned} \text{P: Minimize} \quad & f(x) \\ \text{subject to} \quad & g(x) \le 0 \\ & h(x) = 0 \end{aligned}$$

where f: $R^n \to R$, g: $R^n \to R^{m_1}$, and h: $R^n \to R^{m_2}$ are all continuously differentiable functions.

The augmented Lagrangian function L_μ for the above problem is given by:

$$L_\mu(x,s,\lambda,\pi) = f(x) + \left[\lambda^t h(x) + \frac{\mu}{2} \|h(x)\|^2 \right] + \sum_{i=1}^{m_1} \left[\pi_i \left(g_i(x) + s_i^2 \right) + \frac{\mu}{2} \left| g_i(x) + s_i^2 \right|^2 \right] \tag{1.1}$$

where μ is a positive penalty parameter, λ and π are the Lagrange multiplier estimates for the equality constraints and inequality constraints, respectively, $g(x) \equiv (g_i(x) \text{ for } i = 1,...m_1)$, and s_i is the slack associated with the i^{th} inequality constraint. Note that L_μ in (1.1) can be readily minimized over the slack variables s_i, i=1,…,m, and hence stated essentially in terms of the x variables, given (λ,π). The function $L_\mu(x,\lambda,\pi)$ below refers to this function.

There are two ways by which the unconstrained minimization of $L_\mu(\cdot,\lambda,\pi)$ can possibly yield an optimal solution x* (see Bazaraa et al. [5]):

(1) By driving the penalty parameter μ to infinity in the limit. The cost of infeasibility is high when μ is high, tending to make the unconstrained minima of $L_\mu(\cdot,\lambda,\pi)$ nearly feasible. Since $L_\mu(x,\lambda,\pi) = f(x)$ for a feasible solution x, we expect that $L_\mu(x,\lambda,\pi) \cong f(x)$ for near feasible solutions. Hence an acceptable approximation to x* can be obtained by the unconstrained minimization of $L_\mu(\cdot,\lambda,\pi)$ when μ is high. A rigorous justification of this approach follows from the description of quadratic penalty functions given in Bazaraa et al. [5].

(2) By making (λ,π) approach (λ*,π*), an optimal set of Lagrange multipliers (assumed to exist). For a certain finite penalty parameter μ, and for some $\gamma > 0$ and $\varepsilon > 0$, under certain sufficiency conditions (see Bazaraa et al. [5] and Bertsekas [8]) we have

$$L_\mu(x^*,\lambda^*,\pi^*) \le L_\mu(x,\lambda^*,\pi^*) - \frac{\gamma}{2}\|x - x^*\|^2 \quad \forall x \text{ with } \|x - x^*\| < \varepsilon \tag{1.2}$$

so that x* is a strict local minimum of the augmented Lagrangian $L_\mu(\cdot,\lambda^*,\pi^*)$. The sufficiency conditions for (1.2) to hold true are:

(a) f, g, and h that define the objective function and the inequality and equality constraints of the primal problem, respectively, must be twice differentiable.

(b) $d^t \nabla^2 L_\mu(x^*,\lambda^*,\pi^*) d > 0$, for all $d \in S$, where S is defined by

$$S = \left\{ \begin{array}{l} d \neq 0 : \nabla g_i(x^*)^t d = 0, \text{ for active inequality constraints,} \\ \qquad \nabla g_i(x^*)^t d \le 0, \text{ for inactive inequality constraints,} \\ \qquad \nabla h_i(x^*)^t d = 0, \text{ for } i = 1,..., m_2. \end{array} \right\} \tag{1.3}$$

Hence, this in turn implies that if (λ,π) is close to (λ^*,π^*), a good approximation to x^* can be achieved through the unconstrained minimization of $L_\mu(\cdot,\lambda,\pi)$.

In this thesis, three exterior point algorithms for solving linear programming problems are studied. These methods are an active set l_2 penalty approach (ASL2), an inequality-equality based l_2 penalty approach (IEL2), and an augmented Lagrangian approach (ALAG). The ASL2 method employs a quadratic penalty function that is based on a dynamically updated set of predicted active constraints. The penalty function in IEL2 directly incorporates all of the inequality constraints and is piecewise quadratic and differentiable. The ALAG approach augments the latter function by using a Lagrangian term in order to circumvent the need to take the penalty parameter to infinity for recovering an optimal solution.

The linear programming problem considered in this development is stated in the following form.

$$\begin{aligned} &\text{Minimize} && cx \\ &\text{subject to} && A_i x \geq b_i \quad \text{for } i \in I_1 \\ & && A_i x = b_i \quad \text{for } i \in I_2 \\ & && x \in \Omega \equiv \{x : 0 \leq x \leq u\}, \end{aligned}$$

where $x \in R^n$, c and A_i for $i \in I \equiv I_1 \cup I_2$ are row vectors in R^n with $|I_1| \equiv m_1$, $|I_2| \equiv m_2$, and where we denote $|I| \equiv m = m_1 + m_2$. The variable bounding inequalities absorbed in the set Ω are treated separately and are not accommodated within the various penalty functions.

The augmented Lagrangian function L_μ for the above problem is given as follows, after minimizing the aforementioned form of this function over the slacks in the inequality constraints (see Bazaraa et al. [5]):

$$L_\mu(x,\lambda,\pi) = cx + \sum_{i \in I_2} \left[\lambda_i (b_i - A_i x) + \frac{1}{2} \mu_i (b_i - A_i x)^2 \right] + \frac{1}{2} \sum_{i \in I_1} \mu_i \max^2 \left\{ 0, \frac{\pi_i}{\mu_i} + b_i - A_i x \right\}. \tag{1.4}$$

The augmented Lagrangian function approach (ALAG) benefits from using the Lagrangian multiplier estimates to find a good approximation to the optimal solution with a finite penalty parameter. Moreover, unlike l_1 penalty methods that share the same property of achieving an optimal solution with a finite penalty parameter, ALAG also enjoys the property of being differentiable. Several versions of the ALAG approach are investigated since this approach is expected to behave better than the approaches using the simpler quadratic penalty function.

Furthermore, note that IEL2 and ASL2 penalty functions can be viewed as a special case of ALAG. The inequality-equality based l_2 penalty approach (IEL2) is derived by setting the Lagrange multiplier estimates to zero and selecting an identical penalty parameter for each

constraint (the latter description could be relaxed). As a result, the IEL2-based approach needs to take the penalty parameter to infinity to recover an optimal solution. The active set l_2 penalty approach (ASL2) also takes $(\lambda,\pi) = (0,0)$ at each iteration. In addition, ASL2 introduces the concept of an active set of constraints that are predicted to be binding at an optimal solution. The penalty term is added only for those constraints that are predicted to be active at an optimal solution. Hence, ASL2 is a further specialized case of IEL2, wherein all considered constraints at any stage are treated uniformly as equality restrictions.

It should be noted that the values of the Lagrange multipliers are not known at the beginning of a solution procedure. Therefore, initial estimates for the Lagrange multiplier vector and the rules for updating this vector must be established. This topic will be considered in Chapter 3.

Several related algorithmic strategies are explored and tested on a set of randomly generated, degenerate problems, as well as on a standard collection of NETLIB problems. The randomly generated problems are partitioned into two sets to obtain a better understanding of how each algorithm behaves under different conditions. The first set of the randomly generated problems is comprised of general linear programming problems, whereas the second set contains linear programming problems having equality constraints only. Both the randomly generated problems and the NETLIB test problems are chosen to represent a variety of situations in order to attain results for a broad range of problems. Furthermore, a set of problems with relatively higher density parameter values, as well as a set of low-density problems were used to determine the effect of density on the relative performances of these method.

Particularly effective variants of algorithms using each of these penalty functions are designed by testing alternative algorithmic strategies at certain key stages of the various procedures. The motivation is to study how these methods perform in comparison with each other, and to provide insights into their viability for solving linear programming problems.

A sophisticated simplex implementation, CPLEX 6.0, is also used to provide a reference base for making comparative evaluations. Our implementation is rudimentary in comparison with the commercial software, and CPLEX is used only to benchmark the results. The results indicate that a particular variant (ALAG2) of the augmented Lagrangian approach is best among the tested methods for generally structured inequality and equality constrained problems, while the active set l_2 penalty method ASL2 might be better suited for specially structured instances. Moreover, the l_2 penalty approaches IEL2 and ASL2 yielded a reasonably good second-best performance in comparison with ALAG2 for problems having equality constraints alone.

Our experimentation with the density parameter exposed that for linear programs with a high density parameter, ASL2 is the best alternative among the tested algorithms; whereas, for low-density problems ALAG2 is the fastest method. Moreover, in comparison with CPLEX, all of the tested methods attained a final solution faster than CPLEX for the set of large-scale low-density problems. Although our implementations of the methods are primitive compared to CPLEX, ALAG2 was sometimes as fast as requiring only 16-23% of the CPU effort consumed by CPLEX in achieving a final solution with an acceptable margin of infeasibility.

Average rank tests based on the computational results obtained are performed using two different statistics, namely speed of convergence and quality of solution, in order to determine the relative effectiveness of the algorithms.

All these methods are suitable for obtaining near optimal solutions in the various stated contexts, but they need a more sophisticated implementation and refined algorithmic strategies in order to enhance robustness and derive solutions that are more accurate. However, the use of augmented Lagrangian method to solve large-scale low-density linear programs is very promising and should be explored more extensively.

1.2 Overview of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 introduces the research that has been done prior to our work on interior point algorithms and exterior penalty function approaches. Chapter 3 presents the problem description and notation along with a detailed description of the studied algorithms and their variants. Chapter 4 describes the problem generation procedure and discusses the computational experience obtained using a set of randomly generated problems as well as a set of standard NETLIB test problems arising in various applications. Furthermore, the results of a special experiment on the effect of sparsity of constraint coefficient matrix are included in Chapter 4. In addition, Chapter 4 contains a statistical analysis of the results along with the details of the test runs. Finally, Chapter 5 provides a summary and conclusions, insights into the algorithmic performance, and suggests avenues for further research in this relatively less explored domain of approaches for solving linear programming problems.

Chapter 2

Literature Review

Interior and exterior penalty-based strategies solve mathematical programming problems through a sequence of unconstrained or simply-constrained optimization problems over projected subspaces using different values of a penalty parameter. These methods were long dormant, being regarded as unfashionable and numerically unstable, principally because of the inevitable ill-conditioning that occurs as the penalty parameter tends to the prescribed limit.

Interior penalty function methods, also known as barrier function methods, set a barrier against leaving the feasible region. If the optimal solution is at the boundary of the feasible region, which is always the case in linear programs, the procedure moves from the interior to the boundary. These methods require the region $\{x : g(x) < 0\}$ to be bounded and nonempty; hence, equality constraints must either be dealt with separately or converted to inequality constraints, which increases the total number of constraints. There are other computational difficulties associated with barrier function methods. First, the procedure must initialize with a feasible solution. Consequently, this adds an additional effort to find an initial solution. The computational ill-conditioning effect due to the penalty parameter, as mentioned earlier, is also problematic. Also, as the boundary of the feasible region is approached, the chosen search direction and step size may produce a solution outside the feasible region due to round-off errors. This may lead to a misleading perception of a decrease in the objective function value of a minimization problem, even though the solution is actually infeasible.

The exponential worst-case behavior of the simplex algorithm gave rise to the question of whether a polynomial-time algorithm for solving linear programming problems could be constructed. This question was answered affirmatively by Khachian [16] who proposed a polynomial-time ellipsoid algorithm for determining a solution to linear programming problems with integer data, if a solution exists. This algorithm defines a ball centered at the origin and with a large enough radius, so that the ball contains a large portion of the feasible region. Unless the center of the ball is in the feasible region, the algorithm constructs a sequence of ellipsoids monotonically decreasing in volume, until the former check is realized. Although this new approach caused an initial excitement and wide publicity, its computational performance was disappointing. The computational effort was very much dependent on the worst-case behavior, which in this case was governed by a high-order polynomial. On the other hand, despite the fact that the long-established simplex method is of exponential complexity in the worst case, in practice, it tends to require a number of iterations that is approximately linear in problem dimension.

Consequently, the simplex method, having gone through considerable enhancements in implementation, did not have any serious competition until 1984, when Karmarkar [15] introduced a polynomial-time interior point method for linear programming problems. This algorithm consists of repeated projective transformations each followed by optimizing over an inscribed ball to create a sequence of points, which converges to the optimal solution in polynomial-time. A detailed description of this projective algorithm is given in Bazaraa et al.

[4]. The claims of dramatic computational efficiency of this algorithm resulted in an explosion of research effort on interior point methods for linear and convex programming (see [21] and [26] for a summary). Gill et al. [13] recognized Karmarkar's algorithm as a projected Newton variant of a logarithmic barrier function approach and provided a formal equivalence between the direction associated with Karmarkar's algorithm and the Newton search direction on a suitable barrier function.

The vast majority of theoretical results on interior point methods for linear programming assume that the starting point satisfies exactly the equality constraints and is interior to the inequality constraints. Lustig, Marsten and Shanno [17] have shown that by using a starting point in the interior of the region defined by the inequality constraints alone, it is possible to attain acceptable performance. They also build upon Megiddo's [18] primal-dual interior point algorithm for linear programming problems, by incorporating free variables and handling simple bounds in an easy way.

In addition, Cominetti and Dussault [9] present an exponential-penalty algorithm that does not require an interior starting point by using the idea of sequentially identifying the set of active constraints at an optimum. They indicate that this identification is best performed by interior methods because exterior methods ignore the effect of constraints that are not active at the current iteration, although they may be active at an optimal solution. They also point out that the rate of change of the penalty parameter needs to be only superlinear (4/3) for interior point methods, whereas it may be as high as quadratic for exterior penalty based approaches. Furthermore, when a constraint is violated, the exponential penalty function may produce an overflow when using the usual floating-point representation for real numbers.

On the other hand, exterior point methods for solving mathematical programs are designed to construct optimal solutions as limiting points of trajectories generated external to the feasible region. As such, these methods can be started from an arbitrary solution. In contrast with the interior point methods, there has not been much research done with exterior point methods for solving linear programs.

The basic idea in exterior penalty function methods is to transform a constrained problem into a single or a sequence of unconstrained problems by incorporating the constraints in the objective function to penalize any infeasibility. Associated with these methods is a penalty parameter μ that determines the intensity of the penalty, and as a result, the degree to which the unconstrained problem resembles the original constrained problem. As this parameter takes high values, the approximations become more accurate. To illustrate penalty functions, consider the following problem, where for notational simplicity, we have written just a single inequality and a single equality constraint (multiple constraints are treated similarly).

$$\begin{aligned} \text{Minimize} \quad & f(x) \\ \text{subject to} \quad & g(x) \le 0 \\ & h(x) = 0 \\ & x \in \Omega. \end{aligned}$$

Assume that this problem is transformed to the following unconstrained problem.

$$\text{Minimize} \qquad f(x) + \mu\left(|h(x)|^p + \max^p\{0, g(x)\}\right)$$

$$\text{subject to} \qquad x \in \Omega$$

where p is a positive integer. Clearly, an optimum to the above problem must be near feasible, since otherwise, a large penalty will be imposed, given a large value of μ.

These types of penalty function approaches have been known for a long time, but because of the ill-conditioning produced by using a very large penalty parameter, they have been regarded as outdated or numerically unstable, until more recent stable second-order methods have revived interest in these methods (see Nash and Sofer [20]). When μ is large, attaining feasibility dominates achieving low values of $f(x)$, and most of the available solution procedures will quickly move toward a (near) feasible point. This may result in a premature termination although this point may be far from optimality. These types of computational difficulties motivated the development of other numerical techniques such as exact penalty functions and Lagrange multiplier methods, as introduced in Chapter 1. The l_1 type of penalty function or absolute value penalty function corresponds to the above transformed problem with $p = 1$. However, since the value for the penalty parameter that achieves optimality for the original problem is usually not known, these methods need to examine a sequence of problems with increasing μ values. If μ is too large, ill-conditioning still occurs; and if μ is too small, the transformed problem is a poor representation of the original problem. Moreover, since the l_1 penalty function is nondifferentiable, even for differentiable f, g and h, the available efficient procedures for differentiable problems are inapplicable.

There have been several recent papers on exterior point methods to solve specialized cases of mathematical programming problems. Paparrizos [22] solves assignment problems by introducing an exterior point simplex algorithm of order $O(n^3)$ that initializes with a dual feasible, specially structured basis. He extends this algorithm in [23] to general linear programming problems. The basic idea in these algorithms is that for any two feasible solutions to a linear program, of which one may be an optimal solution, there exists a simplex path (possibly passing through the infeasible region) that connects them. In order to identify such a path, a new pivoting rule is used. Nevertheless, this method again possesses the exponential worst-case behavior. Al-Sultan and Murty [1] develop an exterior penalty algorithm for the nearest point problem, which is that of finding the nearest point (by Euclidean distance) in a convex polyhedral cone to a given point. This is a special convex quadratic program. As shown by Sherali et al. [28], Al-Sultan and Murty's procedure can be converted into a procedure for solving linear programs by sequentially adjusting a lower bound on the objective function. Andrus [2] considers problems having only inequality constraints and proves the convergence of an exterior point method that involves only one constraint in each iteration; hence, every iteration is computationally very simple. Andrus and Schaferkotter [3] build upon this procedure by devising a more efficient algorithm that incorporates equality constraints as well as inequality constraints. Each iteration of this algorithm involves one step within some constraining hyperplane, followed by a series of projections to force the new iterate to fall inside an envelope of the feasible solutions. They claim that this method is considerably faster than a standard

simplex method when the problem size is large. Sherali, Choi and Sen [28] develop an algorithm that constructs a polytope using the objective function and constraints for which the origin is an exterior point. This polytope is then translated, rotated and deformed based on a closest point routine, in order to construct an exterior trajectory of solutions leading to an optimum for the given linear programming problem. Furthermore, the algorithm is improved by generating dual solutions, which are used in the deformation of the polytope, in order for it to be positioned to facilitate a more effective procedure.

Of exterior penalty approaches in nonlinear programming, augmented Lagrangian methods (also known as the multiplier penalty function methods) have gained popularity since they avoid the ill-conditioning difficulties encountered by the classical approach which result as the penalty parameter approaches infinity (see Bazaraa et al. [5], Fletcher [12], and Bertsekas [7, 8]). Furthermore, they enjoy the property of being differentiable. Augmented Lagrangian methods, proposed first by Hestenes [14] and Powell [24], use both a Lagrange multiplier term and a quadratic penalty term in the unconstrained problem of the inner loop. The multiplier estimates are updated in an outer loop. Bertsekas [7] shows that the multiplier method converges at least linearly, and at a rate that is far more favorable than that of the quadratic penalty method.

Sen and Sherali [27] and Sherali and Ulular [30] propose a primal-dual conjugate subgradient algorithm for solving decomposable problems by coordinating an augmented Lagrangian penalty function with a Lagrangian dual function. Conn et al. [10] extend their previous approach, designed for simple bound constrained problems, to include general linear equality constraints. In this work, only the nonlinear constraints (all of which are equality constraints) are included in the Lagrangian function, and the general linear constraints are handled directly in the inner loop where the minimization of the augmented Lagrangian function is performed. Later, in [11], they also include general nonlinear inequality constraints within the framework of [10] with simple bounds being handled at the level of augmented Lagrangian minimization. Ben-Tal and Zibulevski [6] study a class of penalty/barrier multiplier methods, which are based on non-quadratic augmented Lagrangians for which the penalty parameters are functions of multipliers.

As mentioned earlier, little research has been conducted on using exterior point methods to solve linear programs. The primal-dual conjugate subgradient algorithm and the polytope sliding and deformation algorithm presented in [30] and [28], respectively, are among the few implementations of exterior point methods for solving linear programming problems. Large, sparse and specially-structured problem instances arise in the context of discrete and mixed-integer programming problems for which linear programming-based, tight lower bounds are of utmost importance. Sherali and Choi [30] develop a primal-dual conjugate gradient algorithm for solving specially structured or decomposable linear programming problems that might arise in such applications. Their results indicate that this algorithm is successful in reducing the duality gap in contrast with prior results on subgradient methods. Since the proposed primal-dual conjugate gradient algorithm exploits the network structure readily, this algorithm can also be used to solve network flow problems in the presence of side constraints.

The polytope sliding and deformation algorithm developed by Sherali et al. in [28] is based on a new interpretation of Karmarkar's algorithm. The polytope constructed by using the objective function and constraints, with the origin as an exterior point, is translated, rotated, and deformed, to build an exterior trajectory of solutions leading to an optimum. Computational experience

shows that problem size or degeneracy affect the solution time of this algorithm less than the solution time of the simplex method. Furthermore, a near feasible solution can be produced quickly, encouraging a switch to the simplex method or an interior point method using the available solution as an advanced start.

The incentive for this thesis is to study how several approaches of exterior penalty function methods, suitably modified and fine-tuned, perform in comparison with each other, and to provide insights into their viability for solving linear programming problems. In Chapters 3 and 4, the adopted algorithmic strategies of the exterior penalty function methods under investigation will be given. In addition, their performance in comparison with each other will be presented.

Chapter 3

Problem Definition and Algorithms

3.1 Problem Definition

Consider a linear programming problem stated in the following general form.

$$\begin{aligned} &\text{Minimize} && cx \\ &\text{Subject to} && A_i x \geq b_i \quad \text{for } i \in I_1 \\ & && A_i x = b_i \quad \text{for } i \in I_2 \\ & && x \in \Omega \equiv \{x : 0 \leq x \leq u\}, \end{aligned}$$

where $x \in R^n$, c and A_i for $i \in I \equiv I_1 \cup I_2$ are row vectors in R^n with $|I_1| \equiv m_1$, $|I_2| \equiv m_2$, and where $|I| \equiv m = m_1 + m_2$. The constraints $A_i x \geq b_i$ for $i \in I_1$ will be referred to as $A^1 x \geq b^1$ in matrix form, and likewise, $A_i x = b_i$ for $i \in I_2$ as $A^2 x = b^2$. For computational effectiveness, it is assumed that the variables have been scaled so that $u_j = 10$, say, for each $j = 1, \ldots, n$, and that the rows have been scaled so that $\|A_i\| = 1$ for all $i \in I$.

Now, the aforementioned three variants of the exterior point approaches studied in this thesis will be presented. The focus is on composing simple, effective algorithmic strategies in designing these procedures. As far as their theoretical convergence is concerned, it should be pointed out that this can be readily achieved in practice by interspersing a spacer step (see Bazaraa et al. [5]) every finite number of iterations, for example, by adopting an exact line search along the (projected) antigradient direction.

3.2 Active Set l_2 Penalty Approach (ASL2)

Consider the quadratic penalty function

$$\phi_\mu(x) = cx + \frac{1}{2}\mu \sum_{i \in I_a} (b_i - A_i x)^2 \tag{3.1}$$

where I_a is the set of indices for the declared active set of constraints and satisfies $I_2 \subseteq I_a \subseteq I$, where μ is a penalty parameter. Note that I_a includes all of the equality constraints and a set of inequality constraints (possibly null) from I_1 that are predicted to be active at an optimal solution. Fletcher [12], Bazaraa et al. [5], Bertsekas [8], and Nash and Sofer [20] discuss strategies for managing the active set of constraints I_a and the penalty parameter μ in order to recover an optimum as $\mu \to \infty$.

In this implementation, for a given μ and I_a, the bound constrained problem (3.1) is minimized in an inner loop using the gradient projection method, with the conjugate gradient method along with exact line searches being applied while residing over subspaces of the constraining hyperrectangle Ω. The management of the active set I_a along with the penalty parameter μ is performed according to the prescribed algorithm described below. For an alternative, more detailed approach that composes loops of the gradient projection method and the conjugate gradient method, see More and Toraldo [19].

<u>Algorithm 1:</u> Active Set l_2 Penalty Approach (ASL2)

Initialization: Let $I_a \equiv I_2$, set $\mu = 10$, $\bar{x} = (0,...,0) \in R^n$ and $k_{max} = \max\{n/20, 50\}$. (The latter is a limit on the maximum number of iterations performed within an inner loop, and is progressively increased from one outer loop to the next as the algorithm proceeds.)

Inner Loop: Penalty Function Minimization

Step 0: Let $k = 1$, $x^1 = \bar{x}$. Compute $\phi_\mu(x^1)$.

Step 1: Evaluate the gradient

$$\nabla\phi_\mu(x^k) = c^t - \mu\sum_{i\in I_a}(b_i - A_i x^k)A_i^t \tag{3.2}$$

and let the (column) vector g^k be the projected antigradient of $\phi_\mu(x^k)$ having components g_j^k, $j = 1,\ldots,n$, given by

$$g_j^k = \begin{cases} 0 \quad \text{if } x_j^k = 0 \text{ and} - \nabla_j\phi_\mu(x^k) < 0, \text{ or if } x_j^k = u_j \text{ and} - \nabla_j\phi_\mu(x^k) > 0 \\ -\nabla_j\phi_\mu(x^k) \quad \text{otherwise}, \forall j = 1,...,n. \end{cases} \tag{3.3}$$

If the following holds true,

$$\left\|g^k\right\| \le 0.001 \tag{3.4}$$

go to Step 4; else, continue with Step 2.

Step 2: Compute the search direction, d^k, given by the projected antigradient if RESET = 1 (i.e., the procedure has moved to a new face of Ω), or by the conjugate gradient method if RESET = 0:

$$d^k = \begin{cases} g^k & \text{if RESET} = 1 \\ g^k + \dfrac{\left\| g^k \right\|^2}{\left\| g^{k-1} \right\|^2} d^{k-1} & \text{if RESET} = 0. \end{cases} \tag{3.5}$$

If RESET = 0 and if the following Beale-Powell reset criteria hold true (see Bazaraa et al. [5], for example)

$$\max\left\{ \left| g^k \cdot g^{k-1} \right|, \left| \left\| g^k \right\|^2 - g^k \cdot d^k \right| \right\} > 0.2 \left\| g^k \right\|^2, \tag{3.6}$$

then put $d^k = g^k$.

Find the maximum permissible step-length over the box $0 \le x \le u$ as

$$\lambda_{\max} = \max\left\{ \lambda : 0 \le x^k + \lambda d^k \le u \right\}. \tag{3.7}$$

Furthermore, in order to find the unconstrained minimum $\hat{\lambda}$ of the convex quadratic line search problem, compute

$$\theta_1 = \phi_\mu(x^k),\ \theta_2 = \phi_\mu(x^k + \lambda_{\max} d^k),\ \alpha = \nabla \phi_\mu(x^k) \cdot d^k,\ \beta = \frac{\theta_2 - \theta_1 - \alpha \lambda_{\max}}{\lambda_{\max}^2}. \tag{3.8}$$

The unconstrained step-size is then given by

$$\hat{\lambda} = \begin{cases} -\dfrac{\alpha}{2\beta} & \text{if } \beta \ne 0 \\ \infty & \text{otherwise.} \end{cases} \tag{3.9}$$

The constrained step-size is therefore obtained as

$$\lambda_k = \text{minimum}\left\{ \hat{\lambda}, \lambda_{\max} \right\}. \tag{3.10}$$

If $\lambda_k = \lambda_{\max}$ in (3.9), then let RESET = 1; else, if $\lambda_k < \lambda_{\max}$, then let RESET = 0.

Step 3: Compute $x^{k+1} = x^k + \lambda_k d^k$, and increment k by 1. If $k > k_{\max}$, proceed to Step 4; else, return to Step 1.

Outer Loop Penalty Parameter and Active Set Management

Step 4: Put $\bar{x} = x^k$. Based on the Lagrangian multiplier estimates (see Equation (3.2)) and constraint violations, perform the following updates.

Delete constraint $i \in I_1$ from the active set, I_a, if constraint i satisfies

$$\mu\left(b_i - A_i \bar{x}\right) \leq -0.01 \quad \text{for any} \quad i \in I_1 \cap I_a . \tag{3.11}$$

Add constraint i to the active set if constraint i satisfies

$$\left(b_i - A_i \bar{x}\right) \geq 0.01 \quad \text{for any} \quad i \in I_1 - I_a . \tag{3.12}$$

If I_a has not changed and $\left|b_i - A_i \bar{x}\right| \leq 0.01 \ \forall i \in I_a$, stop with $\bar{x}$ as a near optimum.

If $\mu = 10^6$, stop with $\bar{x}$ as the prescribed solution. Else, replace $k_{max} \leftarrow 1.1k_{max}$ and $\mu \leftarrow 10\mu$ and return to Step 0.

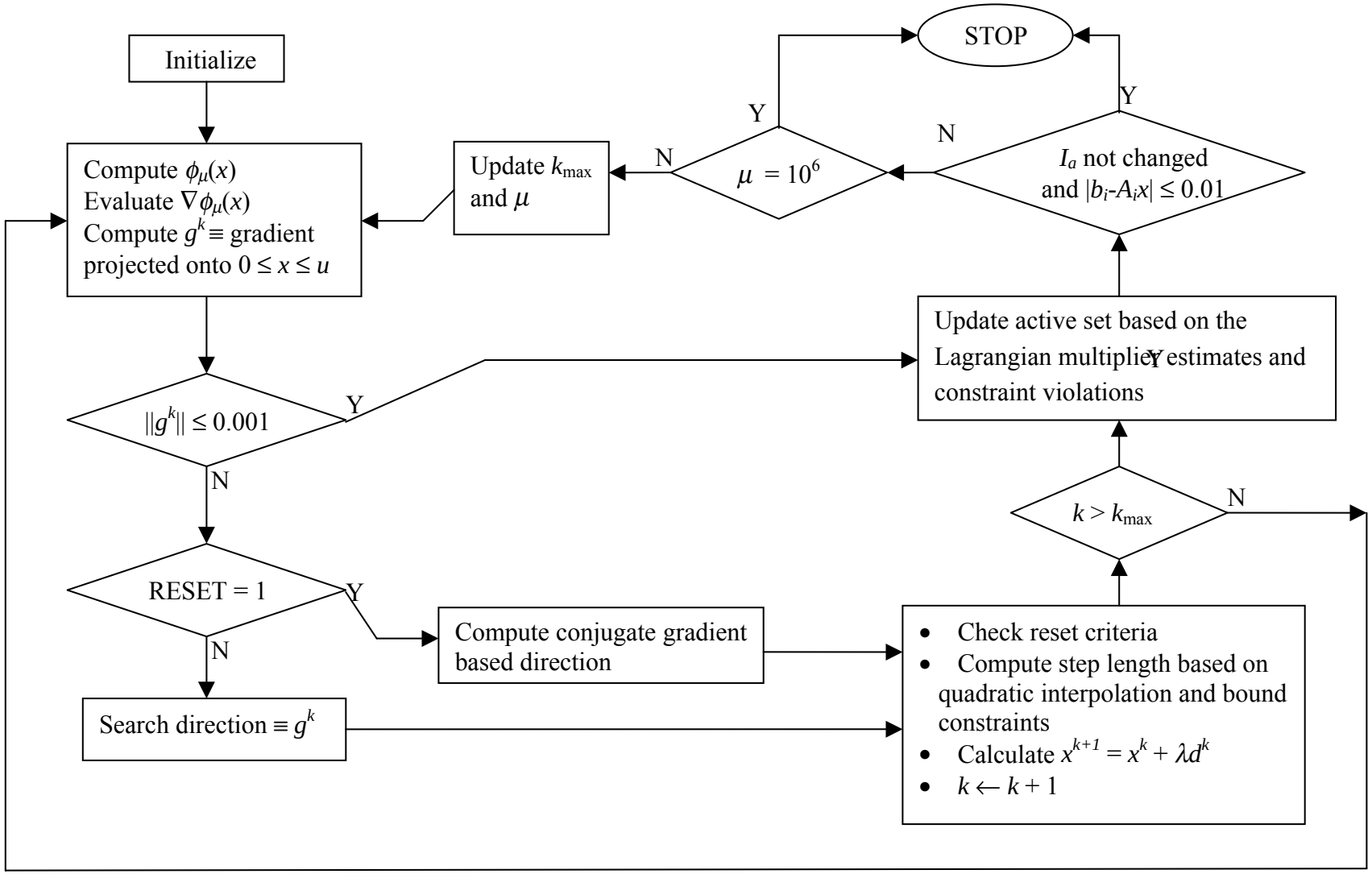

Figure 3.1: Flowchart of Active Set l_2 Penalty Approach

3.3 Inequality-Equality Based l_2 Penalty Approach (IEL2)

It is of interest to study how the active set penalty approach compares with a direct inequality-equality penalty function method. Hence, in this section an alternative algorithm based on the use of the standard quadratic penalty function is investigated (see Bazaraa et al. [5]),

$$\phi_\mu(x) = cx + \frac{1}{2}\mu\left[\sum_{i\in I_1}\max^2\{0, b_i - A_i x\} + \sum_{i\in I_2}(b_i - A_i x)^2\right] \tag{3.13}$$

instead of (3.1). The penalty function is minimized in an inner loop using the same basic strategy as for Algorithm ASL2, except that since now only an inexact line search is performed due to the more complex piecewise quadratic nature of the penalty function (3.13), Sherali and Ulular's [31] conjugate gradient scheme is adopted for generating the search direction while the iterates reside over some face of Ω (the reset parameter RESET equals zero). The algorithm proceeds as follows.

<u>Algorithm 2:</u> Inequality-Equality Based l_2 Penalty Approach (IEL2)

Initialization: Let k = 1, x^1 = $(0,\ldots,0)^t$, RESET = 1, μ = 10 and set k_{max} = $\max\{n/20, 50\}$. Compute $\phi_\mu(x^1)$.

Step 1: Evaluate the gradient

$$\nabla\phi_\mu(x^k) = c^t - \mu\left[\sum_{i\in I_1}\max\{0, b_i - A_i x^k\}A_i^t + \sum_{i\in I_2}(b_i - A_i x^k)A_i^t\right] \tag{3.14}$$

and let the projected antigradient g^k be computed as in (3.3) based on (3.14).

If (3.4) holds true, and if $b_i - A_i x^k \le 0.01\ \forall i \in I_1$ and $\left|b_i - A_i x^k\right| \le 0.01\ \forall i \in I_2$, then stop with x^k as a near optimum; otherwise, go to Step 4.

Step 2: Compute the search direction as

$$d^k = \begin{cases} g^k & \text{if RESET} = 1 \\ g^k + \left[\dfrac{(1/\lambda_{k-1})g^k \cdot p^k - g^k \cdot q^k}{d^{k-1} \cdot q^k}\right] d^{k-1} & \text{if RESET} = 0, \end{cases} \tag{3.15}$$

where

$$p^k \equiv x^k - x^{k-1} \tag{3.16}$$

$$q^k \equiv g^{k-1} - g^k . \tag{3.17}$$

If RESET = 0 and (3.5) holds true, then let $d^k = g^k$.

Compute $\lambda_{\max}$ and $\hat{\lambda}$ using (3.7) - (3.9), except that in (3.8), replace "$\lambda_{\max}$" by $\lambda' \equiv \min\{1, \lambda_{\max}\}$. Note that in (3.8), we have $\alpha < 0$ since d^k is a descent direction and $\beta \geq 0$ by the convexity of ϕ. Hence, compute the step-size λ_k using (3.10).

If $\lambda_k = \lambda_{\max}$ in (3.10), then let RESET = 1; else, if $\lambda_k < \lambda_{\max}$, then let RESET = 0.

Step 3: Compute $x^{k+1} = x^k + \lambda_k d^k$, and increment k by 1. If $k > k_{\max}$, proceed to Step 4; else, return to Step 1.

Step 4: If $\mu = 10^6$, stop with x^k as the prescribed solution. Else, replace $k_{max} \leftarrow 1.1k_{max}$, $\mu \leftarrow 10\mu$, put $x^1 = x^k$, reset $k = 1$, compute $\nabla\phi_\mu(x^1)$, put RESET = 1, and return to Step 1.

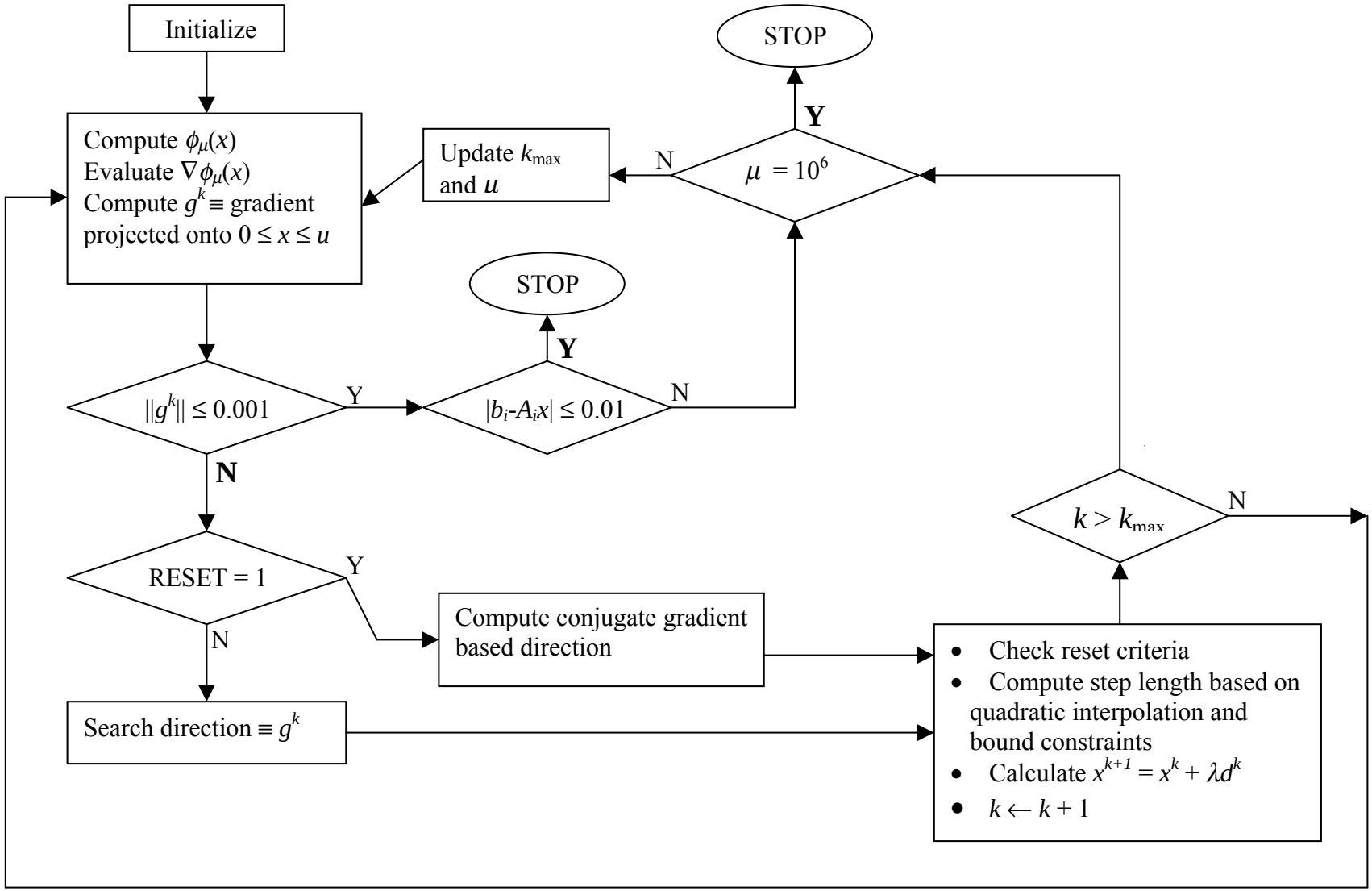

Figure 3.2: Flowchart of Inequality-Equality Based l_2 Penalty Approach

3.4 Augmented Lagrangian Approach (ALAG)

In the ASL2 as well as the IEL2 algorithms, the penalty parameter μ has to be taken to infinity in a limiting sense to recover an optimal solution. This causes the inner loop problems to become progressively ill-conditioned. On the other hand, the augmented Lagrangian penalty function approach recovers an exact optimal solution for finite penalty parameter values, and unlike l_1 penalty methods that share this same property, it also enjoys the property of being differentiable. The ALAG approach produces both primal and dual solutions by using the augmented Lagrangian penalty function by way of combining the algorithmic properties of Lagrangian duality methods and penalty function methods (see Bazaraa et al. [5], Bertsekas [8], and Conn et al. [10, 11] for an exposition on this approach for solving nonlinear programs). It should be pointed out here that more efficient implementations of ALAG methods for general twice-differentiable nonlinear programs such as in Conn et al. [10] compose the augmented Lagrangian penalty function based on the nonlinear constraints alone, and handle the linear constraints explicitly within the inner loop subproblems. Since purely linear programs are solved in this research effort, our interest is to study the performance of such exterior point methods when the general linear constraints are accommodated into the augmented Lagrangian penalty function and only the simple bound constraints are handled implicitly within the inner loop subproblems.

The ALAG approach augments the IEL2 method by using a Lagrangian function in order to avoid taking the penalty parameter to infinity for recovering an optimal solution. Similar to the previous methods, the penalized objective function will be minimized in an inner loop, and the penalty parameters, as well as now, update the Lagrange multiplier estimates will be modified in an outer loop.

The ALAG penalty function is given by

$$\phi_\mu(x,\pi) = cx + \frac{1}{2}\sum_{i\in I_1} \mu_i \max^2\left\{0, \frac{\pi_i}{\mu_i} + b_i - A_i x\right\} + \sum_{i\in I_2}\left[\pi_i(b_i - A_i x) + \frac{1}{2}\mu_i (b_i - A_i x)^2\right] \tag{3.18}$$

where $\pi_i,\ i \in I$, are Lagrange multiplier estimates and $\mu_i,\ i \in I$, are penalty parameters which are tuned separately for each constraint. As pointed out by Conn et al. [10], among others, the manipulation of a separate penalty parameter for each constraint based on its degree of violation turns out to be an effective computational strategy in the context of ALAG approaches.

In order to measure the extent of constraint violation for a given x for the purpose of adjusting penalty parameters as well as designating a suitable stopping criterion, it is needed to define a constraint violation or infeasibility function. Three such possible functions $VIOL(\cdot)$ will be explored as given below, each leading to a respective version of the augmented Lagrangian approach, denoted ALAG1, ALAG2, and ALAG3.

$$VIOL(x) = \max\left\{\max\{0, b_i - A_i x\} \text{ for } i \in I_1, |b_i - A_i x| \text{ for } i \in I_2\right\}, \tag{3.19}$$

$$VIOL(x) = \sqrt{\sum_{i \in I_1} \max^2\{0, b_i - A_i x\} + \sum_{i \in I_2} (b_i - A_i x)^2} \Big/ m_v \tag{3.20}$$

where m_v is the number of constraints violated, and

$$VIOL_i(x) = \begin{cases} \max\{0, b_i - A_i x\} & \text{for } i \in I_1 \\ |b_i - A_i x| & \text{for } i \in I_2. \end{cases} \tag{3.21}$$

Note that in the third version, the infeasibility function (3.21) is defined separately for each constraint, in contrast with the aggregate definitions (3.19) and (3.20). The overall algorithm implemented can now be described as follows.

<u>Algorithm 3:</u> Augmented Lagrangian Approach (ALAG)

Initialization: Let $\bar{\pi} = (0,...,0)$. (Alternatively, some deflected subgradient method applied to a Lagrangian dual formulation that dualizes all the structural constraints ($i \in I$) could be used to find a starting dual solution $\bar{\pi}$. Also, note that while $\bar{\pi}$ is not sufficiently close to an optimal set of multipliers, an accurate minimization of $\phi_\mu(x, \pi)$ with respect to x is not very meaningful. Hence, the termination criterion for the inner loop penalty function minimization iterations is progressively tightened as the algorithm proceeds.)

Let $\bar{x} = (0,\ldots,0)^t \in R^n$, $\mu_i = 10 \ \forall i \in I$, and set $k_{max} = \max\{n/20, 50\}$.

Compute $VIOL(\bar{x})$ using (3.19) or (3.20), or $VIOL_i(\bar{x}) \ \forall i \in I$ using (3.21), according to the respective method ALAG1, ALAG2, or ALAG3 being implemented.

Set the counter of outer loop iterations to $\tau = 0$, and the counter of consecutive inner loop visits without performing Lagrange multiplier updates to $K = 0$.

Inner Loop: Penalty Function Minimization

Step 0: Let $k = 1$, $x^1 = \bar{x}$, RESET = 1. Compute $\phi_\mu(x^1, \bar{\pi})$.

Step 1: Evaluate the gradient

$$\nabla \phi_\mu(x^k, \bar{\pi}) = c^t - \sum_{i \in I_1} \mu_i \max\left\{0, \frac{\bar{\pi}_i}{\mu_i} + b_i - A_i x^k\right\} A_i^t \ - \sum_{i \in I_2} \left[\bar{\pi}_i + \mu_i (b_i - A_i x^k)\right] A_i^t \tag{3.22}$$

and let g^k, the projected antigradient of $\phi_\mu(x^k, \bar{\pi})$, be computed as in (3.3) based on (3.22).

If the following condition holds, go to Step 4:

$$\left\| g^k \right\| \le 10^{-(1+\lfloor 0.4\tau \rfloor)}. \tag{3.23}$$

Step 2: Compute the direction of search as in (3.15), (3.16), and (3.17). If RESET = 0 and (3.6) holds true, then put $d^k = g^k$. Compute $\lambda_{\max}$ and $\hat{\lambda}$ using (3.7) - (3.9), except that in (3.8), replace "$\lambda_{\max}$" by $\lambda' \equiv \min\{1, \lambda_{\max}\}$. Hence, compute the step-size λ_k using (3.10).

If $\lambda_k = \lambda_{\max}$ in (3.9), then let RESET = 1; else if $\lambda_k < \lambda_{\max}$, then let RESET = 0.

Step 3: Compute $x^{k+1} = x^k + \lambda_k d^k$, and increment k by 1. If $k > k_{\max}$, go to Step 4; else, return to Step 1.

Step 4 (ALAG1 & ALAG2): Let $\bar{x}_{new} = x^k$. Compute $VIOL\left(\bar{x}_{new}\right)$ as in (3.19) or (3.20) respectively.

For ALAG1: If $VIOL\left(\bar{x}_{new}\right) \le 0.01$, stop with $\left(\bar{x}_{new}, \bar{\pi}\right)$ as a near optimal pair of primal-dual solutions, where $\bar{\pi}$ is updated as at Step 5.

For ALAG2: If $VIOL\left(\bar{x}_{new}\right) \le 0.00001$, stop with $\left(\bar{x}_{new}, \bar{\pi}\right)$ as a near optimal pair of primal-dual solutions, where $\bar{\pi}$ is updated as at Step 5. (A tighter tolerance was needed here to achieve a comparable degree of feasibility at termination because of the nature of (3.20) versus (3.19) and (3.21).)

Otherwise, if $VIOL\left(\bar{x}_{new}\right) \le 0.25\, VIOL\left(\bar{x}\right)$, go to Step 5;

else, if $VIOL\left(\bar{x}_{new}\right) > 0.25\, VIOL\left(\bar{x}\right)$, then for each $i \in I$ for which the corresponding term $VIOL_i\left(\bar{x}_{new}\right)$ in (3.21) exceeds

$\left[0.25 - 0.1\left(1 - \mathrm{e}^{-K}\right)\right] VIOL_i\left(\bar{x}\right)$, replace μ_i by $10\mu_i$.

If $K < 5$, then increment K by 1, replace $\bar{x}$ by $\bar{x}_{new}$, and return to Step 0;

else, reset $K = 0$, and proceed to Step 5.

Step 4 (ALAG3): Let $\bar{x}_{new} = x^k$. Compute $VIOL_i\left(\bar{x}_{new}\right)\ \forall i \in I$ as in (3.21).

If $\max_{i \in I}\left\{VIOL_i\left(\bar{x}_{new}\right)\right\} \le 0.01$, stop with $\left(\bar{x}_{new}, \bar{\pi}\right)$ as a near optimal pair of primal-dual solutions, where $\bar{\pi}$ is updated as at Step 5.

Otherwise, for each $i \in I$ such that $VIOL_i\left(\bar{x}_{new}\right) \le 0.25\, VIOL_i\left(\bar{x}\right)$ holds true, perform Step 5;

also, for each $i \in I$ such that $VIOL_i\left(\bar{x}_{new}\right) > 0.25\, VIOL_i\left(\bar{x}\right)$ holds true, replace μ_i by $10\mu_i$.

If Step 5 has been performed for any $i \in I$, then replace $\bar{x}$ by $\bar{x}_{new}$ and go to Step 6.

Else, replace $\bar{x}$ by $\bar{x}_{new}$, and return to Step 0.

Outer Loop: Lagrange Multiplier Update

Step 5 (ALAG1 & ALAG2): Replace $\bar{x}$ by $\bar{x}_{new}$, and examining the first-order optimality conditions based on (3.22) (see Bazaraa et al. [5]), update

$$\bar{\pi}_i \leftarrow \bar{\pi}_i + \mu_i \max\left\{-\frac{\bar{\pi}_i}{\mu_i}, b_i - A_i \bar{x}\right\} \quad \forall i \in I_1 \text{ and } \mu_i = \max\left\{\mu_i, 100\pi_i\right\} \quad \forall i \in I_1 \tag{3.24}$$

$$\bar{\pi}_i \leftarrow \bar{\pi}_i + \mu_i \left(b_i - A_i \bar{x}\right) \quad \forall i \in I_2 . \tag{3.25}$$

Proceed to Step 6.

Step 5 (ALAG3): (This is a subroutine called from Step 4 in the version ALAG3, and is motivated similarly to Step 5 for ALAG1 and ALAG2.)

If $i \in I_1$, then update $\bar{\pi}_i \leftarrow \bar{\pi}_i + \mu_i \max\left\{-\frac{\bar{\pi}_i}{\mu_i}, b_i - A_i \bar{x}_{new}\right\}$,

and $\forall i \in I_1$ and $\mu_i = \max\left\{\mu_i, 100\pi_i\right\} \ \forall i \in I_1$.

If $i \in I_2$, then update $\bar{\pi}_i \leftarrow \bar{\pi}_i + \mu_i \left(b_i - A_i \bar{x}_{new}\right)$.

Step 6: If $\tau = 12$, stop with $\left(\bar{x}, \bar{\pi}\right)$ as the prescribed pair of primal-dual solutions. Else, replace $k_{max} \leftarrow 1.1 k_{max}$, increment τ by 1 and return to Step 0.

The main difference between ALAG3 and the versions ALAG1 and ALAG2 is that the degree of constraint violation is checked individually for each constraint $i \in I$ in ALAG3, and an update of the Lagrangian multiplier is performed for constraint i whenever the condition $VIOL_i\left(\bar{x}_{new}\right) \le 0.25\, VIOL_i\left(\bar{x}\right)$ holds true, regardless of the feasibility status of the other constraints. If the

mentioned condition does not hold, then the penalty parameter associated with constraint i is updated. This strategy is similar to that prescribed by Conn et al. [10], and its motivation is to accelerate the convergence of the Lagrange multipliers for the constraints independently based on their individual progress toward attaining feasibility, in contrast to performing this update collectively based on an overall measure of feasibility. Note that for the versions ALAG1 and ALAG2, the implementation as prescribed in Bertsekas [8] and Bazaraa et al. [5] that corresponds to letting $K \equiv 0$ throughout was attempted first. However, the algorithmic performance was quite inadequate and a significant improvement was made by introducing the parameter K as in Step 4. Observe that this modification causes the penalty parameters to be increased more aggressively on consecutive visits to Step 4 without having updated the Lagrange multipliers, and when this counter K is greater than or equal to 5, the algorithm is forced to update the Lagrange multipliers. This prevents too many inner loop iterations from being performed while the Lagrange multipliers remain unchanged. The algorithms iterated considerably lesser after making this modification. No such modification was necessary for ALAG3, since it already more actively updates the Lagrange multiplier estimates at each visit to Step 4.

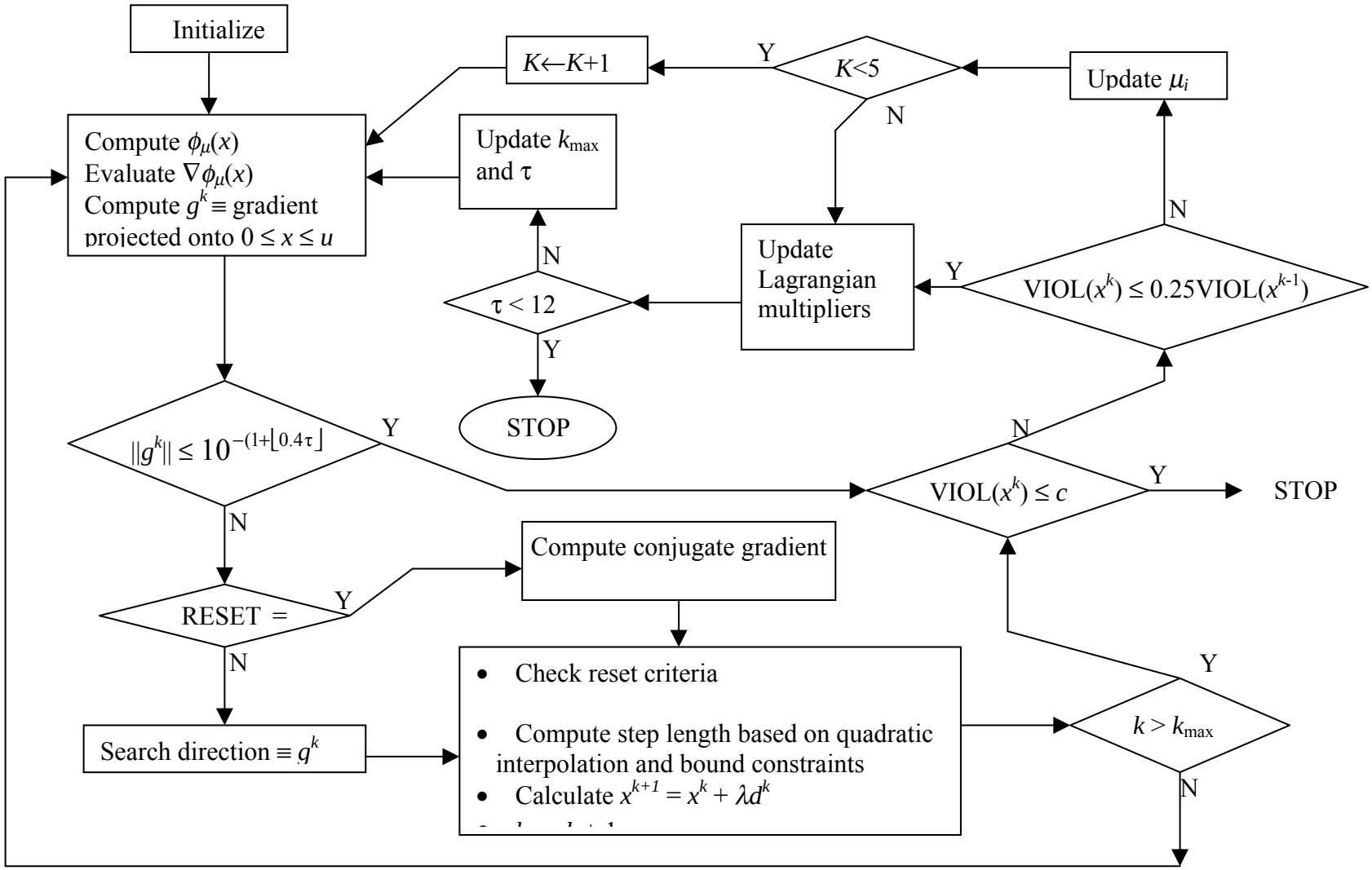

Figure 3.3: Flowchart of Augmented Lagrangian Approach (ALAG1)

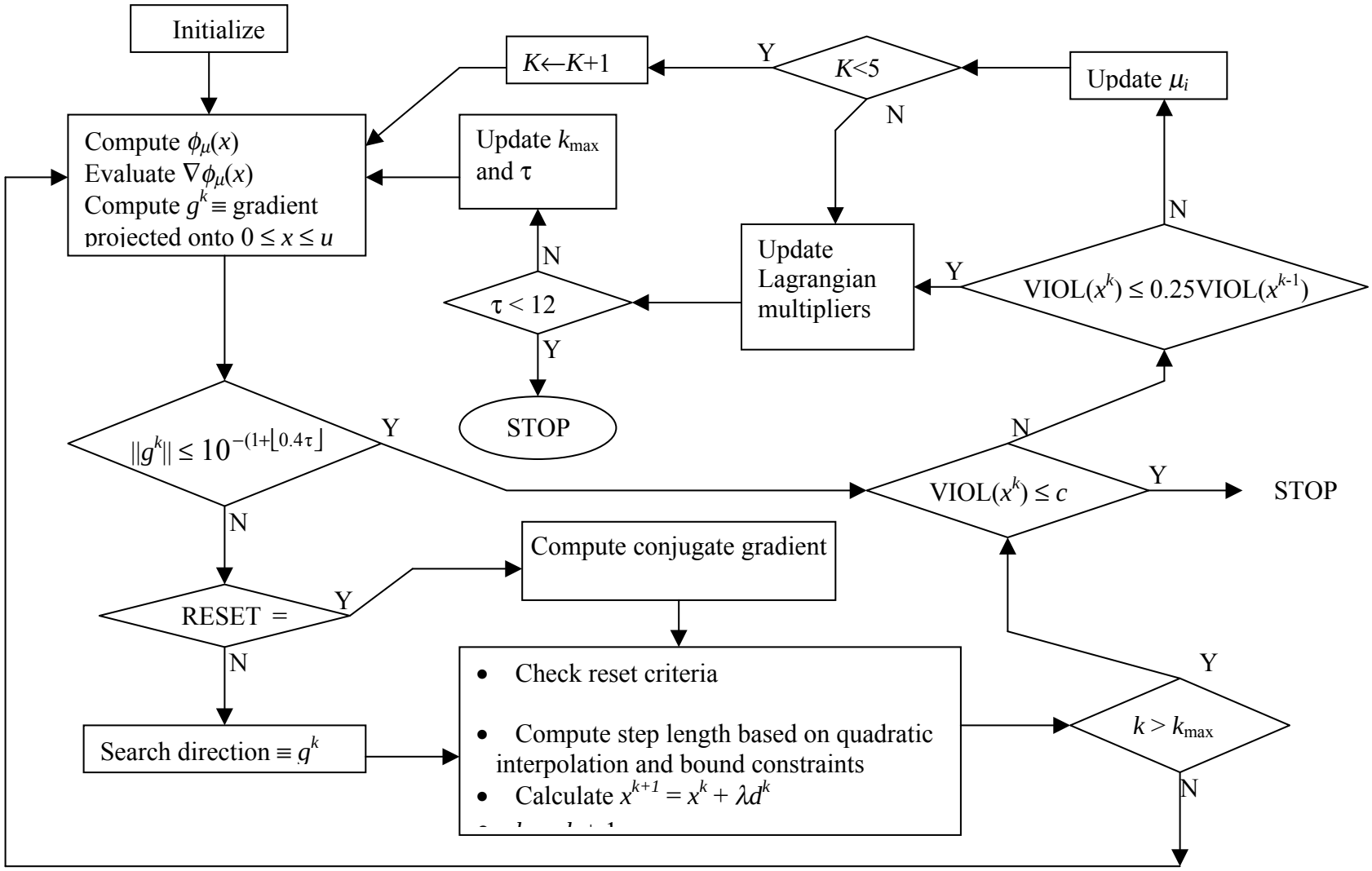

Figure 3.4: Flowchart of Augmented Lagrangian Approach (ALAG2)

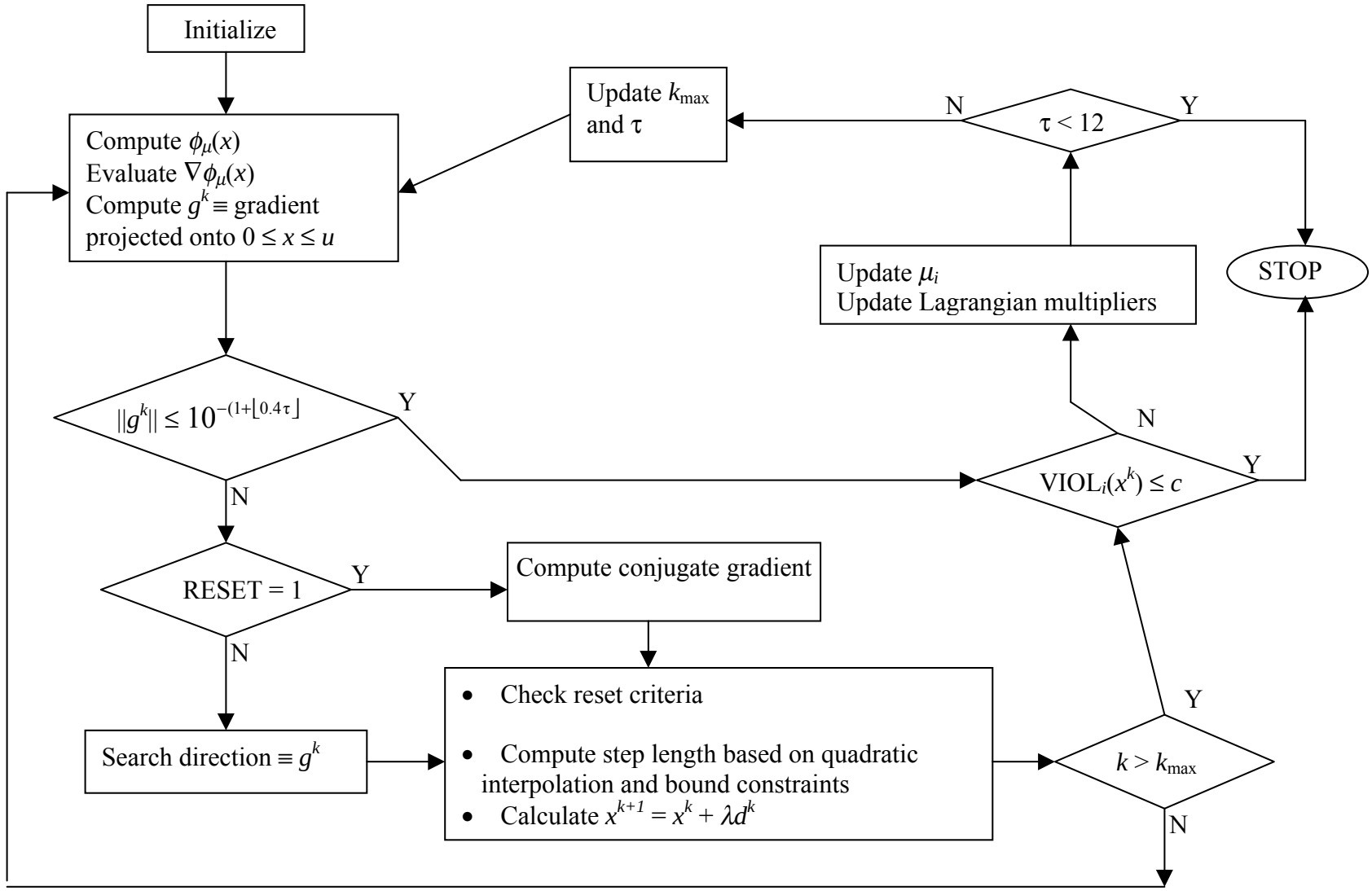

Figure 3.5: Flowchart of Augmented Lagrangian Approach (ALAG3)

Chapter 4

Computational Experience

4.1 Problem Generation Algorithm and NETLIB Problems

For computational purposes, first, a set of 30 random test problems was generated as in Rosen and Suzuki [25] and Sherali and Myers [29]. Components for the primal solution, $\bar{x}$, were generated uniformly on [0,1], with $m_2 + \lfloor m_1/2 \rfloor$ of these components being generated on (0,1) and the remainder being set to either 0 or 1. The upper bound u_j was taken as 1 for all j=1,…,n. Next, the dual solutions, α_i, associated with the inequality constraints $i \in I_1$ were generated, with $m_1 - \lfloor m_1/2 \rfloor$ of these being generated uniformly on (0,5), while the remaining components were set to 0. The dual solutions, β_i, associated with the equality constraints $i \in I_2$ were generated on [-10,10], one third of them being set to 0. The coefficient matrix A^1 for the inequality constraints, and A^2 for the equality constraints were taken to be d% dense having nonzero elements equal to either +1 or –1. The right-hand side b^1 for the inequality constraints was set to $A^1\bar{x}$ with a unit being subtracted from component i if the associated dual variable α_i was 0. The right-hand side b_2 for the equality constraints was set to $A^2\bar{x}$. Finally, the objective function coefficients, c, were generated as $\alpha A^1 + \beta A^2$, with one unit being added to component j if the corresponding value of $\bar{x}_j$ was zero.

Other than the randomly generated test problems, some 15 standard NETLIB test problems arising in various applications were also used to test the performance of the algorithms. The details of all 45 test problems are summarized in Table 4.1. Problems 1-15 are general linear programming problems in the form given in Section 3.1. Problems 16-30 are problems having only equality constraints. The original names of the remaining 15 standard NETLIB test problems are used to facilitate their identification. The results obtained for the algorithms presented in Chapter 3 are compared in the following discussion, and are displayed in Tables 4.2-4.7. The commercial simplex code CPLEX 6.0 is used to benchmark the results.

Tables 4.2, 4.3, and 4.4 respectively summarize the results obtained for the randomly generated problems in general form having both equality and inequality constraints, having only equality constraints, and for the NETLIB test problems. A special experiment was also conducted for the augmented Lagrangian methods to assess their effectiveness under the ideal condition of having a priori knowledge of the Lagrange multipliers. Since the ALAG penalty function is exact once the Lagrangian multipliers have been accurately estimated, the motivation here was to study the relative effectiveness of this ideal ALAG penalty function versus the more realistic l_2 and ALAG variants studied herein. Tables 4.5, 4.6, and 4.7 summarize the corresponding results obtained for the aforementioned three classes of test problems. The results for each problem using different algorithms are summarized one per column in Tables 4.2-4.7. Every problem is denoted by its index number or name from Table 4.1, given at the top of its corresponding column.

To provide a broader view of the relative performances of the algorithms, the average of each performance measure over the problems examined is also computed. Tables 4.8-4.10 present these indicators for the three respective classes of test problems, and Tables 4.11-4.13 present similar indicators for the experiment in which runs were made for the ALAG method with known Lagrange multipliers. Finally, Tables 4.14 and 4.15 exhibit the overall averages over the runs in Tables 4.8-4.10, and 4.11-4.13, respectively.

When comparing our results with that of CPLEX, it should be noted that a single iteration of the simplex method implementation in CPLEX is different from that of ASL2, IEL2, and ALAG. In the latter algorithms, one iteration corresponds to a gradient and functional evaluation, a line search step that involves two additional function evaluations, and a periodic parameter update step, all of complexity O(*mn*). In CPLEX, an iteration involves pricing, a pivoting operation, and a change of basis (roughly of complexity O(*mn*)). CPLEX also uses a preprocessing routine to find a reduced, advanced basis before starting the iterations. Hence, the iteration counts are only roughly comparable. The CPU times needed to compute the solutions are also reported and can be viewed in Tables 4.1-4.7 for each run.

The statistics reported for each run include the LP minimum value attained, the total number of iterations performed, the average constraint violation at termination, and the total execution time in seconds. The computer used to test the exterior point algorithms of Chapter 3 is a Pentium II-300 PC with 64 MB of RAM, operating on Windows95. The coding language used is FORTRAN Visual Workbench, v 1.00, and the compiler used is FORTRAN 32. CPLEX, version 6.0, was run on a Sun Ultra 1 Model 170 workstation with 256 MB of RAM running under Solaris 2.5.

4.2 Results for Randomly Generated Problems having Inequality and Equality Constraints

4.2.1 Active Set Penalty Approach (ASL2):

The performance of this method was comparable with that for several of the other algorithms, though not the best. Usually, the optimum LP solution was not reached. The average constraint violations attained by the prescribed solutions are of the order of 10^{-2} to 10^{-4}. The CPU time to reach the stopping criterion was lesser than that for IEL2 and ALAG1. (The solution time of CPLEX outperformed all of the algorithms.)

4.2.2 Inequality-Equality Based Penalty Approach (IEL2):

This algorithm was able to reach an optimal solution with an acceptable magnitude of constraint violations for all of the test problems. Among all the procedures, the smallest average constraint violations were obtained using this algorithm, ranging from the order of 10^{-5} to 10^{-6}. These are 10 to 10,000 times smaller than the average constraint violations achieved by the other algorithms. Furthermore, the total time and the number of iterations needed to find optimal solutions are comparable with those for the augmented Lagrangian approach and the ASL2 approach -- sometimes better, other times worse.

4.2.3 Augmented Lagrangian Approach (ALAG):

The first version (ALAG1) investigated for this approach performed comparably with ASL2 and IEL2. The performance of the second version (ALAG2) was the best among all the investigated methods in terms of the number of iterations and solution time. The third version (ALAG3), in which the constraints were checked individually and the penalty parameters and Lagrangian multipliers were updated at the end of each inner loop on an individual basis, performed better than the first version as well as the ASL2 and IEL2 methods. Examining the average constraint violations, it can be observed that all of the versions have violations ranging from the order of 10^{-3} to 10^{-5}. The number of iterations needed were usually more than those for CPLEX.

The outer loop of this algorithm updates the penalty parameters and the Lagrangian multiplier estimates. Theoretically, given the optimal Lagrange multipliers, ALAG will reach an exact optimal solution for a set of finite positive penalty parameter values. Hence, all three versions of this approach were run by fixing the Lagrange multipliers at their optimal values, which were predetermined in the generation of the problems, to ascertain the best expected empirical performance of the algorithms. Accordingly, only the penalty parameters were now updated in the outer loops.

Compared with the original algorithms (when the Lagrange multipliers are not known), there is a significant decrease in the number of iterations for all of the cases, ranging from 7.5-99% with respect to the original iteration counts. In general, as expected, the improvement for ALAG1 is more evident than that for ALAG2 or ALAG3. Comparing the number of iterations required by the different versions of this approach when the Lagrange multipliers are known, ALAG1, ALAG2 and ALAG3 perform similarly. ALAG1 is somewhat faster, but ALAG2 and ALAG3 achieve somewhat better quality solutions. There is no considerable change in the average violations -- for certain problems, there is some improvement, and for others, there is some worsening. Compared to CPLEX, the ALAG methods now generally require fewer iterations and comparable solution times. The exceptions are for problems that have a high ratio of the number of inequality constraints to the number of equality constraints.

Based on this observation, it might be conjectured that CPLEX performs better when the constraints are largely inequalities, while the performance of the augmented Lagrangian approach is not affected by the type of constraints. The active set approach ASL2 can also be expected to improve in relative performance when there exist only equality structural constraints. To investigate this behavior, we used the second class of test problems described above. The results obtained are discussed in Section 4.3 below.

4.3 Results for Randomly Generated Problems having Only Equality Constraints

For this class of problems (Problems 16-30), note that the algorithm ASL2 is the same as IEL2, since all the structural constraints are always in the active set. Hence, the differences in the solutions are created by the different search direction methods employed by the algorithms. Consequently, in Sections 4.3.1 and 4.3.2 below, we are actually discussing the results of two different search direction strategies implemented within the same algorithmic procedure.

4.3.1 Active Set Penalty Approach (ASL2):

As expected, in contrast to the results for the test problems in Section 4.2, the active set penalty approach performed considerably better for the set of problems having only equality constraints. As can be observed from Table 9, the average solution time to reach a final solution for this set of problems is comparable with that for CPLEX. For all of the test problems, a slight superoptimality was observed. The number of iterations as well as the CPU times to reach the stopping criterion was less than that for the other algorithms for almost all instances. Furthermore, for some problems, the number of iterations as well as the solution times were lesser than that for the CPLEX-based runs. The average constraint violations at termination are of the order of 10^{-3} to 10^{-4}.

4.3.2 Inequality-Equality Based Penalty Approach (IEL2):

This algorithm, as well as the augmented Lagrangian approach, was able to attain near optimal solutions for all the test problems. In accordance with the previous results, the average constraint violations for most of the problems were again the lowest among the investigated methods, being of the order of 10^{-4} to 10^{-6}. The number of iterations to reach the final solutions were always higher than those required by the ASL2 approach, and in general, comparable to the ALAG3 approach. Furthermore, for some problems, this number was lesser than the iterations performed by CPLEX. In an overall comparison, the relative performances of IEL2 and ASL2 were similar, with IEL2 yielding somewhat better quality solutions in more instances (8 versus 6, with one tie).

4.3.3 Augmented Lagrangian Approach (ALAG):

ALAG2 was again the fastest to reach near optimal solutions among the three variants of the ALAG approach, but the number of iterations were roughly 40% higher than those for ASL2. The average constraint violations attained at termination for all versions are of the order of 10^{-3} to 10^{-5}. Similar to the previous results, ALAG1 performed the poorest in terms of the number of iterations, the CPU times required to satisfy the stopping criterion and the quality of the solution, with a few exceptions.

Again, the test problems were run with known Lagrange multipliers. When these results are compared with the original algorithms, it is seen that the improvement in the number of iterations is significant. Feeding in the Lagrange multipliers caused the algorithms to perform better in terms of the number of iterations and CPU times than CPLEX for more test problems than for the previous class of test cases. Table 12 exhibits that the average CPU time to satisfy the stopping criterion for the ALAG1 and ALAG2 approaches (over the set of problems having only equality constraints) is lower than that for CPLEX. It should be pointed out here that the runs for CPLEX were performed on a Sun workstation and the runs for the investigated algorithms were performed on a Pentium II PC. Again, ALAG2 is the best of the three variants, in accordance with the results of Section 4.2, where ALAG2 again performed somewhat better under the similar experiment using both inequality and equality constrained problems.

4.4 NETLIB Test Problems

For the NETLIB test problems, it was observed that final solutions attained were either suboptimal or superoptimal under the set termination criteria. In particular, for Problem share1b, all of the algorithms performed poorly. Although the maximum allowable number of iterations was used for all algorithms, the average constraint violations at termination were at best only about 0.16 for this problem instance. In all of the runs, a maximum violation of 1.00 or higher (achieved for the same constraint) was found. It should be pointed out that Problem share1b was the only problem that the initial solution was a feasible solution. Further comments given below pertain to the remaining test problems solved from the NETLIB collection.

In general, it can be said that 75% of the runs reached a slightly superoptimal or a slightly suboptimal solution with acceptable margins of infeasibility (10^{-3}-10^{-5}). Of the remaining problems, 50% terminated with lower LP objective values than the CPLEX optimal solutions, again with acceptable infeasibilities (10^{-3}-10^{-5}).

The number of iterations required to attain optimality was usually higher as compared with CPLEX for all of the problems solved using any approach, with a couple of exceptions. Problem bnl1 was solved with a lesser number of iterations than that required by CPLEX for all of the algorithms, although the CPU time was lower for CPLEX. Unlike problem bnl1, problem afiro was solved faster than CPLEX by all versions of the ALAG approach in terms of CPU time, whereas the number of iterations was higher for all the methods as compared with the iterations performed by CPLEX. Problem sc205 was solved with a lesser number of iterations and lesser CPU time than that required by CPLEX using all versions of the ALAG approach.

4.4.1 Active Set Penalty Approach (ASL2):

This method performed the best among the investigated approaches in terms of the number of iterations and CPU time to attain a near optimal solution. Furthermore, the final objective function values were the closest to the optimal objective function values. The order of average constraint violations at termination varied between 10^{-3} to 10^{-5}. The objective values of the solutions found to the problems were either close to or lower (superoptimal) than those for the CPLEX solutions, except for Problem agg2.

4.4.2 Inequality-Equality Based Penalty Approach (IEL2):

The performance of this algorithm was similar to the ASL2 approach for most of the problems, although the number of iterations to reach an optimum was always higher than the number required by CPLEX, and the execution time was somewhat higher than that for ASL2. Furthermore, the order of average constraint violations was the lowest, varying between 10^{-3}-10^{-8}. The objective values of the solutions found for most of the problems were in the neighborhood of the CPLEX solution values, but for some of the problems, these values were superoptimal.

4.4.3 Augmented Lagrangian Approach (ALAG):

Different versions of this approach performed quite differently. In terms of average constraint violations, all of the versions found solutions having violations within the range of an order of 10^{-3} to 10^{-5}. ALAG1 iterated the most to find optimal solutions, and this version was the least successful in achieving near optimal solutions. ALAG2 and ALAG3 attained objective values at termination that were either close to the CPLEX optimal solution values, or were somewhat lower (superoptimal), except for two problems. The exceptions were the NETLIB test problems adlittle and bandm (see Table 10) for which suboptimal solutions were reached. ALAG3 was the best among the augmented Lagrangian approaches, but in general, its performance was inferior to that of ASL2 and IEL2.

We also ran the ALAG versions with the optimal Lagrange multipliers specified as input. The number of iterations as well as the final solution values improved in most of the cases. In this case, the improvement in the number of iterations was more for ALAG1 and ALAG3 than for ALAG2. For some of the problems, inputting the Lagrange multipliers caused the algorithms to perform better than CPLEX in terms of the number of iterations as well as the solution time. The average constraint violations at termination were observed to be in the same range as noted previously.

4.5 Effect of Density

In Sections 4.2 – 4.4, different problem sizes and structures are employed to compare the algorithms described in Chapter 3. However, the density (sparsity) of the constraint coefficient matrix also plays an important role in regard to the relative performances of these methods. In this section, we investigate the effect of density on algorithmic performance for these methods.

The algorithms tested in this experiment are ASL2, IEL2, and ALAG2. ALAG1 and ALAG3 are not considered here since ALAG2 is established to be the best among the augmented Lagrangian methods in the previous discussions. Again as a benchmark, we also run CPLEX on these test instances.

First, five different problem different sizes were chosen. Problems of these sizes were generated using the method described in Section 4.1 using four different density parameters for each size. The properties of this set of 20 problems (Problems 31–50) are displayed in Table 4.1. Table 4.16 presents the results obtained for these problems. The four density parameters are selected as 0.05, 0.10, 0.15, or 0.25, which are relatively higher values than those used for the previously studied problems.

The results indicate that, in general, ASL2 is the fastest in attaining a final solution among the tested methods for most of the problem instances. However, it should be noted that ALAG2 is faster than ASL2 for most of the low-density problem runs. In accordance with the previous discussion, IEL2 achieves the lowest average and maximum constraint violations. For all the runs, the final objective function values are very close to the CPLEX solutions, although with one exception (Problem 32), they are all suboptimal.

Notice that for the two largest problems with 5% density, namely Problems 43 and 47, the CPLEX solution times are higher than the execution times of the employed algorithms. Furthermore, noting the superior performance of ALAG2 for the low-density problems, we generated an additional set of 15 problems having low density parameter values to further study this effect on larger sized problems.

The properties of these new test problems (Problems 51–65) are summarized in Table 4.1. The density parameter is set to 0.01, 0.006, or 0.003 in this experiment. Note that if the density is made even smaller, the problem becomes largely separable and the presolve option of CPLEX reduces the effective problem size to a relatively trivial level. This advantage is most apparent for Problem 19 in the considered test instances, when the preconditioner of CPLEX reduced the problem to 35 variables and no constraints.

The results obtained are presented in Table 4.17. This experiment explicitly shows that for highly sparse problems, ALAG2 is the fastest method, obtaining a final solution faster than the other algorithms for all of the test problems. Again, IEL2 attained the smallest average and maximum constraint violations, and all final objective function values are sufficiently close to those of CPLEX. In comparison with CPLEX, all of the tested methods attained a final solution faster for the set of large-scale low-density problems. Although our implementations of the methods are rudimentary in contrast to CPLEX, ALAG2 was sometimes as fast as requiring only 16-23% of the CPU effort consumed by CPLEX in achieving a final solution with an acceptable margin of infeasibility.

Examining the results for the problems 57–59, it can be concluded that for large, highly sparse problems, the exterior penalty methods considered in this thesis are favorable to the simplex method if some ε infeasibility is permitted. Note that for the problems 57–65, the value of mn is almost the same and is close to the available memory in the computer used to make the runs. Hence, larger problems could not be tested. Furthermore, it can be deduced that for a constant mn, the solution time of the runs decreases with a decreasing number of constraints.

4.6 List of Problems and Results

Table 4.1: List of problems with their properties.

Problem	# of variables	# of equality constraints	# of inequality constraints	density of A	optimal solution objective value	# of CPLEX iterations	CPLEX solution time
1	50	10	20	1.53×10^{-1}	-6.62	32	0.02
2	100	20	50	9.59×10^{-2}	-0.29	87	0.07
3	200	100	150	8.09×10^{-2}	27.1	406	1.63
4	300	100	100	8.15×10^{-2}	-8.00	318	1.30
5	400	50	300	5.04×10^{-2}	4.47	814	5.50
6	500	50	100	3.98×10^{-2}	-13.8	246	0.31
7	600	100	50	5.03×10^{-2}	16.88	252	0.48
8	750	100	200	2.96×10^{-2}	26.87	699	3.45
9	750	200	100	2.03×10^{-2}	52.14	534	2.00
10	1000	100	100	2.98×10^{-2}	3.81	415	0.95
11	1000	500	50	1.03×10^{-2}	-15.2	764	2.91
12	1200	250	300	9.97×10^{-3}	-1.60	1444	8.51
13	1500	300	200	9.87×10^{-3}	-14.06	1115	7.21
14	2000	100	400	9.84×10^{-3}	-14.88	1111	4.38
15	2000	500	100	9.93×10^{-3}	18.31	1185	17.52
16	100	25	0	1.09×10^{-1}	-7.00	15	0.03
17	100	40	0	9.38×10^{-2}	3.32	30	0.03
18	100	80	0	9.48×10^{-2}	12.02	67	0.06
19	200	20	0	8.63×10^{-2}	-6.35	0	0.04
20	200	100	0	7.99×10^{-2}	2.20	91	0.15
21	200	150	0	8.04×10^{-2}	30.13	139	0.40
22	500	100	0	4.94×10^{-2}	-40.92	97	0.15
23	500	200	0	4.89×10^{-2}	8.70	238	1.15
24	500	400	0	4.97×10^{-2}	-2.65	473	10.51
25	800	200	0	3.10×10^{-2}	30.9	245	0.85
26	800	400	0	3.06×10^{-2}	-6.13	593	12.05
27	800	700	0	3.03×10^{-3}	-13.17	806	55.98
28	2000	300	0	2.04×10^{-2}	-34.01	426	2.49
29	2000	400	0	1.02×10^{-2}	23.6	642	0.21
30	2000	800	0	9.99×10^{-3}	30.04	1272	65.82
adlittle	97	15	41	7.05×10^{-2}	4.89	87	0.03
afiro	32	8	19	9.61×10^{-2}	-1.06	13	0.03
agg2	302	60	456	2.75×10^{-2}	-19.9	134	0.12
bandm	472	305	0	1.73×10^{-2}	-80.9	286	0.30
bnl1	1175	232	411	6.78×10^{-3}	6.43	2001	1.75
brandy	249	166	54	4.74×10^{-2}	11.5	208	0.13
degen2	534	221	223	1.68×10^{-2}	-37.8	781	0.68
e226	282	33	190	4.10×10^{-2}	5.11	348	0.16
israel	142	0	174	9.18×10^{-2}	-72.12	144	0.12
lofti	308	95	58	2.29×10^{-2}	-0.151	153	0.08

Table 4.1: List of problems with their properties (continued).

Problem	# of variables	# of equality constraints	# of inequality constraints	density of A	optimal solution objective value	# of CPLEX iterations	CPLEX solution time
sc205	203	91	114	1.33×10^{-2}	-0.287	189	0.18
scorpion	358	280	108	1.03×10^{-2}	12.95	74	0.08
scsd1	760	77	0	4.08×10^{-2}	8.23	275	0.15
share1b	225	89	28	4.37×10^{-2}	-32.56	153	0.09
ship04s	1458	354	48	7.43×10^{-3}	14.28	170	0.12
31	500	300	100	5.04×10^{-2}	-32.06	606	11.05
32	500	300	100	10.06×10^{-2}	56.50	1172	30.91
33	500	300	100	15.06×10^{-2}	-28.42	709	21.92
34	500	300	100	25.12×10^{-2}	-2.93	643	23.12
35	800	200	250	5.06×10^{-2}	-10.11	989	22.24
36	800	200	250	10.08×10^{-2}	7.46	1135	38.97
37	800	200	250	15.05×10^{-2}	6.34	1089	45.43
38	800	200	250	25.12×10^{-2}	6.90	1029	41.95
39	1000	300	250	5.02×10^{-2}	-26.66	1550	64.14
40	1000	300	250	10.04×10^{-2}	47.55	1212	71.99
41	1000	300	250	15.01×10^{-2}	-10.27	1268	84.35
42	1000	300	250	25.03×10^{-2}	3.04	1120	85.53
43	1200	400	400	5.00×10^{-2}	-30.77	2023	202.20
44	1200	400	400	10.05×10^{-2}	-39.91	1865	248.62
45	1200	400	400	15.04×10^{-2}	-0.24	1837	274.57
46	1200	400	400	24.96×10^{-2}	36.37	1611	258.85
47	1500	500	500	5.00×10^{-2}	-12.35	3039	524.74
48	1500	500	500	10.03×10^{-2}	-8.75	2769	615.90
49	1500	500	500	15.01×10^{-2}	-50.87	2593	644.82
50	1500	500	500	25.01×10^{-2}	-51.71	2435	752.08
51	2000	499	999	2.99×10^{-3}	21.77	4046	38.88
52	2000	500	1000	5.97×10^{-3}	-5.22	4842	295.49
53	2000	500	1000	9.92×10^{-3}	44.51	5545	768.67
54	2500	1000	1000	2.99×10^{-3}	-60.35	6177	283.95
55	2500	1000	1000	5.95×10^{-3}	90.55	6922	1369.04
56	2500	1000	1000	9.94×10^{-3}	23.95	6944	2931.20
57	3000	1500	1200	3.03×10^{-3}	-5.23	8964	1558.82
58	3000	1500	1200	6.02×10^{-3}	-25.42	8475	5208.53
59	3000	1500	1200	10.01×10^{-3}	-47.19	8508	8903.53
60	4000	1200	800	3.00×10^{-3}	-64.48	6984	566.28
61	4000	1200	800	5.98×10^{-3}	-69.36	6269	2026.31
62	4000	1200	800	9.99×10^{-3}	26.53	7386	3959.22
63	5000	600	1000	3.00×10^{-3}	-26.01	5397	161.25
64	5000	600	1000	6.01×10^{-3}	-12.39	6413	881.41
65	5000	600	1000	10.04×10^{-3}	34.14	5963	1505.93

Table 4.2: Results for randomly generated problems.

ASL2	1	2	3	4	5
LP min value	-3.94	25.46	31.01	-7.97	5.11
# of iterations	382	382	382	382	382
av. violation	$7.73(10^{-5})$	$1.27(10^{-3})$	$2.40(10^{-4})$	$7.47(10^{-5})$	$1.31(10^{-4})$
max. violation	$2.24(10^{-3})$ (constr 13)	$6.37(10^{-2})$ (constr 27)	$1.50(10^{-2})$ (constr 59)	$5.72(10^{-3})$ (constr 96)	$1.08(10^{-2})$ (constr 154)
Exec. time	0.16	0.49	3.62	5.33	9.72
IEL2					
LP min value	-6.50	-0.20	29.76	-7.86	4.71
# of iterations	382	382	736	382	382
av. violation	$1.96(10^{-6})$	$2.02(10^{-6})$	$3.08(10^{-6})$	$2.27(10^{-6})$	$1.80(10^{-6})$
max. violation	$9.22(10^{-6})$ (constr 26)	$8.91(10^{-6})$ (constr 58)	$2.04(10^{-5})$ (constr 152)	$1.00(10^{-5})$ (constr 133)	$1.06(10^{-5})$ (constr 302)
Exec. time	0.22	0.60	8.46	6.53	11.53
ALAG1					
LP min value	-6.54	-0.19	29.04	-7.65	5.20
# of iterations	110	433	520	465	410
# of outer loops	1	1	1	1	1
av. violation	$1.17(10^{-3})$	$4.89(10^{-4})$	$8.36(10^{-4})$	$7.52(10^{-4})$	$7.18(10^{-4})$
max. violation	$9.88(10^{-3})$ (constr 25)	$6.95(10^{-3})$ (constr 68)	$9.78(10^{-3})$ (constr 167)	$6.25(10^{-3})$ (constr 116)	$8.88(10^{-3})$ (constr 336)
Exec. time	0.05	0.66	6.09	7.58	12.57
ALAG2					
LP min value	-6.49	-0.25	28.37	-7.93	4.66
# of iterations	105	166	335	249	272
# of outer loops	2	2	2	2	2
av. violation	$7.42(10^{-5})$	$1.12(10^{-4})$	$2.29(10^{-4})$	$2.13(10^{-4})$	$5.50(10^{-5})$
max. violation	$5.28(10^{-4})$ (constr 29)	$2.51(10^{-3})$ (constr 70)	$6.31(10^{-3})$ (constr 218)	$9.50(10^{-3})$ (constr 190)	$3.27(10^{-3})$ (constr 13)
Exec. time	0.11	0.22	3.84	4.23	8.57
ALAG3					
LP min value	-6.52	-0.26	29.47	-7.55	4.66
# of iterations	104	193	303	231	231
# of outer loops	2	3	4	3	3
av. violation	$9.24(10^{-4})$	$1.76(10^{-4})$	$1.34(10^{-3})$	$7.48(10^{-4})$	$2.30(10^{-4})$
max. violation	$6.01(10^{-3})$ (constr 30)	$2.17(10^{-3})$ (constr 4)	$8.54(10^{-3})$ (constr 166)	$5.36(10^{-3})$ (constr 175)	$4.72(10^{-3})$ (constr 121)
Exec. time	0.05	0.27	3.68	3.90	7.64

Table 4.2: Results for randomly generated problems (continued).

ASL2	6	7	8	9	10
LP min value	-13.78	16.63	28.14	52.25	3.93
# of iterations	382	231	382	382	382
av. violation	$5.59(10^{-5})$	$2.94(10^{-4})$	$2.41(10^{-4})$	$5.79(10^{-6})$	$8.57(10^{-5})$
max. violation	$6.02(10^{-3})$ (constr 63)	$2.63(10^{-3})$ (constr 45)	$1.17(10^{-2})$ (constr 43)	$7.02(10^{-4})$ (constr 64)	$5.37(10^{-3})$ (constr 61)
Exec. time	5.11	4.78	14.33	13.51	14.17
IEL2					
LP min value	-13.76	16.90	28.16	52.41	4.24
# of iterations	382	382	382	382	382
av. violation	$2.01(10^{-6})$	$2.75(10^{-6})$	$1.91(10^{-6})$	$2.74(10^{-6})$	$2.15(10^{-6})$
max. violation	$1.01(10^{-5})$ (constr 120)	$9.84(10^{-6})$ (constr 110)	$1.04(10^{-5})$ (constr 214)	$1.00(10^{-5})$ (constr 225)	$9.85(10^{-6})$ (constr 117)
Exec. time	5.82	8.57	16.64	13.84	15.38
ALAG1					
LP min value	-13.74	17.06	27.14	52.67	3.97
# of iterations	165	405	420	420	404
# of outer loops	2	1	1	1	1
av. violation	$4.42(10^{-4})$	$5.65(10^{-4})$	$5.88(10^{-4})$	$1.29(10^{-3})$	$4.70(10^{-4})$
max. violation	$6.04(10^{-3})$ (constr 25)	$6.45(10^{-3})$ (constr 146)	$5.72(10^{-3})$ (constr 292)	$9.49(10^{-3})$ (constr 50)	$5.68(10^{-3})$ (constr 190)
Exec. time	2.86	9.29	19.28	15.77	16.26
ALAG2					
LP min value	-13.75	16.89	27.09	52.16	3.93
# of iterations	164	192	270	270	270
# of outer loops	2	2	2	2	2
av. violation	$6.91(10^{-5})$	$6.82(10^{-5})$	$7.79(10^{-5})$	$7.86(10^{-5})$	$4.08(10^{-5})$
max. violation	$3.56(10^{-3})$ (constr 150)	$1.81(10^{-3})$ (constr 124)	$5.27(10^{-3})$ (constr 9)	$5.80(10^{-3})$ (constr 50)	$5.83(10^{-3})$ (constr 21)
Exec. time	2.69	4.39	11.48	10.00	11.04
ALAG3					
LP min value	-13.74	16.89	27.23	52.48	3.95
# of iterations	231	217	231	303	303
# of outer loops	3	3	3	4	4
av. violation	$1.70(10^{-4})$	$1.40(10^{-4})$	$5.26(10^{-4})$	$3.69(10^{-4})$	$1.34(10^{-4})$
max. violation	$5.78(10^{-3})$ (constr 25)	$3.78(10^{-3})$ (constr 95)	$5.72(10^{-3})$ (constr 223)	$6.94(10^{-3})$ (constr 254)	$8.58(10^{-3})$ (constr 185)
Exec. time	3.74	5.21	9.89	11.53	12.31

Table 4.2: Results for randomly generated problems (continued).

ASL2	11	12	13	14	15
LP min value	-13.75	-0.98	-13.92	-14.88	18.30
# of iterations	303	457	572	770	770
av. violation	$3.75(10^{-5})$	$9.26(10^{-5})$	$9.64(10^{-5})$	$1.21(10^{-4})$	$2.50(10^{-5})$
max. violation	$2.34(10^{-3})$ (constr 29)	$9.65(10^{-3})$ (constr 22)	$1.03(10^{-2})$ (constr 60)	$9.90(10^{-3})$ (constr 208)	$8.92(10^{-3})$ (constr 92)
Exec. time	23.29	41.14	59.59	106.17	132.81
IEL2					
LP min value	-12.85	-0.87	-13.59	-14.87	18.33
# of iterations	382	457	572	770	770
av. violation	$7.07(10^{-5})$	$2.20(10^{-6})$	$2.45(10^{-6})$	$1.74(10^{-6})$	$3.09(10^{-6})$
max. violation	$5.56(10^{-3})$ (constr 173)	$1.02(10^{-5})$ (constr 431)	$1.03(10^{-5})$ (constr 237)	$9.81(10^{-6})$ (constr 427)	$9.91(10^{-6})$ (constr 420)
Exec. time	29.93	41.70	60.41	110.13	134.29
ALAG1					
LP min value	-14.60	-1.08	-13.78	-19.25	19.06
# of iterations	575	624	558	544	820
# of outer loops	1	1	1	0	1
av. violation	$9.69(10^{-4})$	$3.93(10^{-4})$	$8.73(10^{-4})$	$2.37(10^{-3})$	$8.83(10^{-4})$
max. violation	$4.52(10^{-3})$ (constr 282)	$5.51(10^{-3})$ (constr 513)	$6.76(10^{-3})$ (constr 304)	$9.99(10^{-3})$ (constr 39)	$6.79(10^{-3})$ (constr 483)
Exec. time	43.72	58.45	60.70	80.52	145.00
ALAG2					
LP min value	-14.16	-1.13	-13.95	-14.87	18.27
# of iterations	330	324	404	254	317
# of outer loops	2	2	2	2	2
av. violation	$3.22(10^{-4})$	$6.24(10^{-5})$	$3.15(10^{-5})$	$9.10(10^{-5})$	$1.53(10^{-4})$
max. violation	$3.54(10^{-2})$ (constr 34)	$2.79(10^{-3})$ (constr 35)	$1.36(10^{-3})$ (constr 44)	$1.28(10^{-2})$ (constr 149)	$2.80(10^{-2})$ (constr 21)
Exec. time	25.65	30.76	43.61	38.45	56.90
ALAG3					
LP min value	-13.33	-1.08	-13.78	-14.85	18.37
# of iterations	468	363	454	432	459
# of outer loops	6	4	4	3	3
av. violation	$7.85(10^{-4})$	$1.76(10^{-4})$	$1.16(10^{-4})$	$1.32(10^{-5})$	$6.92(10^{-5})$
max. violation	$8.72(10^{-3})$ (constr 318)	$4.45(10^{-3})$ (constr 538)	$2.84(10^{-3})$ (constr 218)	$4.62(10^{-4})$ (constr 406)	$4.51(10^{-3})$ (constr 27)
Exec. time	35.75	34.50	49.54	66.52	81.57

Table 4.3: Results for randomly generated problems having only equality constraints.

ASL2	16	17	18	19	20
LP min value	-7.56	2.37	11.84	-6.75	-0.16
# of iterations	84	123	231	65	165
av. violation	$3.31(10^{-3})$	$3.64(10^{-3})$	$5.94(10^{-4})$	$3.34(10^{-3})$	$3.36(10^{-3})$
max. violation	$9.79(10^{-3})^{(\text{constr } 11)}$	$9.71(10^{-3})^{(\text{constr } 1)}$	$1.84(10^{-3})^{(\text{constr } 47)}$	$8.73(10^{-3})^{(\text{constr } 3)}$	$9.98(10^{-3})^{(\text{constr } 41)}$
Exec. time	0.05	0.17	0.39	0.11	0.66
IEL2					
LP min value	-7.00	3.32	11.95	-6.35	2.19
# of iterations	264	382	382	139	382
av. violation	$3.31(10^{-6})$	$3.64(10^{-6})$	$3.20(10^{-3})$	$3.34(10^{-6})$	$3.75(10^{-6})$
max. violation	$9.79(10^{-6})^{(\text{constr } 11)}$	$9.70(10^{-6})^{(\text{constr } 1)}$	$1.60(10^{-2})^{(\text{constr } 45)}$	$8.73(10^{-6})^{(\text{constr } 3)}$	$1.09(10^{-5})^{(\text{constr } 61)}$
Exec. time	0.22	0.39	0.65	0.22	1.65
ALAG1					
LP min value	-6.86	3.41	12.11	-6.35	2.24
# of iterations	290	160	870	86	465
# of outer loops	1	1	2	2	1
av. violation	$3.12(10^{-3})$	$2.13(10^{-3})$	$2.70(10^{-3})$	$8.60(10^{-4})$	$1.91(10^{-3})$
max. violation	$9.37(10^{-3})^{(\text{constr } 24)}$	$6.29(10^{-3})^{(\text{constr } 33)}$	$9.63(10^{-3})^{(\text{constr } 13)}$	$7.85(10^{-3})^{(\text{constr } 18)}$	$9.43(10^{-3})^{(\text{constr } 72)}$
Exec. time	0.22	0.11	1.32	0.17	1.76
ALAG2					
LP min value	-7.00	3.32	11.96	-6.35	2.21
# of iterations	123	296	515	77	208
# of outer loops	2	3	2	2	2
av. violation	$6.88(10^{-5})$	$6.31(10^{-5})$	$9.17(10^{-4})$	$1.15(10^{-4})$	$2.46(10^{-4})$
max. violation	$4.26(10^{-4})^{(\text{constr } 24)}$	$5.66(10^{-4})^{(\text{constr } 39)}$	$9.70(10^{-3})^{(\text{constr } 79)}$	$7.17(10^{-4})^{(\text{constr } 15)}$	$8.20(10^{-3})^{(\text{constr } 44)}$
Exec. time	0.11	0.33	0.82	0.11	0.88
ALAG3					
LP min value	-7.01	3.28	12.07	-6.35	2.24
# of iterations	69	111	1038	44	228
# of outer loops	2	2	11	2	3
av. violation	$9.41(10^{-4})$	$9.50(10^{-4})$	$1.51(10^{-3})$	$3.40(10^{-4})$	$1.35(10^{-3})$
max. violation	$4.04(10^{-3})^{(\text{constr } 25)}$	$3.13(10^{-3})^{(\text{constr } 6)}$	$4.89(10^{-3})^{(\text{constr } 36)}$	$1.48(10^{-3})^{(\text{constr } 18)}$	$8.90(10^{-3})^{(\text{constr } 85)}$
Exec. time	0.06	0.11	1.54	0.06	0.88

Table 4.3: Results for randomly generated problems having only equality constraints (continued).

ASL2	21	22	23	24	25
LP min value	29.76	-43.54	4.16	-3.60	26.20
# of iterations	231	110	150	231	139
av. violation	$1.36(10^{-3})$	$3.69(10^{-3})$	$3.35(10^{-3})$	$9.41(10^{-4})$	$3.46(10^{-3})$
max. violation	$4.43(10^{-3})^{(\text{constr } 22)}$	$9.79(10^{-3})^{(\text{constr } 4)}$	$9.90(10^{-3})^{(\text{constr } 75)}$	$4.38(10^{-3})^{(\text{constr } 77)}$	$9.86(10^{-3})^{(\text{constr } 89)}$
Exec. time	1.38	1.15	3.52	10.21	4.56
IEL2					
LP min value	29.91	-40.92	8.70	-2.85	30.92
# of iterations	382	275	382	382	327
av. violation	$2.88(10^{-3})$	$3.70(10^{-6})$	$3.35(10^{-6})$	$1.20(10^{-3})$	$3.46(10^{-6})$
max. violation	$1.31(10^{-2})^{(\text{constr } 125)}$	$9.79(10^{-6})^{(\text{constr } 4)}$	$9.92(10^{-6})^{(\text{constr } 75)}$	$5.39(10^{-3})^{(\text{constr } 65)}$	$9.86(10^{-6})^{(\text{constr } 89)}$
Exec. time	2.81	3.24	8.78	17.41	10.43
ALAG1					
LP min value	29.95	-40.67	8.91	-2.67	31.30
# of iterations	870	390	465	930	403
# of outer loops	2	1	1	2	1
av. violation	$1.83(10^{-3})$	$1.29(10^{-3})$	$1.11(10^{-3})$	$1.22(10^{-3})$	$1.52(10^{-3})$
max. violation	$8.14(10^{-3})^{(\text{constr } 9)}$	$6.19(10^{-3})^{(\text{constr } 74)}$	$6.67(10^{-3})^{(\text{constr } 162)}$	$8.29(10^{-3})^{(\text{constr } 200)}$	$7.40(10^{-3})^{(\text{constr } 141)}$
Exec. time	5.27	4.34	10.21	40.26	12.41
ALAG2					
LP min value	30.03	-40.92	8.70	-3.10	30.94
# of iterations	455	138	215	275	143
# of outer loops	2	2	2	2	2
av. violation	$6.22(10^{-4})$	$9.73(10^{-5})$	$8.05(10^{-5})$	$1.54(10^{-3})$	$2.16(10^{-4})$
max. violation	$1.99(10^{-2})^{(\text{constr } 34)}$	$3.12(10^{-3})^{(\text{constr } 79)}$	$4.34(10^{-3})^{(\text{constr } 137)}$	$1.11(10^{-2})^{(\text{constr } 337)}$	$1.17(10^{-2})^{(\text{constr } 80)}$
Exec. time	2.91	1.65	5.27	12.86	4.89
ALAG3					
LP min value	29.98	-40.92	8.74	-2.70	31.01
# of iterations	1038	156	303	1038	231
# of outer loops	9	3	4	11	3
av. violation	$1.42(10^{-3})$	$8.25(10^{-5})$	$2.24(10^{-4})$	$9.41(10^{-4})$	$1.90(10^{-4})$
max. violation	$6.72(10^{-3})^{(\text{constr } 79)}$	$3.13(10^{-4})^{(\text{constr } 4)}$	$4.50(10^{-3})^{(\text{constr } 192)}$	$5.54(10^{-3})^{(\text{constr } 378)}$	$1.14(10^{-3})^{(\text{constr } 31)}$
Exec. time	6.20	1.65	6.75	45.86	7.14

Table 4.3: Results for randomly generated problems having only equality constraints (continued).

ASL2	26	27	28	29	30
LP min value	-6.97	-14.96	-41.08	14.21	12.79
# of iterations	231	231	196	246	310
av. violation	$3.24(10^{-4})$	$1.83(10^{-3})$	$3.40(10^{-3})$	$3.43(10^{-3})$	$3.28(10^{-3})$
max. violation	$9.98(10^{-4})$ (constr 167)	$6.94(10^{-3})$ (constr 356)	$1.00(10^{-2})$ (constr 83)	$9.97(10^{-3})$ (constr 17)	$9.95(10^{-3})$ (constr 527)
Exec. time	13.95	24.94	22.73	29.06	73.33
IEL2					
LP min value	-6.14	-13.40	-34.02	23.60	30.03
# of iterations	382	382	318	420	770
av. violation	$3.31(10^{-6})$	$1.58(10^{-3})$	$3.41(10^{-6})$	$3.43(10^{-6})$	$3.28(10^{-6})$
max. violation	$1.05(10^{-5})$ (constr 206)	$6.11(10^{-3})$ (constr 97)	$1.00(10^{-5})$ (constr 83)	$9.97(10^{-6})$ (constr 17)	$9.95(10^{-6})$ (constr 527)
Exec. time	23.56	41.85	37.20	49.60	180.59
ALAG1					
LP min value	-5.73	-13.02	-33.99	24.39	30.32
# of iterations	520	870	582	779	820
# of outer loops	1	2	1	1	1
av. violation	$1.15(10^{-3})$	$1.82(10^{-3})$	$5.68(10^{-4})$	$1.28(10^{-3})$	$1.08(10^{-3})$
max. violation	$8.57(10^{-3})$ (constr 236)	$7.84(10^{-3})$ (constr 695)	$3.89(10^{-3})$ (constr 230)	$7.63(10^{-3})$ (constr 310)	$9.29(10^{-3})$ (constr 481)
Exec. time	31.20	96.23	68.60	92.55	195.20
ALAG2					
LP min value	-6.09	-13.07	-34.02	23.59	30.07
# of iterations	270	270	212	219	338
# of outer loops	2	2	1	1	2
av. violation	$5.70(10^{-4})$	$1.83(10^{-3})$	$7.05(10^{-5})$	$5.20(10^{-5})$	$3.31(10^{-4})$
max. violation	$2.59(10^{-2})$ (constr 284)	$2.94(10^{-2})$ (constr 576)	$1.80(10^{-3})$ (constr 233)	$2.48(10^{-4})$ (constr 104)	$2.43(10^{-2})$ (constr 765)
Exec. time	17.36	31.36	25.21	26.20	81.07
ALAG3					
LP min value	-6.24	-13.32	-33.99	23.68	30.25
# of iterations	382	1038	236	323	610
# of outer loops	5	11	2	3	4
av. violation	$3.22(10^{-4})$	$1.07(10^{-3})$	$5.54(10^{-4})$	$7.63(10^{-5})$	$2.43(10^{-4})$
max. violation	$3.91(10^{-3})$ (constr 30)	$7.39(10^{-3})$ (constr 173)	$4.78(10^{-3})$ (constr 31)	$5.20(10^{-4})$ (constr 279)	$1.84(10^{-3})$ (constr 194)
Exec. time	23.18	112.43	28.12	38.28	143.74

Table 4.4: Results for NETLIB test problems.

ASL2	adlittle	afiro	agg2	bandm	bnl1
LP min value	2.54	-0.99	1064.42	-56.41	2.10
# of iterations	373	382	623	382	437
av. violation	$1.51(10^{-3})$	$7.71(10^{-5})$	$1.73(10^{-4})$	$1.68(10^{-3})$	$5.76(10^{-4})$
max. violation	$1.06(10^{-2})$ (constr 19)	$2.08(10^{-3})$ (constr 14)	$4.55(10^{-2})$ (constr 344)	$2.59(10^{-2})$ (constr 131)	$2.35(10^{-2})$ (constr 19)
Exec. time	0.38	0.11	13.51	8.79	41.91
IEL2					
LP min value	1.40	-1.06	-20.08	-51.03	1.63
# of iterations	382	382	382	623	437
av. violation	$1.12(10^{-3})$	$8.10(10^{-8})$	$1.59(10^{-5})$	$1.11(10^{-3})$	$3.11(10^{-4})$
max. violation	$8.89(10^{-3})$ (constr 42)	$6.20(10^{-7})$ (constr 24)	$3.17(10^{-3})$ (constr 481)	$8.92(10^{-3})$ (constr 131)	$7.01(10^{-3})$ (constr 157)
Exec. time	0.50	0.16	10.55	15.43	45.97
ALAG1					
LP min value	44.35	-0.03	-23.65	-28.37	1.35
# of iterations	1326	19	623	7122	861
# of outer loops	4	0	3	12	2
av. violation	$7.17(10^{-4})$	$2.31(10^{-3})$	$3.43(10^{-4})$	$1.77(10^{-3})$	$7.88(10^{-4})$
max. violation	$8.60(10^{-3})$ (constr 37)	$8.65(10^{-3})$ (constr 23)	$5.08(10^{-3})$ (constr 482)	$3.86(10^{-2})$ (constr 132)	$7.41(10^{-3})$ (constr 228)
Exec. time	1.54	0.00	15.93	153.18	85.80
ALAG2					
LP min value	26.57	-0.03	-22.34	-48.20	1.66
# of iterations	961	80	395	1963	645
# of outer loops	3	2	2	5	2
av. violation	$2.18(10^{-4})$	$1.09(10^{-4})$	$6.77(10^{-5})$	$3.13(10^{-3})$	$4.31(10^{-4})$
max. violation	$6.10(10^{-3})$ (constr 20)	$1.20(10^{-3})$ (constr 27)	$1.32(10^{-2})$ (constr 397)	$1.26(10^{-1})$ (constr 132)	$1.90(10^{-2})$ (constr 27)
Exec. time	1.26	0.05	10.76	45.76	67.78
ALAG3					
LP min value	14.03	-0.01	-20.68	-39.53	1.97
# of iterations	902	71	382	1187	871
# of outer loops	10	2	5	12	9
av. violation	$6.58(10^{-4})$	$5.66(10^{-4})$	$1.19(10^{-4})$	$3.87(10^{-3})$	$2.74(10^{-4})$
max. violation	$9.36(10^{-3})$ (constr 56)	$5.02(10^{-3})$ (constr 10)	$7.53(10^{-3})$ (constr 446)	$7.80(10^{-3})$ (constr 132)	$6.45(10^{-3})$ (constr 158)
Exec. time	1.04	0.06	10.00	25.92	88.65

Table 4.4: Results for NETLIB test problems (continued).

ASL2	brandy	degen2	e226	israel	lofti
LP min value	3.62	-45.02	5.44	-85.84	0.05
# of iterations	382	382	382	352	382
av. violation	$2.31(10^{-3})$	$4.08(10^{-3})$	$1.26(10^{-3})$	$1.05(10^{-3})$	$3.89(10^{-3})$
max. violation	$2.15(10^{-2})$ (constr 49)	$5.43(10^{-2})$ (constr 355)	$3.53(10^{-2})$ (constr 128)	$2.09(10^{-2})$ (constr 44)	$2.39(10^{-2})$ (constr 67)
Exec. time	3.63	13.90	5.00	1.75	2.97
IEL2					
LP min value	4.99	-35.57	6.58	-88.30	0.06
# of iterations	623	382	623	382	382
av. violation	$1.37(10^{-3})$	$5.33(10^{-4})$	$9.35(10^{-4})$	$1.81(10^{-4})$	$1.23(10^{-3})$
max. violation	$1.01(10^{-2})$ (constr 49)	$3.09(10^{-3})$ (constr 270)	$3.31(10^{-2})$ (constr 222)	$7.10(10^{-3})$ (constr 44)	$6.79(10^{-3})$ (constr 144)
Exec. time	6.70	15.44	9.78	3.46	3.02
ALAG1					
LP min value	10.44	-35.84	6.27	-82.05	0.10
# of iterations	7122	750	870	595	784
# of outer loops	12	2	2	2	3
av. violation	$3.82(10^{-3})$	$9.63(10^{-4})$	$6.13(10^{-4})$	$4.38(10^{-4})$	$7.30(10^{-4})$
max. violation	$3.10(10^{-2})$ (constr 61)	$8.71(10^{-3})$ (constr 74)	$6.24(10^{-3})$ (constr 141)	$8.17(10^{-3})$ (constr 89)	$6.65(10^{-3})$ (constr 114)
Exec. time	64.88	27.90	12.63	4.78	5.98
ALAG2					
LP min value	10.50	-35.40	8.50	-80.35	0.09
# of iterations	6127	315	1186	799	560
# of outer loops	12	1	3	3	2
av. violation	$4.86(10^{-3})$	$8.68(10^{-4})$	$4.60(10^{-4})$	$5.48(10^{-5})$	$3.70(10^{-4})$
max. violation	$6.07(10^{-2})$ (constr 102)	$1.31(10^{-2})$ (constr 356)	$1.52(10^{-2})$ (constr 8)	$5.27(10^{-3})$ (constr 160)	$8.78(10^{-3})$ (constr 101)
Exec. time	55.69	11.97	17.20	6.43	4.34
ALAG3					
LP min value	5.25	-35.16	7.33	-82.64	0.09
# of iterations	1187	382	778	773	778
# of outer loops	12	5	9	7	8
av. violation	$8.70(10^{-4})$	$7.22(10^{-4})$	$3.26(10^{-4})$	$1.80(10^{-4})$	$4.12(10^{-4})$
max. violation	$5.91(10^{-3})$ (constr 49)	$7.81(10^{-3})$ (constr 275)	$7.62(10^{-3})$ (constr 160)	$3.60(10^{-3})$ (constr 44)	$1.86(10^{-3})$ (constr 83)
Exec. time	11.27	14.61	11.37	6.05	5.77

Table 4.4: Results for NETLIB test problems (continued).

ASL2	sc205	scorpion	scsd1	share1b	ship04s
LP min value	-0.24	12.68	10.08	-118.16	5.88
# of iterations	382	332	192	382	311
av. violation	$1.09(10^{-3})$	$2.93(10^{-3})$	$1.29(10^{-3})$	$2.07(10^{-1})$	$6.01(10^{-3})$
max. violation	$2.60(10^{-2})^{(\text{constr } 110)}$	$2.24(10^{-2})^{(\text{constr } 87)}$	$2.48(10^{-3})^{(\text{constr } 18)}$	$1.00(10^{0})^{(\text{constr } 33)}$	$2.10(10^{-2})^{(\text{constr } 79)}$
Exec. time	1.71	6.75	2.69	1.42	22.80
IEL2					
LP min value	0.00	10.75	10.32	-109.49	7.04
# of iterations	382	382	382	382	311
av. violation	$4.31(10^{-5})$	$5.12(10^{-3})$	$8.81(10^{-5})$	$2.11(10^{-1})$	$4.95(10^{-3})$
max. violation	$4.24(10^{-4})^{(\text{constr } 133)}$	$2.65(10^{-2})^{(\text{constr } 97)}$	$2.96(10^{-4})^{(\text{constr } 1)}$	$1.00(10^{0})^{(\text{constr } 33)}$	$1.77(10^{-2})^{(\text{constr } 90)}$
Exec. time	1.81	7.58	6.10	1.70	25.54
ALAG1					
LP min value	-0.01	13.92	10.75	-299.671	14.86
# of iterations	10	3099	721	7122	2841
# of outer loops	0	7	2	12	5
av. violation	$2.39(10^{-4})$	$6.29(10^{-4})$	$6.78(10^{-4})$	$1.62(10^{-1})$	$1.36(10^{-3})$
max. violation	$7.26(10^{-3})^{(\text{constr } 119)}$	$4.28(10^{-3})^{(\text{constr } 12)}$	$4.32(10^{-3})^{(\text{constr } 7)}$	$1.00(10^{0})^{(\text{constr } 33)}$	$5.49(10^{-3})^{(\text{constr } 226)}$
Exec. time	0.06	62.94	9.67	28.23	213.93
ALAG2					
LP min value	0.00	14.85	11.79	-299.26	21.11
# of iterations	30	1203	645	7122	1677
# of outer loops	0	3	2	12	3
av. violation	$1.25(10^{-4})$	$1.30(10^{-3})$	$2.93(10^{-4})$	$1.53(10^{-1})$	$1.58(10^{-3})$
max. violation	$3.18(10^{-3})^{(\text{constr } 17)}$	$1.03(10^{-1})^{(\text{constr } 152)}$	$6.94(10^{-3})^{(\text{constr } 5)}$	$1.00(10^{0})^{(\text{constr } 33)}$	$8.25(10^{-2})^{(\text{constr } 2)}$
Exec. time	0.16	23.07	8.40	27.95	127.93
ALAG3					
LP min value	-0.02	13.54	10.08	-245.26	12.85
# of iterations	6	1284	336	1187	571
# of outer loops	0	12	4	12	3
av. violation	$2.63(10^{-4})$	$5.90(10^{-4})$	$8.30(10^{-4})$	$1.59(10^{-1})$	$2.39(10^{-3})$
max. violation	$9.49(10^{-3})^{(\text{constr } 118)}$	$8.93(10^{-3})^{(\text{constr } 357)}$	$7.49(10^{-3})^{(\text{constr } 17)}$	$1.00(10^{0})^{(\text{constr } 33)}$	$7.03(10^{-3})^{(\text{constr } 22)}$
Exec. time	0.00	26.16	4.67	4.99	43.66

Table 4.5: Results for randomly generated problems with known Lagrange multipliers.

ALAG1	1	2	3	4	5
LP min value	-6.36	0.09	28.10	-7.68	4.52
# of iterations	21	66	77	35	46
av. violation	$1.32(10^{-3})$	$8.21(10^{-4})$	$1.08(10^{-3})$	$1.86(10^{-3})$	$7.16(10^{-4})$
max. violation	$8.78(10^{-3})^{(\text{constr } 28)}$	$6.01(10^{-3})^{(\text{constr } 65)}$	$8.66(10^{-3})^{(\text{constr } 51)}$	$9.95(10^{-3})^{(\text{constr } 137)}$	$7.78(10^{-3})^{(\text{constr } 349)}$
Exec. time	0.00	0.11	0.93	0.55	1.43
ALAG2					
LP min value	-6.41	0.18	28.32	-7.73	4.96
# of iterations	104	100	189	60	79
av. violation	$2.27(10^{-5})$	$5.22(10^{-5})$	$1.00(10^{-4})$	$2.85(10^{-4})$	$8.82(10^{-5})$
max. violation	$2.70(10^{-4})^{(\text{constr } 7)}$	$4.92(10^{-4})^{(\text{constr } 65)}$	$3.39(10^{-3})^{(\text{constr } 61)}$	$2.91(10^{-3})^{(\text{constr } 41)}$	$1.47(10^{-3})^{(\text{constr } 18)}$
Exec. time	0.06	0.16	2.41	1.05	2.81
ALAG3					
LP min value	-6.58	-0.31	27.34	-7.94	4.52
# of iterations	47	104	176	147	46
av. violation	$7.98(10^{-4})$	$8.80(10^{-4})$	$8.68(10^{-4})$	$2.32(10^{-4})$	$7.16(10^{-4})$
max. violation	$4.42(10^{-3})^{(\text{constr } 25)}$	$7.87(10^{-3})^{(\text{constr } 20)}$	$9.20(10^{-3})^{(\text{constr } 180)}$	$2.03(10^{-3})^{(\text{constr } 49)}$	$7.78(10^{-3})^{(\text{constr } 349)}$
Exec. time	0.06	0.17	2.19	2.86	1.49

Table 4.5: Results for randomly generated problems with known Lagrange multipliers (continued).

ALAG1	6	7	8	9	10
LP min value	-13.71	16.95	26.88	52.32	4.32
# of iterations	42	43	77	62	50
av. violation	$1.23(10^{-3})$	$1.02(10^{-3})$	$6.77(10^{-4})$	$1.32(10^{-3})$	$1.85(10^{-3})$
max. violation	$8.60(10^{-3})^{(\text{constr } 11)}$	$6.09(10^{-3})^{(\text{constr } 104)}$	$5.26(10^{-3})^{(\text{constr } 252)}$	$9.54(10^{-3})^{(\text{constr } 274)}$	$8.67(10^{-3})^{(\text{constr } 185)}$
Exec. time	0.66	0.94	3.51	2.52	2.31
ALAG2					
LP min value	-13.63	16.96	27.10	52.42	4.01
# of iterations	48	49	87	71	63
av. violation	$1.02(10^{-4})$	$1.99(10^{-4})$	$1.06(10^{-4})$	$2.33(10^{-4})$	$6.34(10^{-5})$
max. violation	$1.35(10^{-3})^{(\text{constr } 70)}$	$1.14(10^{-3})^{(\text{constr } 104)}$	$1.02(10^{-3})^{(\text{constr } 82)}$	$1.76(10^{-3})^{(\text{constr } 274)}$	$6.18(10^{-4})^{(\text{constr } 30)}$
Exec. time	0.84	1.20	4.23	2.97	2.85
ALAG3					
LP min value	-13.71	16.95	26.95	52.16	4.32
# of iterations	42	43	303	158	50
av. violation	$1.23(10^{-3})$	$1.02(10^{-3})$	$2.30(10^{-4})$	$9.96(10^{-4})$	$1.85(10^{-3})$
max. violation	$8.60(10^{-3})^{(\text{constr } 11)}$	$6.09(10^{-3})^{(\text{constr } 104)}$	$4.96(10^{-3})^{(\text{constr } 226)}$	$9.73(10^{-3})^{(\text{constr } 129)}$	$8.67(10^{-3})^{(\text{constr } 185)}$
Exec. time	0.66	0.99	13.08	5.93	2.25

Table 4.5: Results for randomly generated problems with known Lagrange multipliers (continued).

ALAG1	11	12	13	14	15
LP min value	-14.97	-1.49	-14.13	-14.78	18.17
# of iterations	223	77	106	91	96
av. violation	$1.22(10^{-3})$	$9.49(10^{-4})$	$1.05(10^{-3})$	$3.11(10^{-4})$	$1.30(10^{-3})$
max. violation	$6.58(10^{-3})^{(\text{constr } 377)}$	$6.58(10^{-3})^{(\text{constr } 330)}$	$6.33(10^{-3})^{(\text{constr } 307)}$	$3.18(10^{-3})^{(\text{constr } 486)}$	$7.43(10^{-3})^{(\text{constr } 322)}$
Exec. time	17.35	7.31	11.91	13.62	17.08
ALAG2					
LP min value	-14.99	-1.06	-13.73	-14.60	18.44
# of iterations	144	89	130	97	106
av. violation	$6.17(10^{-4})$	$1.45(10^{-4})$	$8.62(10^{-5})$	$2.86(10^{-5})$	$1.24(10^{-4})$
max. violation	$2.37(10^{-2})^{(\text{constr } 173)}$	$3.54(10^{-3})^{(\text{constr } 335)}$	$1.15(10^{-3})^{(\text{constr } 187)}$	$4.67(10^{-4})^{(\text{constr } 79)}$	$6.94(10^{-4})^{(\text{constr } 473)}$
Exec. time	11.26	8.73	14.72	15.00	19.28
ALAG3					
LP min value	-15.03	-1.65	-14.13	-14.78	18.17
# of iterations	382	198	192	91	96
av. violation	$9.15(10^{-4})$	$6.73(10^{-4})$	$7.25(10^{-4})$	$3.11(10^{-4})$	$1.30(10^{-3})$
max. violation	$8.39(10^{-3})^{(\text{constr } 202)}$	$7.49(10^{-3})^{(\text{constr } 336)}$	$8.61(10^{-3})^{(\text{constr } 428)}$	$3.18(10^{-3})^{(\text{constr } 486)}$	$7.43(10^{-3})^{(\text{constr } 322)}$
Exec. time	29.11	18.62	21.04	13.90	17.20

Table 4.6: Results for randomly generated problems having only equality constraints with known Lagrange multipliers.

ALAG1	16	17	18	19	20
LP min value	-6.89	3.51	12.09	-6.47	2.03
# of iterations	16	40	399	14	69
av. violation	$2.22(10^{-3})$	$2.45(10^{-3})$	$2.90(10^{-3})$	$2.09(10^{-3})$	$2.16(10^{-3})$
max. violation	$8.28(10^{-3})^{(\text{constr } 3)}$	$9.98(10^{-3})^{(\text{constr } 36)}$	$6.99(10^{-3})^{(\text{constr } 11)}$	$5.30(10^{-3})^{(\text{constr } 8)}$	$9.84(10^{-3})^{(\text{constr } 69)}$
Exec. time	0.00	0.05	0.60	0.06	0.28
ALAG2					
LP min value	-7.00	3.31	12.04	-6.35	2.19
# of iterations	25	48	485	24	72
av. violation	$8.86(10^{-5})$	$8.55(10^{-5})$	$1.24(10^{-3})$	$3.35(10^{-5})$	$7.85(10^{-5})$
max. violation	$6.74(10^{-4})^{(\text{constr } 2)}$	$1.00(10^{-3})^{(\text{constr } 12)}$	$7.40(10^{-3})^{(\text{constr } 45)}$	$1.73(10^{-4})^{(\text{constr } 9)}$	$1.17(10^{-3})^{(\text{constr } 6)}$
Exec. time	0.00	0.05	0.77	0.00	0.33
ALAG3					
LP min value	-6.96	3.25	12.05	-6.47	2.17
# of iterations	16	62	835	14	59
av. violation	$7.30(10^{-4})$	$1.12(10^{-3})$	$1.25(10^{-3})$	$2.09(10^{-3})$	$8.74(10^{-4})$
max. violation	$2.25(10^{-3})^{(\text{constr } 25)}$	$6.81(10^{-3})^{(\text{constr } 6)}$	$6.57(10^{-3})^{(\text{constr } 45)}$	$5.30(10^{-3})^{(\text{constr } 8)}$	$3.53(10^{-3})^{(\text{constr } 76)}$
Exec. time	0.00	0.06	1.26	0.05	0.22

Table 4.6: Results for randomly generated problems having only equality constraints with known Lagrange multipliers (continued).

ALAG1	21	22	23	24	25
LP min value	30.14	-40.84	8.47	-2.25	31.08
# of iterations	256	44	74	337	42
av. violation	$2.13(10^{-3})$	$1.49(10^{-3})$	$1.95(10^{-3})$	$2.12(10^{-3})$	$1.98(10^{-3})$
max. violation	$7.86(10^{-3})^{(\text{constr } 74)}$	$4.67(10^{-3})^{(\text{constr } 76)}$	$6.54(10^{-3})^{(\text{constr } 181)}$	$9.44(10^{-3})^{(\text{constr } 177)}$	$7.38(10^{-3})^{(\text{constr } 143)}$
Exec. time	1.55	0.48	1.64	15.11	1.32
ALAG2					
LP min value	30.16	-40.91	8.69	-2.29	30.94
# of iterations	256	43	77	282	51
av. violation	$7.49(10^{-4})$	$2.96(10^{-4})$	$3.00(10^{-4})$	$1.24(10^{-3})$	$1.95(10^{-4})$
max. violation	$2.21(10^{-3})^{(\text{constr } 78)}$	$1.43(10^{-3})^{(\text{constr } 36)}$	$2.88(10^{-3})^{(\text{constr } 6)}$	$2.21(10^{-2})^{(\text{constr } 14)}$	$1.02(10^{-3})^{(\text{constr } 143)}$
Exec. time	1.59	0.55	2.02	13.41	1.92
ALAG3					
LP min value	30.41	-40.90	8.68	-2.56	31.08
# of iterations	342	46	75	900	42
av. violation	$2.75(10^{-3})$	$2.80(10^{-4})$	$4.00(10^{-4})$	$7.67(10^{-4})$	$1.98(10^{-3})$
max. violation	$8.19(10^{-3})^{(\text{constr } 114)}$	$1.27(10^{-3})^{(\text{constr } 36)}$	$2.43(10^{-3})^{(\text{constr } 180)}$	$9.15(10^{-3})^{(\text{constr } 8)}$	$7.38(10^{-3})^{(\text{constr } 143)}$
Exec. time	2.03	0.59	1.64	39.38	1.32

Table 4.6: Results for randomly generated problems having only equality constraints with known Lagrange multipliers (continued).

ALAG1	26	27	28	29	30
LP min value	-6.68	-13.08	-33.93	23.47	29.89
# of iterations	129	533	64	88	148
av. violation	$1.56(10^{-3})$	$1.63(10^{-3})$	$6.13(10^{-4})$	$1.11(10^{-3})$	$1.03(10^{-3})$
max. violation	$6.13(10^{-3})^{(\text{constr } 383)}$	$8.67(10^{-3})^{(\text{constr } 424)}$	$3.69(10^{-3})^{(\text{constr } 290)}$	$3.78(10^{-3})^{(\text{constr } 217)}$	$3.74(10^{-3})^{(\text{constr } 496)}$
Exec. time	8.07	58.22	7.91	10.70	35.98
ALAG2					
LP min value	-6.18	-13.22	-34.01	23.59	30.04
# of iterations	140	221	68	89	148
av. violation	$3.27(10^{-4})$	$1.70(10^{-3})$	$4.52(10^{-5})$	$1.26(10^{-4})$	$1.92(10^{-4})$
max. violation	$3.16(10^{-3})^{(\text{constr } 35)}$	$8.72(10^{-3})^{(\text{constr } 255)}$	$3.44(10^{-4})^{(\text{constr } 263)}$	$1.72(10^{-3})^{(\text{constr } 87)}$	$1.34(10^{-3})^{(\text{constr } 471)}$
Exec. time	9.39	25.60	8.73	11.42	38.17
ALAG3					
LP min value	-6.22	-13.17	-33.93	23.60	29.94
# of iterations	231	1187	64	103	196
av. violation	$4.78(10^{-4})$	$6.91(10^{-4})$	$6.13(10^{-4})$	$1.62(10^{-4})$	$1.01(10^{-3})$
max. violation	$5.56(10^{-3})^{(\text{constr } 156)}$	$3.09(10^{-3})^{(\text{constr } 45)}$	$3.69(10^{-3})^{(\text{constr } 290)}$	$2.11(10^{-3})^{(\text{constr } 87)}$	$7.91(10^{-3})^{(\text{constr } 95)}$
Exec. time	13.73	127.81	7.75	12.63	47.07

Table 4.7: Results for NETLIB test problems with known Lagrange multipliers.

ALAG1	adlittle	afiro	agg2	bandm	bnl1
LP min value	9.10	-0.02	-18.19	-29.67	2.23
# of iterations	245	39	113	7122	821
av. violation	$1.08(10^{-3})$	$8.27(10^{-4})$	$1.93(10^{-4})$	$2.77(10^{-3})$	$5.97(10^{-4})$
max. violation	$9.51(10^{-3})^{(\text{constr } 42)}$	$5.08(10^{-3})^{(\text{constr } 10)}$	$7.75(10^{-3})^{(\text{constr } 469)}$	$5.60(10^{-2})^{(\text{constr } 132)}$	$9.72(10^{-3})^{(\text{constr } 123)}$
Exec. time	0.27	0.06	3.24	152.74	82.99
ALAG2					
LP min value	26.77	-0.04	-16.49	-21.25	2.26
# of iterations	1036	116	190	2746	351
av. violation	$2.19(10^{-4})$	$1.89(10^{-4})$	$3.73(10^{-5})$	$3.14(10^{-3})$	$4.66(10^{-4})$
max. violation	$7.93(10^{-3})^{(\text{constr } 20)}$	$7.89(10^{-4})^{(\text{constr } 27)}$	$5.17(10^{-3})^{(\text{constr } 124)}$	$1.68(10^{-1})^{(\text{constr } 132)}$	$2.43(10^{-2})^{(\text{constr } 396)}$
Exec. time	1.26	0.06	5.34	61.51	36.70
ALAG3					
LP min value	15.43	-0.03	-21.88	-17.01	2.12
# of iterations	821	55	216	1187	415
av. violation	$2.31(10^{-4})$	$7.13(10^{-4})$	$7.01(10^{-5})$	$4.91(10^{-3})$	$5.57(10^{-4})$
max. violation	$4.84(10^{-3})^{(\text{constr } 20)}$	$8.52(10^{-3})^{(\text{constr } 24)}$	$3.38(10^{-3})^{(\text{constr } 461)}$	$1.73(10^{-1})^{(\text{constr } 132)}$	$7.70(10^{-3})^{(\text{constr } 390)}$
Exec. time	0.94	0.05	5.77	25.82	42.30

Table 4.7: Results for NETLIB test problems with known Lagrange multipliers (continued).

ALAG1	brandy	degen2	e226	israel	lofti
LP min value	8.45	-36.87	7.09	-41.52	0.04
# of iterations	7122	197	496	27	1602
av. violation	$5.51(10^{-3})$	$8.77(10^{-4})$	$4.94(10^{-4})$	$2.32(10^{-4})$	$4.30(10^{-4})$
max. violation	$7.25(10^{-2})^{(\text{constr } 156)}$	$5.66(10^{-3})^{(\text{constr } 3)}$	$6.71(10^{-3})^{(\text{constr } 62)}$	$5.93(10^{-3})^{(\text{constr } 133)}$	$5.01(10^{-3})^{(\text{constr } 89)}$
Exec. time	64.93	7.63	7.36	0.22	11.59
ALAG2					
LP min value	6.88	-37.17	7.61	-42.04	0.04
# of iterations	6869	361	934	76	680
av. violation	$6.80(10^{-3})$	$3.54(10^{-4})$	$2.98(10^{-4})$	$2.38(10^{-5})$	$7.70(10^{-4})$
max. violation	$1.85(10^{-1})^{(\text{constr } 49)}$	$2.81(10^{-2})^{(\text{constr } 1)}$	$1.65(10^{-2})^{(\text{constr } 7)}$	$7.24(10^{-4})^{(\text{constr } 17)}$	$4.26(10^{-3})^{(\text{constr } 86)}$
Exec. time	82.50	13.62	13.40	0.60	5.05
ALAG3					
LP min value	5.57	-36.68	7.34	-51.69	0.08
# of iterations	1187	303	468	56	778
av. violation	$9.44(10^{-3})$	$7.64(10^{-4})$	$4.16(10^{-4})$	$2.10(10^{-4})$	$6.75(10^{-4})$
max. violation	$3.78(10^{-1})^{(\text{constr } 49)}$	$9.43(10^{-3})^{(\text{constr } 335)}$	$7.82(10^{-3})^{(\text{constr } 160)}$	$8.35(10^{-3})^{(\text{constr } 44)}$	$3.58(10^{-3})^{(\text{constr } 144)}$
Exec. time	11.10	11.59	6.92	0.44	5.76

Table 4.7: Results for NETLIB test problems with known Lagrange multipliers (continued).

ALAG1	sc205	scorpion	scsd1	share1b	ship04s
LP min value	0.00	13.14	8.21	-300.90	13.46
# of iterations	0	2506	253	7122	791
av. violation	0.00	$7.63(10^{-4})$	$2.09(10^{-3})$	$1.67(10^{-1})$	$1.83(10^{-3})$
max. violation	$0.00^{(\text{none})}$	$7.80(10^{-3})^{(\text{constr } 39)}$	$8.75(10^{-3})^{(\text{constr } 1)}$	$1.00(10^{0})^{(\text{constr } 33)}$	$7.59(10^{-3})^{(\text{constr } 129)}$
Exec. time	0.00	47.56	3.36	27.02	59.66
ALAG2					
LP min value	0.00	13.11	8.72	-299.76	14.51
# of iterations	0	688	548	7122	1339
av. violation	0.00	$1.12(10^{-3})$	$4.25(10^{-4})$	$1.52(10^{-1})$	$1.76(10^{-3})$
max. violation	$0.00^{(\text{none})}$	$1.20(10^{-2})^{(\text{constr } 355)}$	$5.56(10^{-3})^{(\text{constr } 9)}$	$1.00(10^{0})^{(\text{constr } 33)}$	$9.40(10^{-2})^{(\text{constr } 27)}$
Exec. time	0.00	13.23	7.03	26.25	101.72
ALAG3					
LP min value	0.00	12.95	8.18	-244.35	14.35
# of iterations	0	1187	365	1350	1278
av. violation	0.00	$9.89(10^{-4})$	$1.57(10^{-3})$	$1.58(10^{-1})$	$8.36(10^{-4})$
max. violation	$0.00^{(\text{none})}$	$1.07(10^{-2})^{(\text{constr } 355)}$	$9.78(10^{-3})^{(\text{constr } 16)}$	$1.00(10^{0})^{(\text{constr } 33)}$	$1.70(10^{-3})^{(\text{constr } 82)}$
Exec. time	0.00	22.85	4.72	5.33	97.56

Table 4.8: Averages of the results for generally structured randomly generated problems.

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
LP min value (opt = 5.01)	7.44	5.60	5.15	5.26	5.46
# of iterations	436.07	475.00	458.20	261.47	301.53
av. violation	$1.90(10^{-4})$	$6.86(10^{-6})$	$8.53(10^{-4})$	$1.12(10^{-4})$	$3.95(10^{-4})$
max. violation	$1.10(10^{-2})$	$3.81(10^{-4})$	$7.25(10^{-3})$	$8.23(10^{-3})$	$5.24(10^{-3})$
Exec. Time (CPLEX: 3.75 sec)	28.95	30.94	31.92	16.80	21.74

Table 4.9: Averages of the results for randomly generated problems having only equality constraints.

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
LP min value (opt = 2.98)	-1.55	2.00	2.22	2.02	2.05
# of iterations	182.87	371.27	566.67	250.27	456.33
av. violation	$2.62(10^{-3})$	$5.93(10^{-4})$	$1.57(10^{-3})$	$4.55(10^{-4})$	$6.81(10^{-4})$
max. violation	$7.75(10^{-3})$	$2.71(10^{-3})$	$7.76(10^{-3})$	$1.01(10^{-2})$	$3.94(10^{-3})$
Exec. Time (CPLEX: 9.99 sec)	12.41	25.24	37.32	14.07	27.73

Table 4.10: Averages of the results for standard NETLIB test problems.

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
LP min value (opt = -12.09)	-11.24	-10.95	-4.85	-7.02	-8.07
# of iterations	378.14	432.50	1910.21	1273.54	679.14
av. violation	$1.99(10^{-3})$	$1.21(10^{-3})$	$1.10(10^{-3})$	$1.06(10^{-3})$	$8.62(10^{-4})$
max. violation	$2.40(10^{-2})$	$9.51(10^{-3})$	$1.07(10^{-2})$	$3.55(10^{-2})$	$6.85(10^{-3})$
Exec. time (CPLEX: 0.29 sec)	8.99	10.86	47.09	29.28	17.80

Table 4.11: Averages of the results for generally structured randomly generated problems with known Lagrange multipliers.

	ALAG1	ALAG2	ALAG3
LP min value (opt = 5.01)	5.22	5.35	5.08
# of iterations	74.13	94.40	138.33
av. violation	$1.11(10^{-3})$	$1.50(10^{-4})$	$8.49(10^{-4})$
max. violation	$7.30(10^{-3})$	$2.93(10^{-3})$	$6.96(10^{-3})$
Exec. Time (CPLEX: 3.75 sec)	5.35	5.84	8.64

Table 4.12: Averages of the results for randomly generated problems having only equality constraints problems with known Lagrange multipliers.

	ALAG1	ALAG2	ALAG3
LP min value (opt = 2.98)	2.04	2.07	2.07
# of iterations	150.20	135.27	278.13
av. violation	$1.83(10^{-3})$	$4.47(10^{-4})$	$1.01(10^{-3})$
max. violation	$6.82(10^{-3})$	$3.69(10^{-3})$	$5.02(10^{-3})$
Exec. Time (CPLEX: 9.99 sec)	9.46	7.60	17.04

Table 4.13: Averages of the results for standard NETLIB test problems with known Lagrange multipliers.

	ALAG1	ALAG2	ALAG3
LP min value (opt = -12.09)	-4.61	-2.85	-4.38
# of iterations	1523.86	1225.69	594.00
av. violation	$1.26(10^{-3})$	$1.20(10^{-3})$	$1.53(10^{-3})$
max. violation	$1.49(10^{-2})$	$4.25(10^{-2})$	$4.47(10^{-2})$
Exec. time (CPLEX: 0.29 sec)	31.54	26.31	16.84

Table 4.14: Averages of the Tables 8-10 (all problems in Sets 1-3).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
LP min value (opt = -1.37)	-1.78	-1.12	0.84	0.08	-0.18
# of iterations	332.36	426.26	978.36	595.09	479.00
av. violation	$1.60(10^{-3})$	$6.03(10^{-4})$	$1.18(10^{-3})$	$5.41(10^{-4})$	$6.46(10^{-4})$
max. violation	$1.42(10^{-2})$	$4.20(10^{-3})$	$8.59(10^{-3})$	$1.79(10^{-2})$	$5.34(10^{-3})$
Exec. time (CPLEX: 4.68 sec)	16.78	22.35	38.78	20.05	22.42

Table 4.15: Averages of the Tables 11-13 (all problems with known Lagrange multipliers).

	ALAG1	ALAG2	ALAG3
LP min value (opt = -1.37)	0.88	1.52	0.93
# of iterations	582.73	485.12	336.82
av. violation	$1.40(10^{-3})$	$5.99(10^{-4})$	$1.13(10^{-3})$
max. violation	$9.66(10^{-3})$	$1.64(10^{-2})$	$1.89(10^{-2})$
Exec. time (CPLEX: 4.68 sec)	15.45	13.25	14.17

Table 4.16: Results for randomly generated problems used in the experiment on effect of density.

ASL2	31	32	33	34	35
LP min value	-31.15	56.02	-28.13	-2.80	-9.87
# of iterations	382	382	382	382	382
av. violation	$1.72(10^{-5})$	$2.60(10^{-5})$	$2.29(10^{-5})$	$3.17(10^{-5})$	$6.47(10^{-5})$
max. violation	$5.21(10^{-3})$ (constr 9)	$5.31(10^{-3})$ (constr 72)	$5.10(10^{-3})$ (constr 89)	$4.13(10^{-3})$ (constr 65)	$8.43(10^{-3})$ (constr 167)
Exec. time	22.52	45.48	88.21	237.39	44.49
IEL2					
LP min value	-30.45	57.28	-27.55	-1.49	-9.76
# of iterations	382	382	382	382	382
av. violation	$4.98(10^{-5})$	$7.79(10^{-5})$	$5.32(10^{-5})$	$7.22(10^{-5})$	$2.20(10^{-6})$
max. violation	$2.98(10^{-4})$ (constr 393)	$3.46(10^{-4})$ (constr 185)	$2.17(10^{-4})$ (constr 304)	$3.00(10^{-4})$ (constr 106)	$1.01(10^{-5})$ (constr 279)
Exec. time	24.38	56.35	111.28	313.62	54.65
ALAG2					
LP min value	-31.16	56.09	-28.07	-2.50	-9.81
# of iterations	320	320	335	375	270
# of outer loops	1	1	2	1	2
av. violation	$4.07(10^{-4})$	$5.57(10^{-4})$	$5.83(10^{-4})$	$4.42(10^{-4})$	$1.56(10^{-4})$
max. violation	$1.53(10^{-2})$ (constr 36)	$1.56(10^{-2})$ (constr 33)	$1.24(10^{-2})$ (constr 376)	$1.35(10^{-2})$ (constr 17)	$1.26(10^{-2})$ (constr 213)
Exec. time	20.98	49.15	102.32	316.32	41.69

Table 4.16: Results for randomly generated problems used in the experiment on effect of density (continued).

ASL2	36	37	38	39	40
LP min value	7.59	7.05	7.10	-25.75	48.06
# of iterations	382	382	382	382	382
av. violation	$2.15(10^{-4})$	$1.59(10^{-4})$	$9.58(10^{-5})$	$7.24(10^{-5})$	$1.56(10^{-4})$
max. violation	$1.00(10^{-2})$ (constr 241)	$8.92(10^{-3})$ (constr 207)	$7.70(10^{-3})$ (constr 142)	$1.03(10^{-2})$ (constr 36)	$9.12(10^{-3})$ (constr 182)
Exec. time	101.17	216.13	632.03	77.77	206.74
IEL2					
LP min value	7.61	6.97	7.26	-25.44	48.41
# of iterations	382	382	382	382	382
av. violation	$2.20(10^{-6})$	$2.22(10^{-6})$	$2.20(10^{-6})$	$2.41(10^{-6})$	$2.39(10^{-6})$
max. violation	$1.02(10^{-5})$ (constr 322)	$1.01(10^{-5})$ (constr 374)	$1.01(10^{-5})$ (constr 322)	$1.08(10^{-5})$ (constr 259)	$1.03(10^{-5})$ (constr 399)
Exec. time	158.68	343.41	1036.06	96.50	313.24
ALAG2					
LP min value	7.57	6.53	7.20	-26.29	48.00
# of iterations	270	270	270	270	270
# of outer loops	2	2	2	2	2
av. violation	$1.34(10^{-4})$	$1.61(10^{-4})$	$1.44(10^{-4})$	$1.05(10^{-4})$	$1.61(10^{-4})$
max. violation	$1.14(10^{-2})$ (constr 217)	$8.94(10^{-3})$ (constr 231)	$3.68(10^{-3})$ (constr 414)	$1.46(10^{-2})$ (constr 33)	$6.65(10^{-3})$ (constr 36)
Exec. time	116.60	255.62	759.13	72.72	228.66

Table 4.16: Results for randomly generated problems used in the experiment on effect of density (continued).

ASL2	41	42	43	44	45
LP min value	-10.13	3.04	-30.37	-39.42	-0.13
# of iterations	382	382	457	457	457
av. violation	$9.41(10^{-5})$	$9.17(10^{-5})$	$1.22(10^{-4})$	$6.62(10^{-5})$	$8.26(10^{-5})$
max. violation	$9.41(10^{-3})$ (constr 148)	$1.02(10^{-2})$ (constr 198)	$7.43(10^{-3})$ (constr 82)	$8.90(10^{-3})$ (constr 239)	$7.64(10^{-3})$ (constr 378)
Exec. time	485.87	1431.52	188.01	570.90	1350.73
IEL2					
LP min value	-9.70	3.31	-30.03	-39.48	0.85
# of iterations	382	382	457	457	457
av. violation	$2.40(10^{-6})$	$2.38(10^{-6})$	$2.24(10^{-6})$	$2.24(10^{-6})$	$2.23(10^{-6})$
max. violation	$1.03(10^{-5})$ (constr 288)	$1.00(10^{-5})$ (constr 362)	$1.12(10^{-5})$ (constr 439)	$1.03(10^{-5})$ (constr 506)	$1.06(10^{-5})$ (constr 597)
Exec. time	712.71	2093.98	249.91	871.56	1970.84
ALAG2					
LP min value	-9.99	3.26	-30.10	-39.31	0.11
# of iterations	270	275	324	324	324
# of outer loops	2	2	2	2	2
av. violation	$1.70(10^{-4})$	$1.17(10^{-4})$	$2.64(10^{-4})$	$2.45(10^{-4})$	$9.06(10^{-5})$
max. violation	$9.96(10^{-3})$ (constr 210)	$9.49(10^{-3})$ (constr 47)	$7.26(10^{-3})$ (constr 209)	$1.60(10^{-2})$ (constr 206)	$3.52(10^{-3})$ (constr 557)
Exec. time	522.89	1560.21	183.46	631.09	1415.10

Table 4.16: Results for randomly generated problems used in the experiment on effect of density (continued).

ASL2	46	47	48	49	50
LP min value	36.88	-12.10	-8.31	-50.30	-50.12
# of iterations	457	572	572	572	572
av. violation	$8.97(10^{-5})$	$1.78(10^{-4})$	$1.14(10^{-4})$	$1.13(10^{-4})$	$1.51(10^{-4})$
max. violation	$9.56(10^{-3})$ (constr 248)	$1.11(10^{-2})$ (constr 397)	$8.85(10^{-3})$ (constr 415)	$1.05(10^{-2})$ (constr 401)	$9.06(10^{-3})$ (constr 315)
Exec. time	3833.20	460.00	1480.13	3396.70	9629.82
IEL2					
LP min value	37.48	-11.68	-7.51	-49.98	-50.76
# of iterations	457	572	572	572	290
av. violation	$2.29(10^{-6})$	$2.42(10^{-6})$	$2.40(10^{-6})$	$2.38(10^{-6})$	$2.55(10^{-4})$
max. violation	$1.04(10^{-5})$ (constr 506)	$1.06(10^{-5})$ (constr 745)	$1.04(10^{-5})$ (constr 555)	$1.02(10^{-5})$ (constr 578)	$1.26(10^{-3})$ (constr 585)
Exec. time	5607.07	624.33	2248.27	5045.51	7205.62
ALAG2					
LP min value	36.75	-11.83	-8.52	-50.64	-49.79
# of iterations	324	494	404	421	412
# of outer loops	2	2	2	3	2
av. violation	$8.72(10^{-5})$	$1.75(10^{-4})$	$1.62(10^{-4})$	$1.79(10^{-4})$	$1.47(10^{-4})$
max. violation	$8.49(10^{-3})$ (constr 141)	$3.56(10^{-2})$ (constr 47)	$1.51(10^{-2})$ (constr 470)	$2.34(10^{-2})$ (constr 206)	$1.12(10^{-2})$ (constr 214)
Exec. time	4048.17	558.54	1617.78	3828.36	10557.89

Table 4.17: Results for randomly generated problems used in the experiment on effect of low density.

ASL2	51	52	53	54	55
LP min value	24.80	-3.64	45.71	-58.68	92.34
# of iterations	770	770	770	957	957
av. violation	$8.63(10^{-5})$	$1.04(10^{-4})$	$1.23(10^{-4})$	$3.90(10^{-5})$	$5.34(10^{-5})$
max. violation	$9.10(10^{-3})^{(\text{constr } 222)}$	$1.05(10^{-2})^{(\text{constr } 868)}$	$9.32(10^{-3})^{(\text{constr } 518)}$	$1.06(10^{-2})^{(\text{constr } 34)}$	$9.36(10^{-3})^{(\text{constr } 893)}$
Exec. time	302.09	314.95	377.06	676.52	669.37
IEL2					
LP min value	23.43	-4.45	45.86	-57.33	92.31
# of iterations	770	770	770	957	957
av. violation	$1.95(10^{-6})$	$1.96(10^{-6})$	$1.96(10^{-6})$	$2.44(10^{-6})$	$2.41(10^{-6})$
max. violation	$1.57(10^{-5})^{(\text{constr } 1080)}$	$1.04(10^{-5})^{(\text{constr } 1319)}$	$1.03(10^{-5})^{(\text{constr } 1319)}$	$1.28(10^{-5})^{(\text{constr1183})}$	$1.07(10^{-5})^{(\text{constr } 1502)}$
Exec. time	344.33	332.25	472.08	625.49	702.49
ALAG2					
LP min value	22.82	-4.71	45.32	-60.04	91.66
# of iterations	541	541	541	674	674
# of outer loops	2	2	2	2	2
av. violation	$1.58(10^{-4})$	$1.42(10^{-4})$	$1.63(10^{-4})$	$4.50(10^{-4})$	$3.01(10^{-4})$
max. violation	$1.87(10^{-2})^{(\text{constr } 429)}$	$2.17(10^{-2})^{(\text{constr } 105)}$	$2.57(10^{-2})^{(\text{constr } 441)}$	$6.61(10^{-2})^{(\text{constr } 444)}$	$4.21(10^{-2})^{(\text{constr } 240)}$
Exec. time	270.40	273.70	349.54	521.02	578.09

Table 4.17: Results for randomly generated problems used in the experiment on effect of low density (continued).

ASL2	56	57	58	59	60
LP min value	24.72	-3.84	-24.06	-46.44	-64.20
# of iterations	957	1152	1152	1152	1541
av. violation	$8.17(10^{-5})$	$3.61(10^{-5})$	$7.11(10^{-5})$	$7.95(10^{-5})$	$4.68(10^{-5})$
max. violation	$1.02(10^{-2})^{(\text{constr } 50)}$	$9.82(10^{-3})^{(\text{constr } 1075)}$	$1.03(10^{-2})^{(\text{constr } 826)}$)	$1.26(10^{-2})^{(\text{constr } 779)}$	$8.11(10^{-3})^{(\text{constr } 550)}$
Exec. time	765.93	1256.03	1340.57	1749.43	1650.35
IEL2					
LP min value	25.20	-2.11	-21.99	-43.87	-63.89
# of iterations	957	1152	1152	1152	1541
av. violation	$2.41(10^{-6})$	$2.60(10^{-6})$	$2.54(10^{-6})$	$2.57(10^{-6})$	$2.47(10^{-6})$
max. violation	$1.10(10^{-5})^{(\text{constr } 1539)}$	$2.25(10^{-5})^{(\text{constr } 2014)}$	$1.25(10^{-5})^{(\text{constr } 2041)}$	$1.25(10^{-5})^{(\text{constr } 2156)}$	$1.01(10^{-5})^{(\text{constr } 1006)}$
Exec. time	866.01	1403.68	1596.30	1980.28	1682.43
ALAG2					
LP min value	24.78	-3.68	-23.69	-45.54	-64.76
# of iterations	674	811	811	811	693
# of outer loops	2	2	2	2	2
av. violation	$2.54(10^{-4})$	$2.87(10^{-4})$	$2.94(10^{-4})$	$2.91(10^{-4})$	$1.68(10^{-4})$
max. violation	$3.17(10^{-2})^{(\text{constr } 390)}$	$3.81(10^{-2})^{(\text{constr } 33)}$	$4.90(10^{-2})^{(\text{constr } 311)}$	$3.15(10^{-2})^{(\text{constr } 892)}$	$3.12(10^{-2})^{(\text{constr } 318)}$
Exec. time	700.25	1016.56	1155.86	1443.11	865.79

Table 4.17: Results for randomly generated problems used in the experiment on effect of low density (continued).

ASL2	61	62	63	64	65
LP min value	-69.21	26.56	-25.99	-12.41	34.12
# of iterations	1541	1541	1925	941	1217
av. violation	$9.36(10^{-5})$	$9.08(10^{-5})$	$8.66(10^{-5})$	$3.07(10^{-5})$	$1.05(10^{-4})$
max. violation	$1.07(10^{-2})$ (constr 691)	$9.72(10^{-3})$ (constr 562)	$9.48(10^{-3})$ (constr 861)	$7.99(10^{-3})$ (constr 909)	$1.02(10^{-2})$ (constr 557)
Exec. time	1859.34	2322.96	2053.83	1145.53	1860.93
IEL2					
LP min value	-69.10	26.64	-25.99	-12.40	34.96
# of iterations	1541	1541	1925	1925	957
av. violation	$2.47(10^{-6})$	$2.47(10^{-6})$	$2.09(10^{-6})$	$2.09(10^{-6})$	$2.09(10^{-6})$
max. violation	$9.99(10^{-6})$ (constr 1533)	$1.00(10^{-5})$ (constr 1482)	$9.97(10^{-6})$ (constr 1084)	$9.97(10^{-6})$ (constr 1084)	$1.00(10^{-5})$ (constr 1311)
Exec. time	1972.05	2636.42	2427.92	2879.63	1944.48
ALAG2					
LP min value	-69.30	26.45	-26.04	-12.40	34.15
# of iterations	838	660	649	577	627
# of outer loops	2	2	1	1	1
av. violation	$7.25(10^{-5})$	$7.88(10^{-5})$	$9.04(10^{-5})$	$5.21(10^{-5})$	$9.60(10^{-5})$
max. violation	$1.39(10^{-2})$ (constr 67)	$5.92(10^{-3})$ (constr 393)	$2.69(10^{-2})$ (constr 334)	$5.33(10^{-3})$ (constr 217)	$3.02(10^{-2})$ (constr 493)
Exec. time	1217.31	1257.63	829.54	906.48	1400.82

Table 18: Averages of the results for high-density test problems.

	ASL2	IEL2	ALAG2
LP min value (opt = -7.04)	-6.64	-6.23	-6.63
# of iterations	435.0	420.9	327.1
av. violation	$9.82(10^{-5})$	$2.71(10^{-5})$	$2.24(10^{-4})$
max. violation	$8.34(10^{-3})$	$1.29(10^{-4})$	$1.27(10^{-2})$
Exec. time (CPLEX: 203.17 sec)	1224.94	1456.90	1344.33

Table 19: Averages of the results for low-density test problems.

	ASL2	IEL2	ALAG2
LP min value (opt = -4.95)	-4.01	-3.52	-4.33
# of iterations	1156.2	1204.5	674.8
av. violation	$7.52(10^{-5})$	$2.30(10^{-6})$	$1.93(10^{-4})$
max. violation	$9.87(10^{-3})$	$1.19(10^{-5})$	$2.92(10^{-2})$
Exec. time (CPLEX: 2030.6 sec)	1222.99	1457.72	852.41

Table 20: Averages of the Tables 18-19 (all problems in Sets 4-5).

	ASL2	IEL2	ALAG2
LP min value (opt = -6.15)	-5.52	-5.07	-5.64
# of iterations	744.1	756.7	476.1
av. violation	$8.83(10^{-5})$	$1.65(10^{-5})$	$2.11(10^{-4})$
max. violation	$9.00(10^{-3})$	$7.87(10^{-5})$	$1.98(10^{-2})$
Exec. time (CPLEX: 986.34 sec)	1224.11	1457.25	1133.51

4.7 Statistical Analysis of Results

Average rank tests based on the computational results obtained are performed using two different statistics in order to determine the relative effectiveness of the algorithms. We first perform these tests separately on each class of problems considered in this thesis, namely on generally structured randomly generated LP problems, LP problems with equality constraints only, the set of NETLIB test problems, and on high-density, and low-density test problems. These sets of problems will be referred to as Set 1, Set 2, Set 3, Set 4, and Set 5, respectively. Then, in order to examine the general behavior of the algorithms, the first three and the last two sets of problems are aggregated and the tests are carried out again on the entire set.

The first of the two statistics used in these tests is given by

$$\beta(x^*, x^0) = \frac{F(x^0) - F(x^*)}{\text{execution time}} \tag{4.1}$$

where $F(x^0)$ and $F(x^*)$ are the penalized objective functions evaluated at the initial and final solutions, respectively. The function, $F(x)$, is given by

$$F(x) = cx + \mu\left[\sum_{i \in I_1} \max\{0, b_i - A_i x\} + \sum_{i \in I_2} |b_i - A_i x|\right]. \tag{4.2}$$

That is, the l_1 penalty function is used as a merit function for all the methods tested. The penalty parameter, μ, is taken to be 10^6 for the purpose of scaling the infeasibility terms. This statistic is designed in order to evaluate the average speed of convergence (improvement rate) of the algorithms with respect to the common merit function $F(x)$. Note that $F(x^0)$ is the same for all methods for a given problem since the initial solution is uniformly selected as the zero vector.

The second statistic is the LP problem objective function penalized with an l_1 type of penalty term that permits some positive tolerance ε in constraint violations before invoking a penalty. This function is given by

$$F_\varepsilon(x) = cx + \mu\alpha(x) \tag{4.3}$$

where

$$\alpha(x) = \sum_{i \in I_1} \max\{0, (b_i - A_i x - \varepsilon)\} + \sum_{i \in I_2} \max\{0, (b_i - A_i x - \varepsilon), (-b_i + A_i x - \varepsilon)\}. \tag{4.4}$$

The tolerance ε reflects the degree of violation permitted for each constraint before a penalty is imposed. Hence, this statistic assesses the quality of the solutions attained by the various algorithms with respect to both the final objective function value and the degree of feasibility while tolerating some violations as reflected by the shifted l_1 type of penalty term. For the purpose of distinguishing the effect of different permissible infeasibility levels, four different

values of ε were adopted. The different values of ε utilized are 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}. Again, the penalty parameter, μ, is taken to be 10^6.

The results of the statistical analysis computations are given in Tables 4.24 – 4.58. Each number in these tables compares the performance of the algorithms in the corresponding row and column as explained in detail below. A negative number indicates that the row algorithm is better, while a positive number indicates that the column algorithm performed better. The magnitude of the entry indicates the confidence level for the corresponding comparison, as explained below.

To illustrate the calculations of confidence levels reported in these tables, let us consider $\beta(x^*, x^0)$ defined by (4.1) and (4.2). Assume that the methods ASL2 and ALAG2 are to be compared over the set of randomly generated generally structured linear programming problems. The statistics collected from these two runs are given in Table 4.21.

Table 4.21: β statistics for the ASL2 and ALAG2 runs.

Problem	ASL2	ALAG2
1	$2.01(10^8)$	$2.92(10^8)$
2	$3.47(10^8)$	$7.77(10^8)$
3	$1.15(10^8)$	$1.09(10^8)$
4	$6.93(10^7)$	$8.73(10^7)$
5	$4.41(10^7)$	$5.01(10^7)$
6	$4.03(10^7)$	$7.65(10^7)$
7	$8.00(10^7)$	$8.71(10^7)$
8	$3.68(10^7)$	$4.60(10^7)$
9	$4.41(10^7)$	$5.95(10^7)$
10	$9.54(10^7)$	$1.22(10^8)$
11	$1.75(10^7)$	$1.59(10^7)$
12	$1.79(10^7)$	$2.39(10^7)$
13	$1.56(10^7)$	$2.13(10^7)$
14	$5.41(10^6)$	$1.49(10^7)$
15	$9.69(10^6)$	$2.26(10^7)$

The calculations for performing an average rank test for this set of data are presented in Table 4.22.

Table 4.22: Average rank test calculations.

Problem	ASL2 (1)	ALAG2 (2)	Difference (3) = (1) – (2)	|difference| (4) = |(1) – (2)|	rank (5)	Signed rank (6) = (5)×(3)/(4)
1	$2.01(10^8)$	$2.92(10^8)$	$-9.13(10^7)$	$9.13(10^7)$	14	-14
2	$3.47(10^8)$	$7.77(10^8)$	$-4.31(10^8)$	$4.31(10^8)$	15	-15
3	$1.15(10^8)$	$1.09(10^8)$	$6.60(10^6)$	$6.60(10^6)$	5	5
4	$6.93(10^7)$	$8.73(10^7)$	$-1.80(10^7)$	$1.80(10^7)$	11	-11
5	$4.41(10^7)$	$5.01(10^7)$	$-5.95(10^6)$	$5.95(10^6)$	3	-3
6	$4.03(10^7)$	$7.65(10^7)$	$-3.62(10^7)$	$3.62(10^7)$	13	-13
7	$8.00(10^7)$	$8.71(10^7)$	$-7.18(10^6)$	$7.18(10^6)$	6	-6
8	$3.68(10^7)$	$4.60(10^7)$	$-9.18(10^6)$	$9.18(10^6)$	7	-7
9	$4.41(10^7)$	$5.95(10^7)$	$-1.54(10^7)$	$1.54(10^7)$	10	-10
10	$9.54(10^7)$	$1.22(10^8)$	$-2.69(10^7)$	$2.69(10^7)$	12	-12
11	$1.75(10^7)$	$1.59(10^7)$	$1.61(10^6)$	$1.61(10^6)$	1	1
12	$1.79(10^7)$	$2.39(10^7)$	$-6.04(10^6)$	$6.04(10^6)$	4	-4
13	$1.56(10^7)$	$2.13(10^7)$	$-5.71(10^6)$	$5.71(10^6)$	2	-2
14	$5.41(10^6)$	$1.49(10^7)$	$-9.53(10^6)$	$9.53(10^6)$	8	-8
15	$9.69(10^6)$	$2.26(10^7)$	$-1.29(10^7)$	$1.29(10^7)$	9	-9

Suppose that a random sample of N differences is selected from a symmetrical population of differences with population median difference, $\eta_D = 0$. Let T be the summation of the values computed as in the column denoted by (6). Then the sampling distribution of T has mean and variance:

$$E\{T\} = 0, \qquad \sigma^2(T) = \frac{N(N+1)(2N+1)}{6}.$$

Note that when N is sufficiently large, the sampling distribution of T is approximately normal. The Wilcoxon signed rank test concerning η_D is based on the standardized test statistic:

$$t^* = \frac{T}{\sigma(T)}.$$

The values of T, $\sigma^2(T)$, and t^* for the data in Table 4.22 are given in Table 4.23. Using statistical tables or statistical software, it follows that $P(-|t^*| \le t \le |t^*|) = 0.9922$, where t is a standard normal random variable with mean 0 and variance 1. Therefore, it is concluded with 99.22% confidence level that the median difference is not zero. Specifically, it is less than zero in this instance, meaning that the median of the statistics collected from the ASL2 run is less than that of the ALAG2 run. For this statistic, higher values are preferred. Hence, ALAG2 performs better than ASL2 with respect to the statistic $\beta(x^*, x^0)$.

Table 4.23: Results of the average rank test performed on ASL2 and ALAG2

T	-108
N	15
$\sigma^2(T)$	35.2136
t^*	-3.0670
$\|t^*\|$	3.0670
$P(.)$	0.9922

The tables 4.24 – 4.58 record signed confidence levels P, where $|P|$ is the confidence level value obtained as above while comparing the performance of the row algorithm versus the column algorithm. If the value of P is negative, then the row method is better than the column method at a -100P% confidence level. Otherwise, the column method is better than the row method with at a 100P% confidence. This type of representation of results is adopted for the sake of convenience.

For example, examining Table 4.24, it is observed that IEL2 has a higher $\beta(x^*, x^0)$ value than ALAG1, and a lower $\beta(x^*, x^0)$ value than ALAG2 with 62.21% and 99.61% confidence levels, respectively.

Tables 4.24 – 4.30 and 4.31 – 4.58 give results for the statistics $\beta(x^*, x^0)$ (defined by (4.1) and (4.2)) and $F_\varepsilon(x)$ (defined by (4.3) and (4.4)), respectively. The results confirm the previous discussion in Sections 4.2 – 4.6. In Tables 4.59 – 4.68, the rankings of the algorithms tested are given for each set of problems as well as the aggregate set of Set 1-3 and the aggregate set of Sets 4 and 5.

For the generally-structured randomly-generated linear programs the convergence rate of ALAG2 is the highest among the algorithms tested. ALAG3 is the second best, ASL2 is the third, with IEL2 fourth and ALAG1 is fifth best. The gaps between the algorithms are quite significant except for that between ALAG2 and ALAG3, and between IEL2 and ALAG1, which have probabilities of 0.7262 and 0.6221, respectively.

For randomly generated problems having only equality constraints, ASL2 has the highest convergence rate with ALAG2 being second best. The third highest rate is for ALAG3, followed by IEL2 and then by ALAG1. Again, the gaps are significant except for that between ALAG2 and ALAG3, and between ALAG3 and IEL2, where the corresponding probabilities are 0.6504 and 0.8070, respectively.

For the set of NETLIB test problems, the ordering of the convergence rates is somewhat different from the previous cases. ASL2 appears best. IEL2 takes the second place. ALAG3 is third, and ALAG2 falls to the fourth place. Again, ALAG1 is the slowest. The gaps in the ranking are of the same magnitude of significance as in the previous cases.

Aggregating the first three sets of the test problems had some effect on the certainties of the ranking. The ranking of the convergence rate turns out to be the same as that for the randomly generated problems having only equality constraints.

For the high-density test problems, ASL2 has the highest convergence rate with ALAG2 being second best. The third highest rate is for IEL2. The lowest significance level is between ASL2 and ALAG2, where the probability is 0.8839.

On the other hand, for the low-density test problems, ALAG2 has the highest convergence rate. The second highest rate is for ASL2, followed by IEL2. All the significance levels for the ranking are higher than 0.98.

The aggregation of Set 4 and Set 5 had some effect on the certainties of the ranking. However, the ranking of the convergence rate turns out to be the same as that for the low-density test problems.

The second statistic, aimed at assessing the quality of solutions, is computed for different levels of infeasibility by varying ε. Tables 4.31 – 4.37, 4.38 – 4.44, 4.45 – 4.51, and 4.52 – 4.58 display the results for $\varepsilon = 10^{-3}$, 10^{-4}, 10^{-5}, and 10^{-6}, respectively. However, changing ε had very little effect, if any, on the ranking of the methods. The results are slightly different for $\varepsilon = 10^{-3}$, but for the remaining levels of infeasibility, the ranking is exactly the same. Only slight changes in confidence levels were observed.

For all the instances of ε, the results for the generally-structured randomly-generated linear programming problems were the same. The best quality solution was attained, in order, by IEL2, ALAG2, ASL2, ALAG3, and last by ALAG1.

The algorithm rankings for the set of randomly generated test problems having only equality constraints was somewhat affected by ε. For $\varepsilon = 10^{-3}$, the ranking order was IEL2, ALAG3, ALAG2, ALAG1, and ASL2. For the other instances of ε, this order turned out to be IEL2, ALAG2, ALAG3, ALAG1, and ASL2.

Again, for the NETLIB test problems, varying ε had no effect on the ranking with respect to the quality of solutions. The augmented Lagrangian algorithms shared the first three places, in the order ALAG2, ALAG1, and, ALAG3 followed by IEL2 and lastly by ASL2.

In the aggregate case, where Set 1, Set 2 and Set 3 are combined, the ranking was again the same as for the randomly generated problems having only equality constraints. That is, the algorithms were ordered as IEL2, ALAG2, ALAG3, ALAG1, and ASL2.

For the high-density and low-density test problems as well as for their aggregate, the ranking order was IEL2, ASL2, and ALAG2 for most of the cases. Only for the high-density problems with $\varepsilon = 10^{-3}$, the ranking was IEL2, ALAG2, and ASL2, having a low significance level between ALAG2 and ASL2.

Table 4.24: Statistical analysis results of β for problems in Set 1.

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	-0.9961	-0.9013	0.9922	0.9804
IEL2	0.9961	X	-0.6221	0.9961	0.9956
ALAG1	0.9013	0.6221	X	0.9804	0.9901
ALAG2	-0.9922	-0.9961	-0.9804	X	-0.7262
ALAG3	-0.9804	-0.9956	-0.9901	0.7262	X

Table 4.25: Statistical analysis results of β for problems in Set 2.

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	-0.9961	-0.9804	-0.9889	-0.9108
IEL2	0.9961	X	-0.9108	0.9844	0.8070
ALAG1	0.9804	0.9108	X	0.9781	0.9693
ALAG2	0.9889	-0.9844	-0.9781	X	-0.6504
ALAG3	0.9108	-0.8070	-0.9693	0.6504	X

Table 4.26: Statistical analysis results of β for problems in Set 3.

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	-0.9663	-0.9838	-0.9393	-0.9460
IEL2	0.9663	X	-0.9838	-0.9519	-0.8816
ALAG1	0.9838	0.9838	X	0.9838	0.9940
ALAG2	0.9393	0.9519	-0.9838	X	0.6219
ALAG3	0.9460	0.8816	-0.9940	-0.6219	X

Table 4.27: Statistical analysis results of β for the collection of test problems in Sets 1-3.

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	-1.0000	-0.9999	-0.7816	-0.8722
IEL2	1.0000	X	-0.9973	0.9994	0.9882
ALAG1	0.9999	0.9973	X	1.0000	0.9999
ALAG2	0.7816	-0.9994	-1.0000	X	-0.8777
ALAG3	0.8722	-0.9882	-0.9999	0.8777	X

Table 4.28: Statistical analysis results of β for problems in Set 4.

	ASL2	IEL2	ALAG2
ASL2	X	-0.9991	-0.8839
IEL2	0.9991	X	0.9771
ALAG2	0.8839	-0.9771	X

Table 4.29: Statistical analysis results of β for problems in Set 5.

	ASL2	IEL2	ALAG2
ASL2	X	-0.9856	0.9947
IEL2	0.9856	X	0.9947
ALAG2	-0.9947	-0.9947	X

Table 4.30: Statistical analysis results of β for the collection of test problems in Sets 4-5

	ASL2	IEL2	ALAG2
ASL2	X	-1.0000	0.8370
IEL2	1.0000	X	1.0000
ALAG2	-0.8370	-1.0000	X

Table 4.31: Statistical analysis results of F_ε for problems in Set 1. ($\varepsilon = 10^{-3}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9922	-0.9876	0.7889	-0.2197
IEL2	-0.9922	X	-0.9961	-0.9961	-0.9961
ALAG1	0.9876	0.9961	X	0.9961	0.9861
ALAG2	-0.7889	0.9961	-0.9961	X	-0.9525
ALAG3	0.2197	0.9961	-0.9861	0.9525	X

Table 4.32: Statistical analysis results of F_ε for problems in Set 2 ($\varepsilon = 10^{-3}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9618	0.8797	0.9725	0.9861
IEL2	-0.9618	X	-0.9754	-0.8396	-0.2620
ALAG1	-0.8797	0.9754	X	0.9961	0.9945
ALAG2	-0.9725	0.8396	-0.9961	X	0.3035
ALAG3	-0.9861	0.2620	-0.9945	-0.3035	X

Table 4.33: Statistical analysis results of F_ε for problems in Set 3 ($\varepsilon = 10^{-3}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.6504	0.9912	0.9961	0.9956
IEL2	-0.6504	X	0.8239	0.9346	0.9108
ALAG1	-0.9912	-0.8239	X	0.0445	-0.5611
ALAG2	-0.9961	-0.9346	-0.0445	X	-0.8070
ALAG3	-0.9956	-0.9108	0.5611	0.8070	X

Table 4.34: Statistical analysis results of F_ε for the collection of test problems in Sets 1-3 ($\varepsilon = 10^{-3}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9963	0.9548	1.0000	0.9998
IEL2	-0.9963	X	-0.9525	-0.6875	-0.2423
ALAG1	-0.9548	0.9525	X	0.9996	0.9943
ALAG2	-1.0000	0.6875	-0.9996	X	-0.9140
ALAG3	-0.9998	0.2423	-0.9943	0.9140	X

Table 4.35: Statistical analysis results of F_ε for problems in Set 4 ($\varepsilon = 10^{-3}$).

	ASL2	IEL2	ALAG2
ASL2	X	0.9992	0.0588
IEL2	-0.9992	X	-0.9992
ALAG2	-0.0588	0.9992	X

Table 4.36: Statistical analysis results of F_ε for problems in Set 5 ($\varepsilon = 10^{-3}$).

	ASL2	IEL2	ALAG2
ASL2	X	0.9961	-0.9655
IEL2	-0.9961	X	-0.9961
ALAG2	0.9655	0.9961	X

Table 4.37: Statistical analysis results of F_ε for the collection of test problems in Sets 4-5 ($\varepsilon = 10^{-3}$).

	ASL2	IEL2	ALAG2
ASL2	X	1.0000	-0.4516
IEL2	-1.0000	X	-1.0000
ALAG2	0.4516	1.0000	X

Table 4.38: Statistical analysis results of F_ε for problems in Set 1 ($\varepsilon = 10^{-4}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9938	-0.9922	0.7485	-0.8910
IEL2	-0.9938	X	-0.9961	-0.9961	-0.9961
ALAG1	0.9922	0.9961	X	0.9961	0.9922
ALAG2	-0.7485	0.9961	-0.9961	X	-0.9754
ALAG3	0.8910	0.9961	-0.9922	0.9754	X

Table 4.39: Statistical analysis results of F_ε for problems in Set 2 ($\varepsilon = 10^{-4}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9693	0.8674	0.9754	0.9861
IEL2	-0.9693	X	-0.9815	-0.9108	-0.3439
ALAG1	-0.8674	0.9815	X	0.9959	0.9959
ALAG2	-0.9754	0.9108	-0.9959	X	-0.2620
ALAG3	-0.9861	0.3439	-0.9959	0.2620	X

Table 4.40: Statistical analysis results of F_ε for problems in Set 5 ($\varepsilon = 10^{-4}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.6504	0.9901	0.9961	0.9956
IEL2	-0.6504	X	0.8239	0.9411	0.9274
ALAG1	-0.9901	-0.8239	X	0.2197	-0.4585
ALAG2	-0.9961	-0.9411	-0.2197	X	-0.8070
ALAG3	-0.9956	-0.9274	0.4585	0.8070	X

Table 4.41: Statistical analysis results of F_ε for the collection of test problems in Sets 1-3 ($\varepsilon = 10^{-4}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9973	0.8750	1.0000	0.9993
IEL2	-0.9973	X	-0.9730	-0.7031	-0.5089
ALAG1	-0.8750	0.9730	X	0.9998	0.9981
ALAG2	-1.0000	0.7031	-0.9998	X	-0.9788
ALAG3	-0.9993	0.5089	-0.9981	0.9788	X

Table 4.42: Statistical analysis results of F_ε for problems in Set 4 ($\varepsilon = 10^{-4}$).

	ASL2	IEL2	ALAG2
ASL2	X	0.9981	-0.8286
IEL2	-0.9981	X	-0.9978
ALAG2	0.8286	0.9978	X

Table 4.43: Statistical analysis results of F_ε for problems in Set 5 ($\varepsilon = 10^{-4}$).

	ASL2	IEL2	ALAG2
ASL2	X	0.9961	-0.7889
IEL2	-0.9961	X	-0.9961
ALAG2	0.7889	0.9961	X

Table 4.44: Statistical analysis results of F_ε for the collection of test problems in Sets 4-5 ($\varepsilon = 10^{-4}$).

	ASL2	IEL2	ALAG2
ASL2	X	1.0000	-0.9343
IEL2	-1.0000	X	-1.0000
ALAG2	0.9343	1.0000	X

Table 4.45: Statistical analysis results of F_ε for problems in Set 1 ($\varepsilon = 10^{-5}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9912	-0.9861	0.8239	-0.9471
IEL2	-0.9912	X	-0.9961	-0.9961	-0.9961
ALAG1	0.9861	0.9961	X	0.9961	0.9922
ALAG2	-0.8239	0.9961	-0.9961	X	-0.9781
ALAG3	0.9471	0.9961	-0.9922	0.9781	X

Table 4.46: Statistical analysis results of F_ε for problems in Set 2 ($\varepsilon = 10^{-5}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9693	0.8674	0.9754	0.9861
IEL2	-0.9693	X	-0.9876	-0.9108	-0.5611
ALAG1	-0.8674	0.9876	X	0.9959	0.9959
ALAG2	-0.9754	0.9108	-0.9959	X	-0.2828
ALAG3	-0.9861	0.5611	-0.9959	0.2828	X

Table 4.47: Statistical analysis results of F_ε for problems in Set 3 ($\varepsilon = 10^{-5}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.6772	0.9901	0.9961	0.9956
IEL2	-0.6772	X	0.8239	0.9411	0.9274
ALAG1	-0.9901	-0.8239	X	0.2197	-0.4585
ALAG2	-0.9961	-0.9411	-0.2197	X	-0.8070
ALAG3	-0.9956	-0.9274	0.4585	0.8070	X

Table 4.48: Statistical analysis results of F_ε for the collection of test problems in Sets 1-3 ($\varepsilon = 10^{-5}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9972	0.8523	0.9999	0.9989
IEL2	-0.9972	X	-0.9770	-0.7279	-0.5825
ALAG1	-0.8523	0.9770	X	0.9998	0.9980
ALAG2	-0.9999	0.7279	-0.9998	X	-0.9804
ALAG3	-0.9989	0.5825	-0.9980	0.9804	X

Table 4.49: Statistical analysis results of F_ε for problems in Set 4 ($\varepsilon = 10^{-5}$).

	ASL2	IEL2	ALAG2
ASL2	X	0.9914	-0.9921
IEL2	-0.9914	X	-0.9978
ALAG2	0.9921	0.9978	X

Table 4.50: Statistical analysis results of F_ε for problems in Set 5 ($\varepsilon = 10^{-5}$).

	ASL2	IEL2	ALAG2
ASL2	X	0.9961	-0.9274
IEL2	-0.9961	X	-0.9961
ALAG2	0.9274	0.9961	X

Table 4.51: Statistical analysis results of F_ε for the collection of test problems in Sets 4-5 ($\varepsilon = 10^{-5}$).

	ASL2	IEL2	ALAG2
ASL2	X	1.0000	-0.9982
IEL2	-1.0000	X	-1.0000
ALAG2	0.9982	1.0000	X

Table 4.52: Statistical analysis results of F_ε for problems in Set 1 ($\varepsilon = 10^{-6}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9912	-0.9930	0.5611	-0.9346
IEL2	-0.9912	X	-0.9961	-0.9961	-0.9961
ALAG1	0.9930	0.9961	X	0.9961	0.9922
ALAG2	-0.5611	0.9961	-0.9961	X	-0.9781
ALAG3	0.9346	0.9961	-0.9922	0.9781	X

Table 4.53: Statistical analysis results of F_ε for problems in Set 2 ($\varepsilon = 10^{-6}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9693	0.8674	0.9754	0.9844
IEL2	-0.9693	X	-0.9861	-0.9108	-0.5611
ALAG1	-0.8674	0.9861	X	0.9959	0.9959
ALAG2	-0.9754	0.9108	-0.9959	X	-0.2828
ALAG3	-0.9844	0.5611	-0.9959	0.2828	X

Table 4.54: Statistical analysis results of F_ε for problems in Set 3 ($\varepsilon = 10^{-6}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.6772	0.9901	0.9961	0.9956
IEL2	-0.6772	X	0.8239	0.9525	0.9274
ALAG1	-0.9901	-0.8239	X	0.2197	-0.3035
ALAG2	-0.9961	-0.9525	-0.2197	X	-0.8070
ALAG3	-0.9956	-0.9274	0.3035	0.8070	X

Table 4.55: Statistical analysis results of F_ε for the collection of test problems in Sets 1-3 ($\varepsilon = 10^{-6}$).

	ASL2	IEL2	ALAG1	ALAG2	ALAG3
ASL2	X	0.9972	0.8523	0.9999	0.9988
IEL2	-0.9972	X	-0.9770	-0.7182	-0.5825
ALAG1	-0.8523	0.9770	X	0.9998	0.9980
ALAG2	-0.9999	0.7182	-0.9998	X	-0.9804
ALAG3	-0.9988	0.5825	-0.9980	0.9804	X

Table 4.56: Statistical analysis results of F_ε for problems in Set 4 ($\varepsilon = 10^{-6}$).

	ASL2	IEL2	ALAG2
ASL2	X	0.9914	-0.9948
IEL2	-0.9914	X	-0.9978
ALAG2	0.9948	0.9978	X

Table 4.57: Statistical analysis results of F_ε for problems in Set 5 ($\varepsilon = 10^{-6}$).

	ASL2	IEL2	ALAG2
ASL2	X	0.9961	-0.9346
IEL2	-0.9961	X	-0.9961
ALAG2	0.9346	0.9961	X

Table 4.58: Statistical analysis results of F_ε for the collection of test problems in Sets 4-5 ($\varepsilon = 10^{-6}$).

	ASL2	IEL2	ALAG2
ASL2	X	1.0000	-0.9989
IEL2	-1.0000	X	-1.0000
ALAG2	0.9989	1.0000	X

Table 4.59: Results of the average rank test performed on $\beta(x^*, x^0)$.

	BEST	→	…	→	…	→	…	→	WORST
Set 1	ALAG2	→	ALAG3	→	ASL2	→	IEL2	→	ALAG1
Set 2	ASL2	→	ALAG2	→	ALAG3	→	IEL2	→	ALAG1
Set 3	ASL2	→	IEL2	→	ALAG3	→	ALAG2	→	ALAG1
Aggregate	ASL2	→	ALAG2	→	ALAG3	→	IEL2	→	ALAG1

Table 4.60: Results of the average rank test performed on $F_\varepsilon(x)$ ($\varepsilon = 10^{-3}$).

	BEST	→	…	→	…	→	…	→	WORST
Set 1	IEL2	→	ALAG2	→	ASL2	→	ALAG3	→	ALAG1
Set 2	IEL2	→	ALAG2	→	ALAG3	→	ALAG1	→	ASL2
Set 3	ALAG2	→	ALAG1	→	ALAG3	→	IEL2	→	ASL2
Aggregate	IEL2	→	ALAG2	→	ALAG3	→	ALAG1	→	ASL2

Table 4.61: Results of the average rank test performed on $F_\varepsilon(x)$ ($\varepsilon = 10^{-4}$).

	BEST	→	…	→	…	→	…	→	WORST
Set 1	IEL2	→	ALAG2	→	ASL2	→	ALAG3	→	ALAG1
Set 2	IEL2	→	ALAG2	→	ALAG3	→	ALAG1	→	ASL2
Set 3	ALAG2	→	ALAG1	→	ALAG3	→	IEL2	→	ASL2
Aggregate	IEL2	→	ALAG2	→	ALAG3	→	ALAG1	→	ASL2

Table 4.62: Results of the average rank test performed on $F_\varepsilon(x)$ ($\varepsilon = 10^{-5}$).

	BEST	→	…	→	…	→	…	→	WORST
Set 1	IEL2	→	ALAG2	→	ASL2	→	ALAG3	→	ALAG1
Set 2	IEL2	→	ALAG2	→	ALAG3	→	ALAG1	→	ASL2
Set 3	ALAG2	→	ALAG1	→	ALAG3	→	IEL2	→	ASL2
Aggregate	IEL2	→	ALAG2	→	ALAG3	→	ALAG1	→	ASL2

Table 4.63: Results of the average rank test performed on $F_\varepsilon(x)$ ($\varepsilon = 10^{-6}$).

	BEST	→	…	→	…	→	…	→	WORST
Set 1	IEL2	→	ALAG2	→	ASL2	→	ALAG3	→	ALAG1
Set 2	IEL2	→	ALAG2	→	ALAG3	→	ALAG1	→	ASL2
Set 3	ALAG2	→	ALAG1	→	ALAG3	→	IEL2	→	ASL2
Aggregate	IEL2	→	ALAG2	→	ALAG3	→	ALAG1	→	ASL2

Table 5.64: Results of the average rank test performed on $\beta(x^*, x^0)$.

	BEST	→	…	→	WORST
Set 4	ASL2	→	ALAG2	→	IEL2
Set 5	ALAG2	→	ASL2	→	IEL2
Aggregate	ALAG2	→	ASL2	→	IEL2

Table 5.65: Results of the average rank test performed on $F_\varepsilon(x)$ ($\varepsilon = 10^{-3}$).

	BEST	→	…	→	WORST
Set 4	IEL2	→	ALAG2	→	ASL2
Set 5	IEL2	→	ASL2	→	ALAG2
Aggregate	IEL2	→	ASL2	→	ALAG2

Table 5.66: Results of the average rank test performed on $F_\varepsilon(x)$ ($\varepsilon = 10^{-4}$).

	BEST	$\rightarrow$	…	$\rightarrow$	WORST
Set 4	IEL2	$\rightarrow$	ASL2	$\rightarrow$	ALAG2
Set 5	IEL2	$\rightarrow$	ASL2	$\rightarrow$	ALAG2
Aggregate	IEL2	$\rightarrow$	ASL2	$\rightarrow$	ALAG2

Table 5.67: Results of the average rank test performed on $F_\varepsilon(x)$ ($\varepsilon = 10^{-5}$).

	BEST	$\rightarrow$	…	$\rightarrow$	WORST
Set 4	IEL2	$\rightarrow$	ASL2	$\rightarrow$	ALAG2
Set 5	IEL2	$\rightarrow$	ASL2	$\rightarrow$	ALAG2
Aggregate	IEL2	$\rightarrow$	ASL2	$\rightarrow$	ALAG2

Table 5.68: Results of the average rank test performed on $F_\varepsilon(x)$ ($\varepsilon = 10^{-6}$).

	BEST	$\rightarrow$	…	$\rightarrow$	WORST
Set 4	IEL2	$\rightarrow$	ASL2	$\rightarrow$	ALAG2
Set 5	IEL2	$\rightarrow$	ASL2	$\rightarrow$	ALAG2
Aggregate	IEL2	$\rightarrow$	ASL2	$\rightarrow$	ALAG2

Chapter 5

Summary and Conclusions

In this research effort, three exterior penalty function approaches for solving linear programming problems have been investigated: an active set penalty approach (ASL2), an inequality-equality based penalty approach (IEL2), and some variants of an augmented Lagrangian approach (ALAG). The motivation was to study their performance in comparison with each other, using a sophisticated simplex implementation of CPLEX by way of benchmarking the results.

These algorithms have the advantage that they can be initialized at arbitrary starting solutions, although they would naturally benefit from having an advanced start solution. In addition, they possess a great deal of flexibility in designing a particular variant of the procedure. However, this flexibility also implies that the performance of the algorithms can vary greatly with different values of parameters, and this usually entails an appreciable degree of fine-tuning.

In our computational experiments, for randomly generated problems having equality and inequality constraints, ALAG2 performed better than the other approaches in terms of speed and final objective function values (see Table 4.8). In the presence of only equality constraints, ALAG2 again yielded the best overall performance, with IEL2 and ASL2 achieving a reasonably good second-best performance. Almost always, ALAG2 produced somewhat better quality solutions, but ASL2 converged somewhat faster to a final solution as compared with ALAG2, although terminating at solutions that were slightly superoptimal (see Table 4.9). ASL2 and ALAG2 were considerably faster in converging to a final solution when compared with the other approaches. Another interesting observation was that for the NETLIB test problems, although the ASL2 approach was better than the other methods in speed of convergence, the quality of solution attained by this method was not satisfactory (see Table 4.10). It is worthwhile investigating alternative direction generation and step-size strategies within the context of IEL2 or ASL2 for equality constrained problems.

The results of the statistical analysis on the speed of convergence and the quality of solution are in accordance with the above observation for all sets of problems. As for speed of convergence, the order of ranking of the algorithms is ALAG2, ALAG3, ASL2, IEL2, and ALAG1 for the generally structured randomly generated linear programs. For the NETLIB test problems, the ranking was ASL2, IEL2, ALAG3, ALAG2, and ALAG1. For the set of linear programs having only equality constraints, the ranking was ASL2, ALAG2, ALAG3, IEL2, and ALAG1. The statistical analysis on the quality of solution verifies that IEL2 produces the best quality solutions. The order of ranking of the other methods is ALAG2, ALAG3, ALAG1, and ASL2. The statistical analysis on the speed of convergence for the high-density test problems admitted the order of ranking as ASL2, ALAG2, and IEL2. For the low-density test problems, the order was ALAG2, ASL2, and IEL2. The statistical analysis on the quality of solution both for the high-density and the low-density problems confirms that IEL2 produces the best quality solutions again. The order of ranking of the other methods is ASL2, and ALAG2. The actual numbers indicate that ASL2 achieves better quality solutions for larger problems due to a higher maximum number of inner loops permitted. However, the quality of solution is invariant with respect to problem size for ALAG2.

It is apparent from the experiments performed with the ALAG approach using known Lagrange multipliers that having better Lagrange multiplier estimates initially, or designing improved multiplier updating schemes for ALAG, is promising. For example, a deflected subgradient method applied to a Lagrangian dual formulation that dualizes all the structural constraints could be used to possibly find a beneficial starting dual solution.

Density (sparsity) is one of the parameters that affects the speed of convergence and the relative performances of the methods tested. For high-density problems, although CPLEX achieved an optimal solution much faster than any of the algorithms, ASL2 seems to be the best alternative among the methods tested. However, for highly sparse problems, ALAG2 attained the highest speed of convergence. Moreover, for most of the test problems, all of the investigated methods obtained a final solution faster than CPLEX, sometimes as fast as requiring only 16-23% of CPLEX's execution times.

In conclusion, when only equality constraints are present, the l_2 exterior penalty approach using a conjugate gradient strategy with exact line searches, and the augmented Lagrangian approach using the second violation criterion as in (3.20), appear to be the favorable choices among the methods tested. In the presence of inequality constraints (as well), the same version of the augmented Lagrangian approach appears to be a preferred approach, while the former l_2 penalty approach might be more suited for specially structured problems. However, the augmented Lagrangian method using a more frequent, individualized, updating scheme for the Lagrange multipliers associated with each constraint (as in the version ALAG3) seems to be a more robust approach for solving linear programming problems since its performance is not affected by the structure of the problem as much as the other methods considered. Both classes of methods could benefit by further experimentation using different search direction and step-size strategies, to explore their effects on the speed of convergence and the accuracy of the solution obtained. Furthermore, for large-scale, low-density linear programs, augmented Lagrangian method is faster in attaining a final solution than the other tested methods as well as CPLEX. Hence, we recommend that this method should be explored more extensively, perhaps using some revised multiplier updating scheme. Besides, although we have exploited the sparsity of the constraint coefficients in our computations, about 90% of the computational time was spent in the routine calculation of the gradient of the penalty function. Therefore, any improvement in this piece of code, perhaps exploiting inherent special structures, would be instrumental in reducing solution effort. Such algorithmic investigations, and more intensive computational testing on different classes of linear programs are recommended for further study.

BIBLIOGRAPHY

1. Al-Sultan, K. S., and Murty, K. G., Exterior point algorithms for nearest points and convex quadratic programs. *Mathematical Programming*, 1992, 57, 145-161.
2. Andrus, J. F., An exterior point method for convex programming problems. *J. Optimization Theory and Applications*, 1992, 72(1), 37-63.
3. Andrus, J. F. and Schaferkotter, M. R., An exterior-point method for linear programming problems. *J. Optimization Theory and Applications*, 1996, 91(3), 561-583.
4. Bazaraa, M. S., Jarvis, J. J. and Sherali, H. D., *Linear Programming and Network Flows*, 2nd edition. John Wiley & Sons, Singapore, 1990.
5. Bazaraa, M. S., Sherali, H. D. and Shetty, C. M., *Nonlinear Programming Theory and Algorithms*, 2nd edition. John Wiley & Sons, New York, 1993.
6. Ben-Tal, A. and Zibulevski, M., Penalty/barrier multiplier methods for convex programming problems. *SIAM J. Optimization*, 1997, 7(2), 347-366.
7. Bertsekas, D. P., On penalty and multiplier methods for constrained minimization. *SIAM J. Control and Optimization*. 1976, 14(2), 216-235.
8. Bertsekas, D. P., *Nonlinear Programming*, Athena Scientific, Belmont, 1995.
9. Cominetti, R. and Dussault, J. P., Stable exponential-penalty algorithm with superlinear convergence. *J. Optimization Theory and Applications*, 1994, 83(2), 285-309.
10. Conn, A. R., Gould, N. I. M., Sartenaer, A. and Toint Ph. L., Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints. *SIAM J. Optimization*, 1996, 6(3), 674-703.
11. Conn, A. R., Gould, N. I. M. and Toint Ph. L., A globally convergent Lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. *Mathematics of Computation*, 1997, 66(217), 261-288.
12. Fletcher, R., *Practical Methods of Optimization*, 2nd edition. John Wiley & Sons, New York, 1987.
13. Gill, P. E., Murray, W., Saunders, M. A., Tomlin, J. A., and Wright, M. H., On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Mathematical Programming*, 1986, 193-209.
14. Hestenes, M. R., Multiplier and gradient methods. *J. Optimization Theory and Applications*, 1969, 4, 303-320.
15. Karmarkar, N., A new polynomial-time algorithm for linear programming. *Combinatorica*, 1984, 4, 373-395.
16. Khachian, L. G., A polynomial algorithm in linear programming. *Soviet Mathematics and Doklady*, 1979, 20, 191-194.

17. Lustig, I. J., Marsten, R. E. and Shanno, D. F., Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra and its Applications*, 1991, 152, 191-222.

18. Megiddo, N., *Progress in mathematical programming: interior-point and related methods*, Springer-Verlag, New York, NY, 1989.

19. More, J. J., Toraldo, G., On the solution of large quadratic programming problems with bound constraints. *SIAM J. Optimization*, 1991, 1(1), 93-113.

20. Nash, S. G., Sofer, A., *Linear and Nonlinear Programming*. McGraw Hill, New York, 1996.

21. Nesterov, Y. E. and Nemirovskii, A., *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, Philadelphia, 1994.

22. Paparrizos, K., An infeasible (exterior point) simplex algorithm for assignment problems. *Mathematical Programming*, 1991, 51, 45-54.

23. Paparrizos, K., An exterior point simplex algorithm for (general) linear programming problems. *Annals of Operations Research*, 1993, 47, 497-508.

24. Powell, M. J. D., A method for nonlinear constraints in minimization problems. *Optimization*, R. Fletcher (Ed.), 1969.

25. Rosen, J. B. and Suzuki, S., Construction of nonlinear programming test problems. *Communication of ACM*, 1965, 8, 113.

26. Saigal, R., *Linear Programming: A Modern Integrated Analysis*. Kluwer Academic Publishers, Boston, 1995.

27. Sen, S. and Sherali, H. D., A class of convergent primal-dual subgradient algorithms for decomposable convex programs. *Mathematical Programming*, 1986, 35(3), 279-297.

28. Sherali, H. D., Choi, G. and Sen, S., An exterior-point polytope sliding and deformation algorithm for linear programming problems. *Informatica*, 1997, 8(4), 559-582.

29. Sherali, H. D. and Myers, D. C., Dual formulations and subgradient optimization strategies for linear programming relaxations of mixed-integer programs. *Discrete Applied Mathematics*, 1988, 20, 51-68.

30. Sherali, H. D. and Ulular, O., A primal-dual conjugate subgradient algorithm for specially structured linear and convex programming problems. *Applied Mathematics and Optimization*, 1989, 20, 193-221.

31. Sherali, H. D. and Ulular, O., Conjugate gradient methods using quasi-Newton updates with inexact line searches. *J. Mathematical Analysis and Applications*, 1990, 150(2), 359-377.

32. Terlaky, T., *Interior Point Methods of Mathematical Programming*, Kluwer, Boston, 1996.

VITA

Burak Özdaryal was born on March 15, 1973, in Istanbul, Turkey. He completed his pre-university education in Robert College, Istanbul in 1992. He received his Bachelor of Science Degree in Chemical Engineering in June 1997 from Boğaziçi University, Istanbul.

He entered the graduate program of the Industrial and Systems Engineering Department of Virginia Polytechnic Institute and State University in August 1997. The same department employed him as a Graduate Teaching Assistant from August 1997 to May 1998. From June 1998 to July 1999, he was employed by Dr. Hanif D. Sherali as a Graduate Research Assistant. He received his Master of Science Degree in Industrial and Systems Engineering Department in the Operations Research option, from Virginia Polytechnic Institute and State University in July 1999.

Upon completing the program, he was employed by United Airlines, Chicago, as an Operations Research Scientist.