Algorithm 1035: A Gradient-based Implementation of the Polyhedral Active Set Algorithm

WILLIAM W. HAGER, University of Florida, USA
HONGCHAO ZHANG, Louisiana State University, USA

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.

CCS Concepts: • **Theory of computation** → *Mathematical optimization*; **Nonconvex optimization;**

Additional Key Words and Phrases: Nonlinear optimization, polyhedral-constrained optimization, active set method, gradient projection method, projection on polyhedron, conjugate gradient method, PASA, PPROJ, CG_DESCENT, NAPHEAP

ACM Reference format:
William W. Hager and Hongchao Zhang. 2023. Algorithm 1035: A Gradient-based Implementation of the Polyhedral Active Set Algorithm. *ACM Trans. Math. Softw.* 49, 2, Article 20 (June 2023), 13 pages.
https://doi.org/10.1145/3583559

1 INTRODUCTION

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

$$\min f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{x} \in \Omega, \tag{1}$$

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

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \le \mathbf{b}\}, \tag{2}$$

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

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{bl} \le \mathbf{A}\mathbf{x} \le \mathbf{bu}, \quad \mathbf{lo} \le \mathbf{x} \le \mathbf{hi}\}, \tag{3}$$

The assistance of Nicholas Gould and Dominique Orban in configuring CUTEst to enable its operation with PASA was greatly appreciated. An initial draft of the PASA MATLAB interface by James Diffenderfer is gratefully acknowledged. The authors gratefully acknowledge support by the National Science Foundation under grants 1819002, 1819161, 2031213, and 2110722, and by the Office of Naval Research under grants N00014-15-1-2048, N00014-18-1-2100, and N00014-22-1-2397.
Authors' addresses: W. W. Hager, University of Florida, Department of Mathematics, P.O. Box 118105, Gainesville, FL, 32611-8105, USA; email: hager@ufl.edu; H. Zhang, Louisiana State University, Department of Mathematics, 303 Lockett Hall, Baton Rouge, LA, 70803-4918, USA; email: hozhang@math.lsu.edu.

0098-3500/2023/06-ART20 $15.00
https://doi.org/10.1145/3583559

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 [13, 18–20, 23], Timothy Davis and William Hager developed techniques for modifying a sparse Cholesky factorization of a matrix of the form $\mathbf{AA}^{\mathsf{T}}$ after adding or deleting 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 a series of papers [21, 22, 35–40], 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 [21, 22], it was applied to linear programming using the newly developed update/downdate techniques. More recently, in Reference [48] 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 [41, 42, 45], 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 Reference [46], 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 References [43, 44]. This active set algorithm had two phases: In phase one, the gradient projection algorithm and a cyclic Barzilai/Borwein [4, 17] step was 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 Reference [47] is a generalization of the two-phase algorithm in Reference [43] 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

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

the projection is computed using the Newton/heap-based algorithm NAPHEAP of Reference [24]. 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 [51], NPSOL [29], OPTPACK [33, 36], and LANCELOT [16]. Murtagh and Saunders' MINOS is based on Robinson's algorithm [52], 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 References [49, 50]. 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 [28], which employs active set methods and dense linear algebra to solve constrained linear least-squares problems and convex quadratic programming problems. OPTPACK [33, 36] 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 that is minimized within a region defined by the bound constraints. The bound-constrained problem is solved by an algorithm that combines projected gradient techniques [14] and special structures to exploit the group partially separable structure of a problem [15].

More recently, NPSOL was the starting point for **SNOPT (Sparse Nonlinear Optimizer)** [27] and **DNOPT (Dense Nonlinear Optimizer)** [30]. 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 References [25, 26], 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, with the theoretical basis for the algorithms detailed in the book cited in Reference [5]. 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 [53] (a merit line-search interior point method based on a quadratic program solver also named LOQO), Waltz and Nocedal's KNITRO [55] (an interior point approach based on sequential quadratic programming and trust regions [11, 12]), and Biegler and Wächter's IPOPT [54], 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)**, which is managed by the COIN-OR Foundation.

Performance data often shows IPOPT is among the best-performing NLP solvers. For example, in Reference [6] Birgin and Martínez provide performance data showing that ALGENCAN and IPOPT are competitive with each other. Although ALGENCAN was more robust in the experiments, IPOPT had slightly better CPU time performance on a set of 688 problems from **CUTEst (Constrained and Unconstrained Testing Environment with Safe Threads)** [31], where both codes found equivalent solutions. In Chapter 21 of Reference [2], Andrei observed that for a set of 93 test problems, KNITRO and IPOPT had similar performance in terms of number of iterations [2, Figure 21.1], while KNITRO had somewhat better performance in terms of CPU time [2, Figure 21.2]. In Reference [54], Biegler and Wächter found that an early version of IPOPT had better performance than an early version of KNITRO on a test set consisting of 979 CUTE problems. In Reference [56] Wan and Biegler observed that in a set of 227 problems from CUTEst, KNITRO and IPOPT were very similar in performance. KNITRO was faster than IPOPT before incorporating regularization techniques, but slower after using regularization.

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) to the violation in the local 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

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

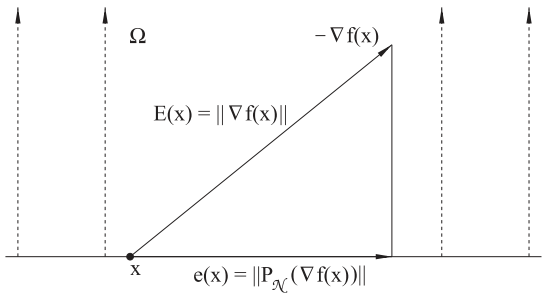

Fig. 1. Global error $E(x)$ versus local error $e(x)$.

where P_Ω denotes the Euclidean norm projection given by

$$P_\Omega(\mathbf{x}) := \arg\min_{\mathbf{y}}\{\|\mathbf{x}-\mathbf{y}\|^2 : \mathbf{y}\in\Omega\}. \tag{4}$$

Recall [43, P7] that $E(\mathbf{x}) = 0$ if and only if $\mathbf{x}$ is a stationary point for the global problem (1). After a change of variables, we obtain

$$E(\mathbf{x}) = \|\mathbf{y}(\mathbf{x})\| \text{ where } \mathbf{y}(\mathbf{x}) = \arg\min_{\mathbf{y}}\{\|\mathbf{y}+\nabla f(\mathbf{x})\| : \mathbf{y}\in\Omega-\mathbf{x}\}. \tag{5}$$

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 (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

$$\min f(\mathbf{z}) \quad \text{subject to} \quad \mathbf{z}\in\mathcal{M}(\mathbf{x}). \tag{6}$$

By Equation (5), with Ω replaced by $\mathcal{M}(\mathbf{x})$, $\mathbf{x}$ is a stationary point for the local problem (6) 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 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 1. As seen in Algorithm 1, the branching between the two phases of PASA is based on a comparison between E and e at the current iterate $\mathbf{x}_k$. It is shown in Reference [47] that

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

whenever the algorithm in phase two satisfies the following conditions:

P1. For each k, $\mathbf{x}_k\in\Omega$ and $f(\mathbf{x}_{k+1}) \le f(\mathbf{x}_k)$.
P2. For each k, $\mathcal{A}(\mathbf{x}_k)\subset\mathcal{A}(\mathbf{x}_{k+1})$.
P3. If $\mathcal{A}(\mathbf{x}_{j+1}) = \mathcal{A}(\mathbf{x}_j)$ for $j\ge k$, then $\liminf_{j\to\infty} e(\mathbf{x}_j) = 0$.

ALGORITHM 1: Sketch of **Polyhedral Active Set Algorithm (PASA)**

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

Moreover, under either a nondegeneracy condition or a strong second-order sufficient optimality condition 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 is made in Reference [47] 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 multiplier is sufficiently positive and the constraint is sufficiently inactive. By the first-order optimality conditions, large multipliers are associated with active constraints. Hence, an inactive constraint and a large multiplier contradict the first-order optimality conditions, which leads us to classify the constraint as undecided.

Determining whether the undecided index set is empty requires an additional projection that detracts from the efficiency of PASA. However, any update to θ in phase one that drives it to zero guarantees that the phase two iterates are asymptotically generated by phase two under either the nondegeneracy condition or the strong second-order sufficient optimality condition. 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 when more than one iteration is performed in phase one.

The rules for branching between phases one and two are based on the following considerations: First note that phase one, by itself, is typically globally convergent, as established in Reference [43, Theorem 2.2]. The convergence rate, however, is at best linear. The purpose of phase two is to improve efficiency by using a superlinearly convergent algorithm to find an optimum over the manifold defined by the active constraints. Nonetheless, efficiency is lost if the optimum over the manifold is computed with too much precision. Since the default value for θ is 0.01, we would branch from phase two to phase one if the local error $e(\mathbf{x}_k)$ is less than 0.01 times the global error $E(\mathbf{x}_k)$. In phase one, typically only one iteration is performed, some constraints that were active become inactive, and the iterate moves to a new manifold. In the previous iteration, the inequality $e(\mathbf{x}_k) < \theta E(\mathbf{x}_k)$ was satisfied. After performing a single gradient projection step in phase one and moving to a new active manifold, we usually find that $e(\mathbf{x}_k) > \theta E(\mathbf{x}_k)$, so PASA branches back to phase two and begins to explore a new active manifold.

4 PHASE ONE

The version of the gradient projection algorithm that we utilize is depicted in Figure 2. 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

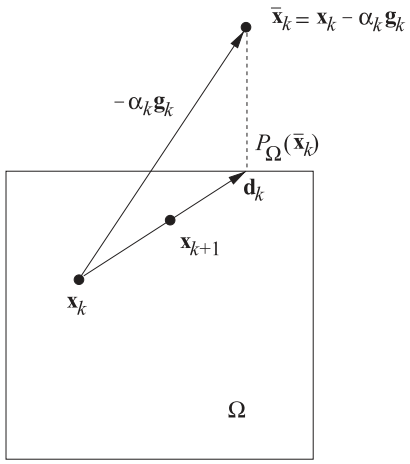

Fig. 2. Sketch of the gradient projection algorithm.

reach a point $\bar{\mathbf{x}}_k$, whose projection onto the polyhedron Ω is $P_\Omega(\bar{\mathbf{x}}_k)$. A line search is performed along the search direction $\mathbf{d}_k = P_\Omega(\bar{\mathbf{x}}_k) - \mathbf{x}_k$ to obtain the next iterate $\mathbf{x}_{k+1}$. Our choice for the step size α_k is based on the BB formula [4] with a cyclic implementation that is explained in Reference [17]. In a cyclic implementation, the step size 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 Reference [17], a cyclic implementation can lead to better performance.

Our version of the gradient projection algorithm is similar to the version in SPG [7], since the line search is along the line segment connecting $\mathbf{x}_k$ to the projection point $P_\Omega(\bar{\mathbf{x}}_k)$. 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 step size 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 References [8–10] and the references therein. For this scheme, the active constraints at a minimizer can be identified in a finite number of iterations under suitable assumptions; however, multiple projections may be needed to determine an acceptable step, unlike the version of the gradient projection scheme used by PASA that only uses one projection in each iteration. Although the gradient projection algorithm implemented in PASA may not identify the active constraints, we show in Reference [47, Lemma 6.2] that the level of constraint inactivity is on the order of the squared error in the iterate $\mathbf{x}_k$.

The line search implemented in PASA, shown in Algorithm 2, is of Armijo type [3]. 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 Reference [32], but with a more sophisticated choice of f_k^r, based on the procedure given in the appendix of Reference [43]. 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 Reference [41].

5 PHASE TWO

According to the theory developed in Reference [47], any algorithm with the properties (P1)–(P3) can be used in phase two. 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

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

ALGORITHM 2: Phase one (gradient projection algorithm)

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

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

$$\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{d}_k, \quad \text{where } \mathbf{d}_k = \mathcal{P}_{\Omega_k}(\mathbf{x}_k - \alpha_k \mathbf{g}_k) - \mathbf{x}_k, \quad 0 < s_k \leq 1.$$

This is the gradient projection step of Algorithm 2 except that Ω is replaced by Ω_k. Since the constraints that are active at $\mathbf{x}_k$ are also active at $\mathcal{P}_{\Omega_k}(\mathbf{x}_k - \alpha_k \mathbf{g}_k)$, all the constraints active at $\mathbf{x}_k$ are also active at $\mathbf{x}_{k+1}$. Potentially, when $s_k = 1$, additional constraints could be active at $\mathbf{x}_{k+1}$. As long as $\mathcal{A}(\mathbf{x}_k)$ is strictly contained in $\mathcal{A}(\mathbf{x}_{k+1})$ and phase two does not reach one of the termination conditions in Algorithm 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 a conjugate gradient scheme. This switch is done to exploit the faster convergence of the conjugate gradient method when compared to gradient descent. When the active set is growing, it is pointless to switch to conjugate gradients, since CG needs to be restarted whenever a new constraint becomes active. Hence, the switch to CG is not attempted until $\mathcal{A}(\mathbf{x}_k) = \mathcal{A}(\mathbf{x}_{k-1})$ in A-GP.

Our implementation of conjugate gradients in phase two is now explained. Let $\mathbf{A}_k$ be the submatrix of $\mathbf{A}$ associated with the active constraint gradients at $\mathbf{x}_k$, and let $\mathbf{b}_k$ denote the associated right side of the constraint. During phase two, the constraint $\mathbf{A}_k\mathbf{x} \leq \mathbf{b}_k$ is enforced as an equality, while the remaining constraints are strict inequalities at $\mathbf{x}_k$. Since $\mathbf{A}_k\mathbf{x}_k = \mathbf{b}_k$, the change of variables $\mathbf{x} = \mathbf{x}_k + \mathbf{z}$ yields the equation $\mathbf{A}_k\mathbf{z} = \mathbf{0}$. After this change of variables, the optimization problem in phase two is written

$$\min f(\mathbf{x}_k + \mathbf{z}) \quad \text{subject to} \quad \mathbf{A}_k\mathbf{z} = \mathbf{0}, \quad \mathbf{z} \in \Omega - \mathbf{x}_k. \tag{7}$$

If $\mathbf{P}_k \in \mathbb{R}^{n\times n}$ denotes the orthogonal projection onto the null space $\mathcal{N}(\mathbf{A}_k)$, then the change of variables $\mathbf{z} = \mathbf{P}_k\mathbf{y}$ in Equation (7) yields the locally unconstrained problem

$$\min f(\mathbf{x}_k + \mathbf{P}_k\mathbf{y}) \quad \text{subject to} \quad \mathbf{x}_k + \mathbf{P}_k\mathbf{y} \in \Omega. \tag{8}$$

This problem is locally unconstrained, since $\mathbf{x}_k + \mathbf{P}_k\mathbf{y}$ lies in Ω for $\mathbf{y}$ near $\mathbf{0}$. Hence, the conjugate gradient algorithm can be applied to Equation (8) generating iterates $\mathbf{y}_j$, $j \geq 0$, starting from $\mathbf{y}_0 = \mathbf{0}$.

Each conjugate gradient iteration involves a search direction $\mathbf{d}_j$ and a line search along $\mathbf{d}_j$. If α is the step size along $\mathbf{d}_j$, the line search enforces the constraint

$$\mathbf{x}_k + \mathbf{P}_k(\mathbf{y}_j + \alpha \mathbf{d}_j) \in \Omega. \tag{9}$$

Assuming $\mathbf{x}_k + \mathbf{P}_k\mathbf{y}$ lies in Ω for $\mathbf{y}$ near $\mathbf{y}_j$, we let $\alpha_{\max}$ denote the largest α such that the inclusion (9) holds. With this notation, the conjugate gradient line search focuses on the optimization problem

$$\min \quad f(\mathbf{x}_k + \mathbf{P}_k(\mathbf{y}_j + \alpha \mathbf{d}_j)) \quad \text{subject to} \quad 0 \leq \alpha \leq \alpha_{\max}.$$

If the solution of this problem is $\alpha = \alpha_{\max}$, then one or more constraints are activated and we return to A-GP. Otherwise, the projected conjugate gradient iteration continues. Phase two is summarized in Algorithm 3.

ALGORITHM 3: Phase two (A-GP and CG_DESCENT)

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 (8); branch to step 1 when reaching $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{P}_k(\mathbf{y}_j + \alpha_{\max}\mathbf{d}_j)$ on the boundary of Ω.

End

A stability issue arises when applying the conjugate gradient method to the unconstrained problem (8). 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 Equation (8) 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 Equation (8) 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 Reference [46, Section 2])

$$\mathbf{d}_0 = -\widetilde{\mathbf{P}}_k^2\mathbf{g}_0, \quad \mathbf{d}_{k+1} = -\widetilde{\mathbf{P}}_k^2\mathbf{g}_{k+1} + \beta_k\mathbf{d}_k \quad \text{for } k \geq 0, \tag{10}$$

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

$$\beta_k = \frac{\mathbf{y}_k^{\mathsf{T}}\widetilde{\mathbf{P}}_k^2\mathbf{g}_{k+1}}{\mathbf{d}_k^{\mathsf{T}}\mathbf{y}_k} - \eta\frac{\|\widetilde{\mathbf{P}}_k\mathbf{y}_k\|^2}{\mathbf{d}_k^{\mathsf{T}}\mathbf{y}_k}\frac{\mathbf{d}_k^{\mathsf{T}}\mathbf{g}_{k+1}}{\mathbf{d}_k^{\mathsf{T}}\mathbf{y}_k}, \quad \mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k. \tag{11}$$

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 Equation (10), 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 Equation (10), and these errors in the search direction can accumulate. The following iteration is equivalent to Equation (10) 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 $\mathbf{A}_k$:

$$\mathbf{D}_0 = -\widetilde{\mathbf{P}}_k\mathbf{g}_0, \quad \mathbf{D}_{k+1} = -\widetilde{\mathbf{P}}_k\mathbf{g}_{k+1} + \beta_k\mathbf{D}_k, \quad \mathbf{d}_{k+1} = \widetilde{\mathbf{P}}_k\mathbf{D}_{k+1} \text{ for } k \geq 0. \tag{12}$$

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 [31] 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 when 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 $\mathbf{A}\mathbf{A}^{\mathsf{T}}$ 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 [34]. We have not yet had time to incorporate Woodbury updates in PASA. Moreover, if the Woodbury updates were incorporated in the code, then termination may occur at the starting point, and the problem would be excluded by the rules given in the next paragraph.

After these exclusions, we start with 655 problems that we tried to solve to the accuracy tolerance 1.e−6. If the objective values computed by each solver agreed to four significant digits, then we accepted the problem. If four-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 four significant digits but the solvers were converging to the same solution, we then adjusted the accuracy tolerance of the less accurate solver 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 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 and gradient) when compared to the

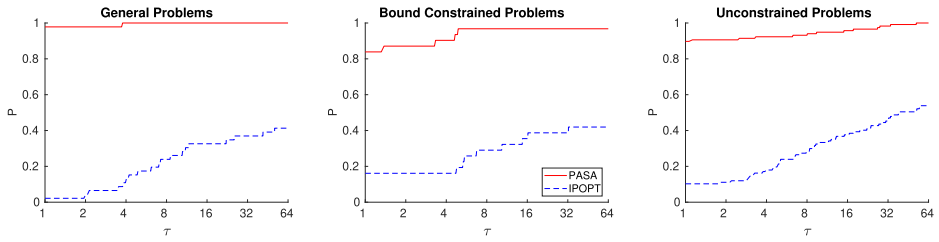

Fig. 3. Wall time performance profiles for general, bound, and unconstrained programs.

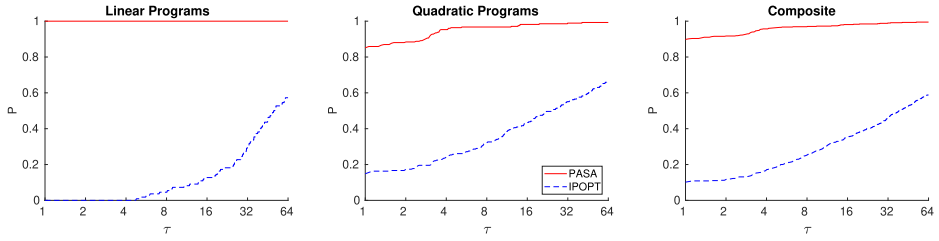

Fig. 4. Wall time performance profiles for linear, quadratic, and composite programs.

gradient-based PASA. The tradeoffs between convergence rate of an algorithm and evaluation time are not studied in this article; 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 eight Intel i7-865U CPUs operating at 1.90 GHz (four 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 3 and 4 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 five 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 five 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 [46], Version 8.0.0. 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 [43] for bound-constrained problems. Hence, the performance on bound-constrained problem essentially reflects the performance of the active set algorithm [43]. Linear programs are solved in PASA by a series of gradient projection steps, where the step size choice is crucial. The performance corresponds to the first-order algorithm in Reference [22]. 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] Roberto Andreani, Ernesto G. Birgin, J. Mario Martínez, and Maria L. Schuverdt. 2007. On augmented Lagrangian methods with general lower-level constraints. *SIAM J. Optim.* 18, 4 (2007), 1286–1309.

[2] Neculai Andrei. 2017. *Continuous Nonlinear Optimization for Engineering Applications in GAMS Technology*. Springer.

[3] Larry Armijo. 1966. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacif. J. Math.* 16 (1966), 1–3.

[4] Jonathan Barzilai and Jonathan M. Borwein. 1988. Two point step size gradient methods. *IMA J. Numer. Anal.* 8 (1988), 141–148.

[5] Ernesto G. Birgin and J. Mario Martínez. 2014. *Practical Augmented Lagrangian Methods for Constrained Optimization.* Society for Industrial and Applied Mathematics, Philadelphia, PA.

[6] Ernesto G. Birgin and J. Mario Martínez. 2020. Complexity and performance of an augmented Lagrangian algorithm. *Optim. Meth. Softw.* 35 (2020), 885–920.

[7] Ernesto G. Birgin, J. Mario Martínez, and Marcos Raydan. 2001. Algorithm 813: SPG—Software for convex-constrained optimization. *ACM Trans. Math. Softw.* 27 (2001), 340–349.

[8] James V. Burke and Jorge J. Moré. 1988. On the identification of active constraints. *SIAM J. Numer. Anal.* 25 (1988), 1197–1211.

[9] James V. Burke and Jorge J. Moré. 1994. Exposing constraints. *SIAM J. Optim.* 25 (1994), 573–595.

[10] James V. Burke, Jorge J. Moré, and Gerardo Toraldo. 1990. Convergence properties of trust region methods for linear and convex constraints. *Math. Progam.* 47 (1990), 305–336.

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

[12] Richard H. Byrd, Jorge Nocedal, and Richard A. Waltz. 2006. KNITRO: An integrated package for nonlinear optimization. In *Large-scale Nonlinear Optimization*, G. di Pillo and M. Roma (Eds.). Springer-Verlag, 35–59.

[13] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. 2009. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans. Math. Softw.* 35, 3 (2009), 22:1–14.

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

[15] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. 1990. 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.). SIAM, Philadelphia, PA, 42–54.
[16] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. 1992. *LANCELOT: A Fortran Package for Large-scale Nonlinear Optimization (Release A)*. Springer.
[17] Yu-Hong Dai, William W. Hager, Klaus Schittkowski, and Hongchao Zhang. 2006. The cyclic Barzilai-Borwein method for unconstrained optimization. *IMA J. Numer. Anal.* 26 (2006), 604–627.
[18] Timothy A. Davis and William W. Hager. 1999. Modifying a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.* 20, 3 (1999), 606–627.
[19] Timothy A. Davis and William W. Hager. 2001. Multiple-rank modifications of a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.* 22 (2001), 997–1013.
[20] Timothy A. Davis and William W. Hager. 2005. Row modifications of a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.* 26, 3 (2005), 621–639.
[21] Timothy A. Davis and William W. Hager. 2008. Dual multilevel optimization. *Math. Program.* 112, 2 (Apr. 2008), 403–425.
[22] Timothy A. Davis and William W. Hager. 2008. A sparse proximal implementation of the LP dual active set algorithm. *Math. Program.* 112, 2 (Apr. 2008), 275–301.
[23] Timothy A. Davis and William W. Hager. 2009. Dynamic supernodes in sparse Cholesky update/downdate and triangular solves. *ACM Trans. Math. Softw.* 35, 4 (2009), 27:1–23.
[24] Timothy A. Davis, William W. Hager, and James T. Hungerford. 2016. An efficient hybrid algorithm for the separable convex quadratic knapsack problem. *ACM Trans. Math. Softw.* 42 (2016), 22:1–22:25.
[25] Roger Fletcher and Sven Leyffer. 2002. Nonlinear programming without a penalty function. *Math. Program.* 91 (2002), 239–270.
[26] Roger Fletcher, Sven Leyffer, and Philippe L. Toint. 2002. On the global convergence of a filter-SQP algorithm. *SIAM J. Optim.* 13 (2002), 44–59.
[27] Philip E. Gill, Walter Murray, and Michael A. Saunders. 2005. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev.* 47 (2005), 99–131.
[28] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. 1986. *User's Guide for LSSOL (Version 1.0)*. Technical Report. Report No. 86-1. Stanford University, Department of Operations Research, Stanford, CA.
[29] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. 1986. *User's Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming*. Technical Report. Report No. 86-2. Stanford University, Department of Operations Research, Stanford, CA.
[30] Philip E. Gill, Michael A. Saunders, and Elizabeth Wong. 2016. *An SQP Method for Medium-scale Nonlinear Programming*. Technical Report CCoM 16-2. Center for Computational Mathematics, Department of Mathematics, University of California, La Jolla, CA.
[31] Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. 2015. CUTEst: A constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.* 60 (2015), 545–557.
[32] Luigi Grippo, Francesco Lampariello, and Stefano Lucidi. 1986. A nonmonotone line search technique for Newton's method. *SIAM J. Numer. Anal.* 23 (1986), 707–716.
[33] William W. Hager. 1987. Dual techniques for constrained optimization. *J. Optim. Theory Appl.* 55 (1987), 37–71.
[34] William W. Hager. 1989. Updating the inverse of a matrix. *SIAM Rev.* 31, 2 (1989), 221–239.
[35] William W. Hager. 1992. The dual active set algorithm. In *Advances in Optimization and Parallel Computing*, P. M. Pardalos (Ed.). North Holland, Amsterdam, 137–142.
[36] William W. Hager. 1993. Analysis and implementation of a dual algorithm for constrained optimization. *J. Optim. Theor. Appl.* 79 (1993), 427–462.
[37] William W. Hager. 1998. The LP dual active set algorithm. In *High Performance Algorithms and Software in Nonlinear Optimization*, R. De Leone, A. Murli, P. M. Pardalos, and G. Toraldo (Eds.). Kluwer, Dordrecht, 243–254.
[38] William W. Hager. 2002. The dual active set algorithm and its application to linear programming. *Comput. Optim. Appl.* 21 (2002), 263–275.
[39] William W. Hager. 2003. The dual active set algorithm and the iterative solution of linear programs. In *Novel Approaches to Hard Discrete Optimization*, Vol. 37, P. M. Pardalos and H. Wolkowicz (Eds.). Fields Institute Communications, 95–107.
[40] William W. Hager and Donald W. Hearn. 1993. Application of the dual active set algorithm to quadratic network optimization. *Comput. Optim. Appl.* 1 (1993), 349–373.
[41] William W. Hager and Hongchao Zhang. 2005. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM J. Optim.* 16 (2005), 170–192.

[42] William W. Hager and Hongchao Zhang. 2006. Algorithm 851: CG_DESCENT, A conjugate gradient method with guaranteed descent. *ACM Trans. Math. Softw.* 32 (2006), 113–137.
[43] William W. Hager and Hongchao Zhang. 2006. A new active set algorithm for box constrained optimization. *SIAM J. Optim.* 17 (2006), 526–557.
[44] William W. Hager and Hongchao Zhang. 2006. 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, 67–82.
[45] William W. Hager and Hongchao Zhang. 2006. A survey of nonlinear conjugate gradient methods. *Pacif. J. Optim.* 2 (2006), 35–58.
[46] William W. Hager and Hongchao Zhang. 2013. The limited memory conjugate gradient method. *SIAM J. Optim.* 23 (2013), 2150–2168.
[47] William W. Hager and Hongchao Zhang. 2016. An active set algorithm for nonlinear optimization with polyhedral constraints. *Sci. China Math.* 59 (2016), 1525–1542.
[48] William W. Hager and Hongchao Zhang. 2016. Projection onto a polyhedron that exploits sparsity. *SIAM J. Optim.* 29 (2016), 1773–1798.
[49] Bruce A. Murtagh and Michael A. Saunders. 1978. Large-scale linearly constrained optimization. *Math. Progam.* 14 (1978), 41–72.
[50] Bruce A. Murtagh and Michael A. Saunders. 1982. A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. *Math. Progam. Stud.* 16 (1982), 84–117.
[51] Bruce A. Murtagh and Michael A. Saunders. 1987. *MINOS 5.0: User's Guide.* Technical Report. Report No. 83-20R. Stanford University, Department of Operations Research, Stanford, CA.
[52] Stephen M. Robinson. 1972. A quadratically-convergent algorithm for general nonlinear programming problems. *Math. Progam.* 3 (1972), 145–156.
[53] Robert J. Vanderbei and David F. Shanno. 1999. An interior-point algorithm for nonconvex nonlinear programming. *Comput. Optim. Appl.* 13 (1999), 231–252.
[54] Andreas Wächter and Lorenz T. Biegler. 2006. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Program.* 106 (2006), 25–57.
[55] Richard A. Waltz and Jorge Nocedal. 2003. *KNITRO user's manual.* Technical Report. Optimization Technology Center, Northwestern University, Evanston, IL.
[56] Wei Wan and Lorenz T. Biegler. 2017. Structured regularization for barrier NLP solvers. *Comput. Optim. Appl.* 66 (2017), 401–424.

Received 17 February 2022; revised 4 August 2022; accepted 28 November 2022