A GRADIENT-BASED IMPLEMENTATION OF THE POLYHEDRAL ACTIVE SET ALGORITHM *

JAMES D. DIFFENDERFER[†], WILLIAM W. HAGER[‡], AND HONGCHAO ZHANG[§]

Abstract. The Polyhedral Active Set Algorithm (PASA) is designed to optimize a general nonlinear function over a polyhedron. Phase one of the algorithm is a nonmonotone gradient projection algorithm, while phase two is an active set algorithm that explores faces of the constraint polyhedron. A gradient-based implementation is presented, where a projected version of the conjugate gradient algorithm is employed in phase two. Asymptotically, only phase two is performed. Comparisons are given with IPOPT using polyhedral constrained problems from CUTEst and the Maros/Meszaros quadratic programming test set.

Key words. Nonlinear optimization; Polyhedral constrained optimization; Active set method; Gradient projection method; Projection on polyhedron; Conjugate gradient method; PASA; PPROJ; CG_DESCENT; NAPHEAP;

AMS subject classifications. 90C06, 90C26, 65Y20

1. Introduction. The polyhedral active set algorithm PASA is designed to solve the problem

$$(1.1) \qquad \min f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{x} \in \Omega,$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and Ω is a polyhedron. Throughout the paper, it is assumed that

$$(1.2) \qquad \Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \leq \mathbf{b}\},$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. The PASA software, on the other hand, utilizes the representation

$$(1.3) \qquad \Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{bl} \leq \mathbf{A}\mathbf{x} \leq \mathbf{bu}, \quad \mathbf{lo} \leq \mathbf{x} \leq \mathbf{hi}\},$$

with $\mathbf{bl}$ and $\mathbf{bu} \in \mathbb{R}^m$ and $\mathbf{lo}$ and $\mathbf{hi} \in \mathbb{R}^n$; any of the inequalities could be vacuous. The software is designed to exploit sparsity in $\mathbf{A}$.

The algorithms implemented in PASA have been developed over more than 20 years. In one series of papers [10, 15, 16, 17, 20], Timothy Davis and William Hager developed techniques for modifying a sparse Cholesky factorization of a matrix of the form $\mathbf{A}\mathbf{A}^\mathsf{T}$ after adding or deleting a small number of columns and rows from $\mathbf{A}$. These update/downdate techniques are optimal in the sense that their running time is proportional to the number of nonzeros in the Cholesky factorization that change. In another series of papers [18, 19, 32, 33, 34, 35, 36, 37], Hager developed the Dual Active Set Algorithm (DASA), first in a general context, and then with Donald Hearn, it was applied to quadratic network optimization; later with Timothy Davis [18, 19], it was applied to linear programming using the newly developed update/downdate

[†]`diffenderfer2@llnl.gov`, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94550.

[‡]`hager@ufl.edu`, http://people.clas.ufl.edu/hager/, PO Box 118105, Department of Mathematics, University of Florida, Gainesville, FL 32611-8105. Phone (352) 294-2308.

[§]`hozhang@math.lsu.edu`, http://www.math.lsu.edu/∼hozhang, Department of Mathematics, Louisiana State University, Baton Rouge, LA 70803-4918. Phone (225) 578-1982. Fax (225) 578-4276.

techniques. More recently, in [45] both DASA and the update/downdate techniques were used in an algorithm PPROJ to project a point onto a polyhedron.

In another series of papers, [38, 39, 42] William Hager and Hongchao Zhang developed a fast version of the conjugate gradient method known as CG_DESCENT since the search directions were always descent directions, independent of the line search. In [43], CG_DESCENT was enhanced using limited memory techniques. As an application of CG_DESCENT, an active set method for purely bound constrained problems was developed in [40, 41]. This active set algorithm had two phases, in phase one the gradient projection algorithm and a cyclic Barzilai/Borwein [3, 14] step were used to identify active constraints, and in phase two, an unconstrained solver, such as CG_DESCENT, optimized the objective over faces of the polyhedral constraint. Whenever a new constraint in the polyhedron became active, the optimization was restricted to the resulting smaller face of the polyhedron.

The polyhedral active set algorithm PASA in [44] is a generalization of the two phase algorithm in [40] from bound constraints to polyhedral constraints. Under nondegeneracy type assumptions, only the second phase is executed asymptotically; consequently, the asymptotic convergence speed coincides with that of the algorithm used to optimize the objective over the faces of a polyhedron. For a general polyhedron, the projected gradients of phase one are computed using PPROJ, while in the special case where the polyhedron is a knapsack-type constraint

$$\{\mathbf{x} \in \mathbb{R}^n : bl \le \mathbf{a}^\mathsf{T}\mathbf{x} \le bu, \ \mathbf{lo} \le \mathbf{x} \le \mathbf{hi}\}, \quad \mathbf{a} \in \mathbb{R}^n,$$

the projection is computed using the Newton/heap-based algorithm NAPHEAP of [21]. The current version of PASA uses a projected conjugate gradient iteration in phase two to optimize over a shrinking series of faces of the polyhedron.

2. Literature Review. We briefly summarize continuous nonlinear optimization algorithm development during the past 30 years. Early codes in this timeframe include MINOS [48], NPSOL [25], OPTPACK [30, 33], and LANCELOT [13]. Murtagh and Saunders' MINOS is based on Robinson's algorithm [49] which is locally quadratically convergent. The Lagrangian in Robinson's algorithm is replaced by an augmented Lagrangian, and the subproblem associated with the linearized constraints are solved by a reduced gradient algorithm combined with a quasi-Newton method as described in [46, 47]. Gill, Murray, Saunders, and Wright's NPSOL is a sequential quadratic programming method (SQP) where a positive definite quasi-Newton approximation to the true Lagrangian Hessian is utilized. The resulting quadratic programming problem is solved by codes in the LSSOL package [26], which employs active set methods and dense linear algebra to solve constrained linear least-squares problems and convex quadratic programming problems. OPTPACK [30, 33] alternates between a constraint step based on Newton's method and an optimization step based on the minimization of an augmented Lagrangian over linearized constraints. The combined steps are locally quadratically convergent, and the implementation employs dense linear algebra. Conn, Gould, and Toint's LANCELOT treats nonlinear constraints using an augmented Lagrangian which is minimized within a region defined by the bound constraints. The bound constrained problem is solved by an algorithm that combines projected gradient techniques [11] and special structures to exploit the group partially separable structure of a problem [12].

More recently, NPSOL was the starting point for SNOPT (Sparse Nonlinear Optimizer) [24] and DNOPT (Dense Nonlinear Optimizer) [27]. Again, a positive definite quasi-Newton approximation to the true Lagrangian Hessian is employed. However,

in SNOPT sparse linear algebra is used to solve the resulting quadratic program, while DNOPT employs dense linear algebra. A different SQP algorithm is developed by Fletcher and Leyffer in [22, 23], where a trust region approach is applied to the quadratic programming problem, and the accepted iterates are chosen using a filter method. The authors of the augmented Lagrangian-based code LANCELOT changed their focus to efficient quadratic programming (QP) solvers that could be used in the implementation of SQP methods. The new software formed the package GALAHAD. Further development of reliable augmented Lagrangian techniques for general nonlinear optimization were continued by Birgin, Martínez, and others in the ALGENCAN [1] package. After the success of interior point methods for linear programming, interior point algorithms were also developed for general nonlinear programming including Vanderbei and Shanno's LOQO [50] (a merit line-search interior point method based on a quadratic program solver also named LOQO), Waltz and Nocedal's KNITRO [52] (an interior point approach based on sequential quadratic programming and trust regions [8, 9]), and Biegler and Wächter's IPOPT [51] which employs an interior point method with a filter line-search. IPOPT has been adopted as a COIN-OR project (Computational Infrastructure for Operations Research); performance data often shows IPOPT with the best performance among the NLP solvers.

3. Overview of PASA. As discussed in the introduction, PASA has two phases: gradient projection iterations over the entire polyhedron in phase one and projected (conjugate) gradient iterations in phase two to optimize over faces of the polyhedron. To choose between the two phases, we compare the violation of the local optimality conditions for the global problem (1.1) to the violation in the optimality conditions on the current face of the polyhedron (the local problem). An estimate of the violation in the optimality conditions for the global problem is given by

$$E(\mathbf{x}) = \|P_\Omega(\mathbf{x} - \nabla f(\mathbf{x})) - \mathbf{x}\|,$$

where P_Ω denotes the Euclidean norm projection given by

$$(3.1) \qquad P_\Omega(\mathbf{x}) := \arg\min_{\mathbf{y}}\{\|\mathbf{x} - \mathbf{y}\|^2 : \mathbf{y} \in \Omega\}.$$

Recall [40, P7] that $E(\mathbf{x}) = 0$ if and only if $\mathbf{x}$ is a stationary point for the global problem (1.1). After a change of variables, we obtain

$$(3.2) \qquad E(\mathbf{x}) = \|\mathbf{y}(\mathbf{x})\| \text{ where } \mathbf{y}(\mathbf{x}) = \arg\min_{\mathbf{y}}\{\|\mathbf{y} + \nabla f(\mathbf{x})\| : \mathbf{y} \in \Omega - \mathbf{x}\}.$$

Thus the global error at $\mathbf{x}$ is the projection of the negative gradient $-\nabla f(\mathbf{x})$ onto the shifted polyhedron $\Omega - \mathbf{x}$.

Suppose that Ω is expressed in the form (1.2), and for any feasible point $\mathbf{x}$, let $\mathcal{A}(\mathbf{x})$ denote the active (binding) constraints:
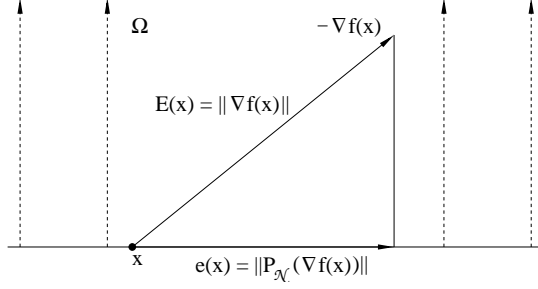
$$\mathcal{A}(\mathbf{x}) = \{i : (\mathbf{A}\mathbf{x} - \mathbf{b})_i = 0\}.$$

The active manifold at $\mathbf{x}$ is

$$\mathcal{M}(\mathbf{x}) = \mathbf{x} + \mathcal{N}(\mathbf{A}_B), \quad B = \mathcal{A}(\mathbf{x}),$$

where $\mathbf{A}_B$ is the submatrix of $\mathbf{A}$ corresponding to row indices in $\mathcal{A}(\mathbf{x})$ and $\mathcal{N}(\mathbf{A}_B)$ is the null space of $\mathbf{A}_B$. The local problem corresponding to the active manifold at $\mathbf{x}$ is

$$(3.3) \qquad \min f(\mathbf{z}) \quad \text{subject to} \quad \mathbf{z} \in \mathcal{M}(\mathbf{x}).$$

FIG. 3.1. *Global error $E(x)$ versus local error $e(x)$.*

By (3.2) with Ω replaced by $\mathcal{M}(\mathbf{x})$, $\mathbf{x}$ is a stationary point for the local problem (3.3) if and only if $e(\mathbf{x}) = 0$ where

$$e(\mathbf{x}) = \|\mathbf{y}_B(\mathbf{x})\|, \quad \mathbf{y}_B(\mathbf{x}) = \arg\min_{\mathbf{y}}\{\|\mathbf{y} + \nabla f(\mathbf{x})\| : \mathbf{A}_B\mathbf{y} = \mathbf{0}\}.$$

Thus the local error bound at $\mathbf{x}$ is the projection of the negative gradient $-\nabla f(\mathbf{x})$ onto $\mathcal{N}(\mathbf{A}_B)$.

A simple illustration of the local and global errors is given in Figure 3.1. The set Ω is the upper half-space and the point $\mathbf{x}$ lies on the boundary of Ω. Since the negative gradient at $\mathbf{x}$ points into Ω, $E(\mathbf{x})$ is simply the norm of the negative gradient. The active manifold is the horizontal axis, and $e(\mathbf{x})$ is the projection of the negative gradient onto the horizontal axis.

In implementing PASA, we choose a parameter $\theta \in (0,1)$, and then operate in either phase one or phase two as indicated in Algorithm 3.2.

Parameters: $\theta \in (0,1)$, $\tau \in (0,\infty)$, start guess $\mathbf{x}_0 \in \mathbb{R}^n$
Initialization: $\mathbf{x}_1 = \mathcal{P}_\Omega(\mathbf{x}_0)$, $k = 1$
Phase one: While $E(\mathbf{x}) > \tau$, execute phase one
                  Possibly reduce θ
                  If $e(\mathbf{x}_k) \geq \theta E(\mathbf{x}_k)$, goto phase two;
                  else $k \leftarrow k+1$.
              End

Phase two: While $E(\mathbf{x}) > \tau$, execute phase two
                  Possibly reduce θ
                  If $e(\mathbf{x}_k) < \theta E(\mathbf{x}_k)$, goto phase one;
                  else $k \leftarrow k+1$.
              End

ALG. 3.2. *Sketch of Polyhedral Active Set Algorithm (PASA).*

As seen in Algorithm 3.2, the branching between the two phases of PASA is based on a comparison between E and e at the current iterate $\mathbf{x}_k$. Regardless of the adjustments to θ, it is shown in [44] that

$$\liminf_{k\to\infty} E(\mathbf{x}_k) = 0,$$

whenever the algorithm in phase two satisfies the conditions (P1)–(P3) given below in Section 5. Moreover, under either a nondegeneracy or strong second-order sufficient
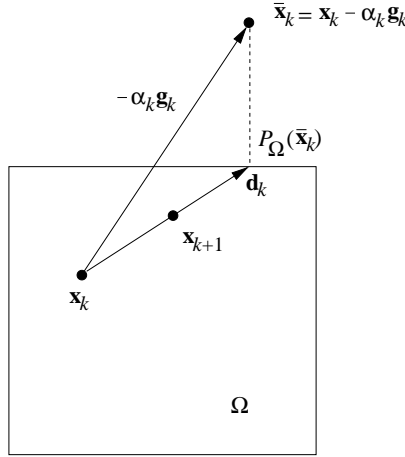
FIG. 4.1. *Sketch of the gradient projection algorithm.*

optimality with linear independence of the active constraint gradients, the iterates of PASA are only generated by phase two when k is sufficiently large. To achieve this stronger property, an adjustment was made in [44] of the form $\theta \leftarrow \mu\theta$, $\mu \in (0,1)$, whenever the "undecided index set was empty" in phase one. The undecided indices corresponded to constraints for which the associated multipliers were either sufficiently positive or the constraint was sufficiently active. Determining whether the undecided index set was empty required an additional projection which detracted from the efficiency of PASA. Note that any update to θ in phase one which drives it to zero guarantees that the iterates are generated by phase two asymptotically under the condition given above. Since phase one often branches to phase two after a single iteration, a practical approach for driving θ to zero when too much time is spent in phase one is to decrease θ whenever more than one iteration is performed in phase one.

4. Phase One. The version of the gradient projection algorithm that we utilize is depicted in Figure 4.1. At the current iterate $\mathbf{x}_k$, a step of length α_k is taken along the negative gradient $-\mathbf{g}_k = -\nabla f(\mathbf{x}_k)$ to reach a point $\overline{\mathbf{x}}_k$, whose projection onto the polyhedron Ω is $P_\Omega(\overline{\mathbf{x}}_k)$. A line search is performed along the search direction $\mathbf{d}_k = P_\Omega(\overline{\mathbf{x}}_k) - \mathbf{x}_k$ to obtain the next iterate $\mathbf{x}_{k+1}$. Our choice for the stepsize α_k is based on the BB formula [3] with a cyclic implementation which is explained in [14]. In a cyclic implementation, the stepsize is kept fixed in some iterations (that is, $\alpha_{k+1} = \alpha_k$), while in other iterations, it is given by the BB formula. As shown in [14], a cyclic implementation can lead to better performance.

Our version of the gradient projection algorithm is similar to the version in SPG [4] since the line search is along the line segment connecting $\mathbf{x}_k$ to the projected point $P_\Omega(\overline{\mathbf{x}}_k)$. There is another version of the gradient projection method in which $\mathbf{x}_{k+1} = P_\Omega(\mathbf{x}_k - s_k \nabla f(\mathbf{x}_k))$ where the stepsize s_k is chosen to satisfy both a descent condition and a condition to ensure the s_k is not too small; for example, see [5, 6, 7] and the references therein. For this scheme, the active constraints at a minimizer can be identified in finite number of iterations under suitable assumptions, however, more than one projection may be needed to determine an acceptable step. Although the gradient projection algorithm implemented in PASA may not identify the active

---

Parameters: δ and $\eta \in (0,1)$, $\alpha_k \in (0, \infty)$

While $E(\mathbf{x}_k) > \max \{\tau, \ e(\mathbf{x}_k)/\theta\}$

    1. $\mathbf{d}_k = \mathcal{P}_\Omega(\mathbf{x}_k - \alpha_k \mathbf{g}_k) - \mathbf{x}_k$

    2. $s_k = \eta^j$ where $j \geq 0$ is smallest integer such that
        $f(\mathbf{x}_k + s_k \mathbf{d}_k) \leq f_k^r + s_k \delta \nabla f(\mathbf{x}_k) \mathbf{d}_k$

    3. $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{d}_k$ and $k \leftarrow k + 1$

End

ALG. 4.2. *Phase one (gradient projection algorithm).*

constraints, we show in [44, Lemma 6.2] that the violation in the active constraints is on the order of the squared error in the iterate $\mathbf{x}_k$.

The line search implemented in PASA, shown in Algorithm 4.2, is of Armijo type [2]. If $f_k^r = f(\mathbf{x}_k)$, then this is an ordinary Armijo line search restricted to the feasible set Ω. However, the implemented line search is nonmonotone, such as in [29], but with a more sophisticated choice of f_k^r, based on the procedure given in the appendix of [40]. Also, to avoid potential breakdown of the line search in a neighborhood of an optimum, an approximate (but more accurate) line search is used near a local minimizer; see the approximate Wolfe line search in [38].

5. Phase Two. Next, we discuss the requirements of the active set algorithm in phase two. According to the theory developed in [44], any algorithm with the following properties can be used in phase two:

    P1. For each k, $\mathbf{x}_k \in \Omega$ and $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$.

    P2. For each k, $\mathcal{A}(\mathbf{x}_k) \subset \mathcal{A}(\mathbf{x}_{k+1})$.

    P2. If $\mathcal{A}(\mathbf{x}_{j+1}) = \mathcal{A}(\mathbf{x}_j)$ for $j \geq k$, then $\liminf_{j \to \infty} e(\mathbf{x}_j) = 0$.

The current gradient-based implementation of PASA combines an *active set* gradient projection algorithm with a *projected* version of the conjugate gradient method. Let us define the set

$$\Omega_k = \{\mathbf{x} \in \Omega : (\mathbf{A}\mathbf{x} - \mathbf{b})_i = 0 \text{ for all } i \in \mathcal{A}(\mathbf{x}_k)\}.$$

By an active set gradient projection algorithm (A-GP), we mean that

$$\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{d}_k, \quad \text{where } \mathbf{d}_k = \mathcal{P}_{\Omega_k}(\mathbf{x}_k - \alpha_k \mathbf{g}_k) - \mathbf{x}_k.$$

This is the gradient project step of Algorithm 4.2 except that Ω is replaced by Ω_k. Since the new iterate $\mathbf{x}_{k+1}$ may have additional active constraints, we have $\mathcal{A}(\mathbf{x}_k) \subset \mathcal{A}(\mathbf{x}_{k+1})$. As long as phase two does not reach one of the termination conditions in Algorithm 3.2 and $\mathcal{A}(\mathbf{x}_k)$ is strictly contained in $\mathcal{A}(\mathbf{x}_{k+1})$, A-GP continues to operate. At any iterate $\mathbf{x}_k$ where $\mathcal{A}(\mathbf{x}_k) = \mathcal{A}(\mathbf{x}_{k-1})$, we switch in phase two from A-GP to the projected conjugate gradient scheme, which is now explained.

Let $\mathbf{A}_k$ denote the submatrix of $\mathbf{A}$ associated with the active constraint gradients at $\mathbf{x}_k$, and let $\mathbf{P}_k \in \mathbb{R}^{n \times n}$ denote the orthogonal projection onto the null space $\mathcal{N}(\mathbf{A}_k)$. If $\mathbf{x}_k$ is the iterate of the A-GP where $\mathcal{A}(\mathbf{x}_k) = \mathcal{A}(\mathbf{x}_{k-1})$, then we apply the limited memory CG_DESCENT method [43] to the unconstrained optimization problem

(5.1) $\min \{f(\mathbf{x}_k + \mathbf{P}_k \mathbf{y}) : \mathbf{y} \in \mathbb{R}^n\}$,

While $e(\mathbf{x}_k)/\theta \geq E(\mathbf{x}_k) > \tau$

    1. Perform A-GP until $\mathcal{A}(\mathbf{x}_k) = \mathcal{A}(\mathbf{x}_{k-1})$, then branch to step 2.

    2. Apply limited memory CG_DESCENT to (5.1); branch to

       step 1 when reaching $\mathbf{x}_{k+1}$ on the boundary of Ω and $k \leftarrow k+1$.

End

ALG. 5.1. *Phase two (A-GP and CG_DESCENT).*

starting from $\mathbf{y}_0 = \mathbf{x}_k$. Whenever an iterate $\mathbf{y}_{j+1}$ lies outside of Ω, we let $\mathbf{x}_{k+1}$ be the last point on the line segment from $\mathbf{y}_j$ to $\mathbf{y}_{j+1}$ with the property that $\mathbf{x}_{k+1} \in \Omega$. A-GP continues from the new iterate $\mathbf{x}_{k+1}$. Phase two is summarized in Algorithm 5.1

A stability issue arises when applying the conjugate gradient method to the unconstrained problem (5.1). Theoretically, the projection can be expressed as

$$\mathbf{P}_k = \mathbf{I} - \mathbf{A}_k^\mathsf{T}(\mathbf{A}_k\mathbf{A}_k^\mathsf{T})^{-1}\mathbf{A}_k,$$

where the inverse should be replaced by a pseudoinverse when the rows of $\mathbf{A}_k$ are linearly dependent, and $\mathbf{P}_k$ is a positive semidefinite matrix. The routine PPROJ, used in phase one computes the factorization

$$\mathbf{A}_k\mathbf{A}_k^\mathsf{T} + \sigma\mathbf{I} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T},$$

where $\sigma > 0$ is relatively small, $\mathbf{L}$ is lower triangular with ones on the diagonal, and $\mathbf{D}$ is diagonal. This leads us to replace $\mathbf{P}_k$ in (5.1) by its approximation

$$\widetilde{\mathbf{P}}_k = \mathbf{I} - \mathbf{A}_k^\mathsf{T}(\mathbf{A}_k\mathbf{A}_k^\mathsf{T} + \sigma\mathbf{I})^{-1}\mathbf{A}_k = \mathbf{I} - \mathbf{A}_k^\mathsf{T}(\mathbf{L}\mathbf{D}\mathbf{L})^{-\mathsf{T}}\mathbf{A}_k,$$

a positive definite matrix.

Since our goal is to optimize the objective over $\mathbf{x} = \mathbf{x}_k + \mathbf{P}_k\mathbf{y}$, it is convenient to formulate the algorithm for solving (5.1) in terms of $\mathbf{x}$ rather than in terms of $\mathbf{y}$. In particular, when the iterates are given by the CG_DESCENT family parameterized by $\eta > 1/4$, the search directions are (see [43, Sect. 2])

$$(5.2) \qquad \mathbf{d}_0 = -\widetilde{\mathbf{P}}_k^2\mathbf{g}_0, \quad \mathbf{d}_{k+1} = -\widetilde{\mathbf{P}}_k^2\mathbf{g}_{k+1} + \beta_k\mathbf{d}_k \quad \text{for } k \geq 0,$$

where $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ and

$$(5.3) \qquad \beta_k = \frac{\mathbf{y}_k^\mathsf{T}\widetilde{\mathbf{P}}_k^2\mathbf{g}_{k+1}}{\mathbf{d}_k^\mathsf{T}\mathbf{y}_k} - \eta\frac{\|\widetilde{\mathbf{P}}_k\mathbf{y}_k\|^2}{\mathbf{d}_k^\mathsf{T}\mathbf{y}_k}\frac{\mathbf{d}_k^\mathsf{T}\mathbf{g}_{k+1}}{\mathbf{d}_k^\mathsf{T}\mathbf{y}_k}, \quad \mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k.$$

Even though $\widetilde{\mathbf{P}}_k$ is an approximation to $\mathbf{P}_k$ and $\mathbf{P}_k^2 = \mathbf{P}_k$, the replacement of $\widetilde{\mathbf{P}}_k^2$ by $\widetilde{\mathbf{P}}_k$ leads to very poor performance. Moreover, when we compute the search directions by the formula (5.2), the iterates quickly lose feasibility. The reason is that the component of the error in $\mathbf{d}_k$ pointing out of $\mathcal{N}(\mathbf{A}_k)$ is added into $\mathbf{d}_{k+1}$ in (5.2), and these errors in the search direction can accumulate. The following iteration is equivalent to (5.2) and numerically stable since $\mathbf{d}_{k+1}$ is the product of an intermediate vector $\mathbf{D}_{k+1}$ with $\widetilde{\mathbf{P}}_k$ which removes error components orthogonal to the null space of $\mathbf{A}_k$:

$$(5.4) \quad \mathbf{D}_0 = -\widetilde{\mathbf{P}}_k\mathbf{g}_0, \quad \mathbf{D}_{k+1} = -\widetilde{\mathbf{P}}_k\mathbf{g}_{k+1} + \beta_k\mathbf{D}_k, \quad \mathbf{d}_{k+1} = \widetilde{\mathbf{P}}_k\mathbf{D}_{k+1} \text{ for } k \geq 0.$$

6. Results. We compare the performance of PASA Version 2.0.0 to the performance of IPOPT Version 3.14.5 using the CUTEst platform [28], and polyhedral constrained optimization problems from CUTEst along with the Maros/Meszaros quadratic programming test set. IPOPT can operate in a gradient-based mode, where the Hessian of the Lagrangian in the KKT system is approximated by a limited memory quasi-Newton method (L-BFGS), and a Hessian-based mode where both the gradient and Hessian of the objective and the constraints are provided, and a direct solver is used for the linear systems. We installed both of the recommended linear solvers: MUMPS 5.4.1 and the HSL software, which includes MA57, Version 3.11.0. When IPOPT was run, it always chose the MUMPS linear solver. In comparisons between the gradient and Hessian-based IPOPT, the Hessian-based version performed much better. Hence, our comparisons are with Hessian-based IPOPT. Note that comparisons between the Hessian-based PASA (currently under development) and the gradient-based PASA also indicate that Hessian-based PASA is superior to gradient-based PASA.

In selecting the problem set for the numerical experiments, 42 of the QPs from CUTEst were excluded. The names of these problems begin with the letter A followed by either 0 or 2 or 5. There were two issues with this subset of the CUTEst test set. First, in many cases, the starting point is essentially a stationary point, and gradient-based PASA immediately terminates. Second, these problems have between 15,000 and 20,000 linear constraints, and a small number of dense columns. Due to the dense columns, the matrix $\mathbf{AA}^\mathsf{T}$ is dense with dimension between 15,000 and 20,000. To handle these problems efficiently, the dense columns need to be removed and processed using a Woodbury update [31]. We have not yet had time to incorporate Woodbury updates in PASA. Moreover, if the Woodbury updates were incorporated in the code, termination may occur at the starting point, and the problem would be excluded by the rules given in the next paragraph. Note that the Hessian-based PASA should handle these problems without difficulty since the KKT system has a sparse factorization.

After these exclusions, we start with 655 problems which we tried to solve to the accuracy tolerance 1.e−6. If the objective values computed by each solver agreed to 4 significant digits, then we accepted the problem. If 4 digit agreement was not achieved, then we examined the computed solutions. If the solvers were converging to different solutions, then we removed the problem from the test set; in other words, we focused on problems where both solvers started from the same initial guess and reached the same solution. There were 75 problems where the solvers converged to different solutions. In 38 cases, the solution computed by IPOPT had a better objective value, and in 37 cases, the solution computed by PASA had a better objective value. Note that among the 38 cases where IPOPT had a better objective value, it was observed that in a number of these cases, the starting guess was essentially a stationary point, and PASA stopped immediately, while the Hessian-based IPOPT did not stop at the starting point. Both solvers, however, are only guaranteed to converge to a stationary point.

After pruning the 75 problems where the solvers converged to different solutions, there were 580 remaining test problems. If the objective values disagreed by more than 4 significant digits but the solvers were converging to the same solution, we then adjusted the accuracy tolerance of the less accurate solver so as to achieve comparable accuracy to that of the more accurate solver. In these cases where one solver was more accurate than the other, we found that the PASA estimate $E(\mathbf{x})$ for the solution
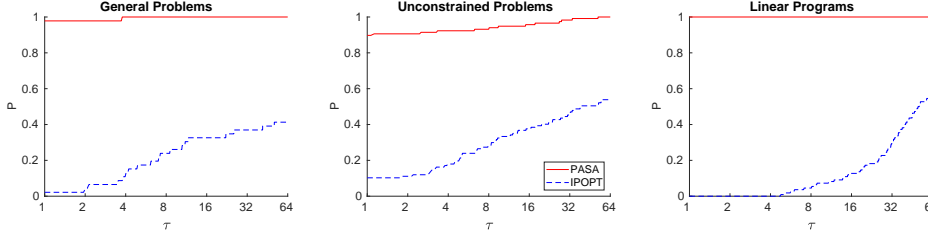
FIG. 6.1. *Wall time performance profiles for general, unconstrained, and linear programs*
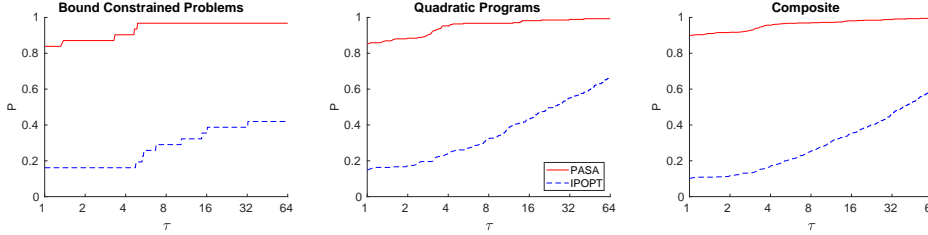

FIG. 6.2. *Wall time performance profiles for bound, quadratic, and composite programs*

tolerance resulted in a more accurate objective value in most cases. When the accuracy tolerance of IPOPT was adjusted to match the accuracy of PASA, often just one or two more iterations were needed. When a solver was unable to achieve the accuracy tolerance 1.e−6 for a problem, its computing time was set to ∞.

The performance of the gradient-based PASA and Hessian-based IPOPT are compared using wall time. Note that Hessian-based algorithms such as either IPOPT or the Hessian-based PASA, typically require fewer iterations and evaluations (function, gradient, and Hessian) when compared to the gradient-based PASA. The trade-offs between convergence rate of an algorithm and evaluation time are not studied in this paper, instead we focus on wall time. The run data is available at:

https://people.clas.ufl.edu/hager/files/IPOPTresults.txt
https://people.clas.ufl.edu/hager/files/PASAresults.txt

Both solvers exploit multiple processors when matrices are factored and linear systems are solved. The software was run on a Lenovo ThinkPad with 8 Intel i7-865U CPUs operating at 1.90GHz (4 cpu cores) with 8,192 KB cache and 16 GB memory. The operating system was Ubuntu Linux with Intel's MKL (Math Kernel Library) BLAS. PASA used the timer *gettimeofday* with microsecond accuracy, while IPOPT appears to use the timer *ftime* with millisecond accuracy (embedded inside the function IpCoinGetTimeOfDay).

Figures 6.1–6.2 plot the wall time performance profiles for the two codes. The vertical axis gives the fraction P of problems for which any given method is within a factor τ (horizontal axis) of the best time. The top curve is the method that solved the most problems in a time that was within a factor τ of the best time. The percentage of the test problems for which a method is fastest is given on the left axis of the plot. The right side of the plot gives the percentage of the test problems that were successfully solved by each of the methods. In essence, the right side is a measure of an algorithm's robustness.

In preparing the plots, the problems in the test set were partitioned into 5 groups:

linear and quadratic programs denote polyhedral constrained problem for which the objective is linear or quadratic respectively. General problems have nonquadratic nonlinear objectives with additional linear and possibly bound constraints. Bound constrained problems have nonquadratic nonlinear objectives and only bound constraints. Unconstrained problems have nonquadratic nonlinear objectives without constraints. The composite problems are the union of all 5 groups. Based on the plots, gradient-based PASA performed relatively well on this collection of test problems where function and gradient evaluations are relatively cheap; the cost of the linear algebra in IPOPT for solving the linear systems of equations outweighed the savings associated with a lower number of evaluations.

When a problem is unconstrained, $E(\mathbf{x}) = e(\mathbf{x})$ and since $\theta < 1$, PASA immediately branches to phase two, where it remains until the convergence tolerance is satisfied. Hence, the performance on unconstrained problems essentially reflects the performance of limited memory CG_DESCENT [43]. For bound constrained problems, PASA's local and global error estimators $e(\mathbf{x})$ and $E(\mathbf{x})$ reduce to the same estimators that were used in the algorithm [40] for bound constrained problems. Hence, the performance on bound constrained problem essentially reflects the performance of the active set algorithm [40]. Linear programs are solved in PASA by a series of gradient projection steps, where the stepsize choice is crucial. The performance corresponds to the first-order algorithm in [19]. Details will be provided in a separate paper.

7. Conclusion. A gradient-based implementation of the Polyhedral Active Set Algorithm (PASA) was presented. The algorithm was composed of two phases, the gradient projection algorithm was used in phase one, while phase two optimized the objective over faces of the polyhedron. Branching between phases was determined by the relationship between local and a global error estimators e and E respectively. At a feasible point $\mathbf{x}$ for the polyhedron, we branch from phase one to phase two when $e(\mathbf{x}) \geq \theta E(\mathbf{x})$, where $\theta \in (0, 1)$ is a given parameter; we branch from phase two to phase one when $e(\mathbf{x}) < \theta E(\mathbf{x})$. With suitable adjustments to θ, the iterates perform phase two asymptotically. It was found that PASA had significantly better wall time performance when compared to IPOPT using a collection of 580 test problems taken from both CUTEst and the Maros/Meszaros quadratic programming test set. Even though Hessian-based IPOPT used significantly fewer evaluations of the objective and gradient when compared to gradient-based PASA, the time for the linear algebra in IPOPT outweighed the savings derived from the fewer evaluations in the test set.

REFERENCES

[1] R. ANDREANI, E. G. BIRGIN, J. M. MARTÍNEZ, AND M. L. SCHUVERDT, *On augmented Lagrangian methods with general lower-level constraints*, SIAM J. Optim., (2007), pp. 1286–1309.
[2] L. ARMIJO, *Minimization of functions having Lipschitz continuous first partial derivatives*, Pacific J. Math., 16 (1966), pp. 1–3.
[3] J. BARZILAI AND J. M. BORWEIN, *Two point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.
[4] E. G. BIRGIN, J. M. MARTÍNEZ, AND M. RAYDAN, *Algorithm 813: SPG - software for convex-constrained optimization*, ACM Trans. Math. Software, 27 (2001), pp. 340–349.
[5] J. V. BURKE AND J. J. MORÉ, *On the identification of active constraints*, SIAM J. Numer. Anal., 25 (1988), pp. 1197–1211.

[6] ———, *Exposing constraints*, SIAM J. Optim., 25 (1994), pp. 573–595.

[7] J. V. Burke, J. J. Moré, and G. Toraldo, *Convergence properties of trust region methods for linear and convex constraints*, Math. Prog., 47 (1990), pp. 305–336.

[8] R. Byrd, J. Nocedal, and R. A. Waltz, *Knitro: An integrated package for nonlinear optimization*, in Large-Scale Nonlinear Optimization, G. di Pillo and M. Roma, eds., Springer-Verlag, 2006, pp. 35–59.

[9] R. H. Byrd, M. E. Hribar, and J. Nocedal, *An interior point method for large scale nonlinear programming*, SIAM J. Optim., 9 (1999), pp. 877–900.

[10] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, *Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate*, ACM Trans. Math. Software, 35 (2009), pp. 22:1–14.

[11] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Global convergence of a class of trust region algorithms for optimization with simple bounds*, SIAM J. Numer. Anal., 25 (1988), pp. 433–460.

[12] ———, *An introduction to the structure of large scale nonlinear optimization problems and the LANCELOT project*, in Computing Methods in Applied Sciences and Engineering, R. Glowinski and A. Lichnewsky, eds., Philadelphia, 1990, SIAM, pp. 42–54.

[13] ———, *LANCELOT: a Fortran Package for Large-scale Nonlinear Optimization (Release A)*, Springer, 1992.

[14] Y. H. Dai, W. W. Hager, K. Schittkowski, and H. Zhang, *The cyclic Barzilai-Borwein method for unconstrained optimization*, IMA J. Numer. Anal., 26 (2006), pp. 604–627.

[15] T. A. Davis and W. W. Hager, *Modifying a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 606–627.

[16] ———, *Multiple-rank modifications of a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 997–1013.

[17] ———, *Row modifications of a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 621–639.

[18] ———, *Dual multilevel optimization*, Math. Program., 112 (2008), pp. 403–425.

[19] ———, *A sparse proximal implementation of the LP Dual Active Set Algorithm*, Math. Program., 112 (2008), pp. 275–301.

[20] ———, *Dynamic supernodes in sparse Cholesky update/downdate and triangular solves*, ACM Trans. Math. Software, 35 (2009), pp. 27:1–23.

[21] T. A. Davis, W. W. Hager, and J. T. Hungerford, *An efficient hybrid algorithm for the separable convex quadratic knapsack problem*, ACM Trans. Math. Software, 42 (2016), pp. 22:1–22:25.

[22] R. Fletcher and S. Leyffer, *Nonlinear programming without a penalty function*, Math. Program., 91 (2002), pp. 239–270.

[23] R. Fletcher, S. Leyffer, and P. L. Toint, *On the global convergence of a filter-SQP algorithm*, SIAM J. Optim., 13 (2002), pp. 44–59.

[24] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Review, 47 (2005), pp. 99–131.

[25] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *User's Guide for LSSOL (Version 1.0)*, Tech. Rep. Report No. 86-1, Stanford University, Department of Operations Research, Stanford, CA, 1986.

[26] ———, *User's Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming*, Tech. Rep. Report No. 86-2, Stanford University, Department of Operations Research, Stanford, CA, 1986.

[27] P. E. Gill, M. A. Saunders, and E. Wong, *An SQP method for medium-scale nonlinear programming*, Tech. Rep. CCoM 16-2, Center for Computational Mathematics, Department of Mathematics, University of California, La Jolla, California, 2016.

[28] N. I. M. Gould, D. Orban, and P. L. Toint, *CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization*, Comput. Optim. Appl., 60 (2015), pp. 545–557.

[29] L. Grippo, F. Lampariello, and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM J. Numer. Anal., 23 (1986), pp. 707–716.

[30] W. W. Hager, *Dual techniques for constrained optimization*, J. Optim. Theory Appl., 55 (1987), pp. 37–71.

[31] ———, *Updating the inverse of a matrix*, SIAM Review, 31 (1989), pp. 221–239.

[32] ———, *The dual active set algorithm*, in Advances in Optimization and Parallel Computing, P. M. Pardalos, ed., North Holland, Amsterdam, 1992, pp. 137–142.

[33] ———, *Analysis and implementation of a dual algorithm for constrained optimization*, J. Optim. Theory Appl., 79 (1993), pp. 427–462.

[34] ———, *The LP dual active set algorithm*, in High Performance Algorithms and Software in Nonlinear Optimization, R. D. Leone, A. Murli, P. M. Pardalos, and G. Toraldo, eds., Dordrecht, 1998, Kluwer, pp. 243–254.

[35] ———, *The dual active set algorithm and its application to linear programming*, Comput. Optim. Appl., 21 (2002), pp. 263–275.

[36] ———, *The dual active set algorithm and the iterative solution of linear programs*, in Novel Approaches to Hard Discrete Optimization, P. M. Pardalos and H. Wolkowicz, eds., vol. 37, Fields Institute Communications, 2003, pp. 95–107.

[37] W. W. HAGER AND D. W. HEARN, *Application of the dual active set algorithm to quadratic network optimization*, Comput. Optim. Appl., 1 (1993), pp. 349–373.

[38] W. W. HAGER AND H. ZHANG, *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM J. Optim., 16 (2005), pp. 170–192.

[39] ———, *Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent*, ACM Trans. Math. Software, 32 (2006), pp. 113–137.

[40] ———, *A new active set algorithm for box constrained optimization*, SIAM J. Optim., 17 (2006), pp. 526–557.

[41] ———, *Recent advances in bound constrained optimization*, in System Modeling and Optimization, Proceedings of the 22nd IFIP TC7 Conference, Turin, Italy, July 18–22, 2005, Turin, Italy, F. Ceragioli, A. Dontchev, H. Furuta, K. Marti, and L. Pandolfi, eds., Springer, 2006, pp. 67–82.

[42] ———, *A survey of nonlinear conjugate gradient methods*, Pacific J. Optim., 2 (2006), pp. 35–58.

[43] ———, *The limited memory conjugate gradient method*, SIAM J. Optim., 23 (2013), pp. 2150–2168.

[44] ———, *An active set algorithm for nonlinear optimization with polyhedral constraints*, Sci. China Math., 59 (2016), pp. 1525–1542.

[45] ———, *Projection onto a polyhedron that exploits sparsity*, SIAM J. Optim., 29 (2016), pp. 1773–1798.

[46] B. A. MURTAGH AND M. A. SAUNDERS, *Large-scale linearly constrained optimization*, Math. Prog., 14 (1978), pp. 41–72.

[47] ———, *A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints*, Math. Prog. Study, 16 (1982), pp. 84–117.

[48] ———, *MINOS 5.0: User's Guide*, Tech. Rep. Report No. 83-20R, Stanford University, Department of Operations Research, Stanford, CA, 1987.

[49] S. M. ROBINSON, *A quadratically-convergent algorithm for general nonlinear programming problems*, Math. Prog., 3 (1972), pp. 145–156.

[50] R. J. VANDERBEI AND D. F. SHANNO, *An interior-point algorithm for nonconvex nonlinear programming*, Comput. Optim. Appl., 13 (1999), pp. 231–252.

[51] A. WÄCHTER AND L. T. BIEGLER, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Math. Program., 106 (2006), pp. 25–57.

[52] R. A. WALTZ AND J. NOCEDAL, *KNITRO user's manual*, tech. rep., Optimization Technology Center, Northwestern University, Evanston, IL, 2003.