Fields Institute Communications
Volume **37**, 2003

The Dual Active Set Algorithm and the Iterative Solution of Linear Programs

William W. Hager
Department of Mathematics
University of Florida
Gainesville, FL 32611
hager@math.ufl.edu
http://www.math.ufl.edu/∼hager

Abstract. In this paper, we examine an iterative implementation of the Dual Active Set Algorithm, focusing on its application to linear programs such as those associated with lower bounds to the quadratic assignment problem.

1 Introduction

In [9, 10, 12, 13, 14] we present the Dual Active Set Algorithm (DASA) and prove its convergence both for linear programming and for optimization problems satisfying a strict convexity assumption. In [14] a local quadratic convergence result for a "full step" version of this algorithm is given in the context of optimal control, while line search versions of the algorithm for general optimization problems appear in [9, 10, 12, 13]. Also, in [2] Bergounioux and Kunisch establish convergence for a full step version in quadratic optimization problems where the matrix in the cost function has a diagonal dominance property.

In this paper, we examine iterative implementations of the DASA. In the earlier work, it was assumed that the subproblems were solved exactly in each step, in which case convergence to an optimal solution is obtained in a finite number of steps. In an iterative implementation, convergence is achieved in the limit. We illustrate the convergence properties of the iterative schemes using linear programming (LP) test problems, focusing in particular on LPs associated with lower bounds for the quadratic assignment problem (see [6, 15, 21]). The lack of strict convexity in the LP setting is handled using proximal techniques.

Earlier research on iterative methods for the solution of linear programming problems using proximal techniques includes that of [5, 18, 25]. Some success has been reported with these iterative methods when applied to randomly generated

2000 *Mathematics Subject Classification.* Primary 90C05, 90C25, 90C49; Secondary 65K05, 65F10.

NSF support gratefully acknowledged.

problems. These methods did not perform as well when applied to test problems arising in applications, such as those contained in the Netlib test suite. In our iterative approach, we employ an SSOR preconditioned conjugate gradient iteration which we show in [11] correctly handles the singularities often present in LPs arising in practice. By utilizing this iterative solver within the DASA, we are able to handle the changes in active set in a way similar to that of the DASA algorithm itself, for which finite convergence holds.

Even though convergence is not finite for the iterative version of DASA, the algorithm is rather efficient in some cases. For LPs with a sparse matrix and sparse factorization, a factorization-based approach such as the simplex method or an interior point method or a multilevel LPDASA should be preferable. On the other hand, for a sparse LP whose factorization is relatively dense, an iterative approach can be relatively efficient. In this case, multiplying a vector by the matrix is much faster than either factoring the matrix or solving a factored system.

Another iterative approach [21], which has been applied with success to the linear programs associated with the quadratic assignment problem, involves using a preconditioned conjugate gradient method to evaluate the search directions generated by an interior point method. Note though that this preconditioning technique involves a partial factorization of the interior point matrix, with an increasing number of nonzero elements in the preconditioner as the iterations converge. The storage associated with our iterative approach is proportional to the number of nonzeros in the original matrix, not its factorization.

2 DASA with exact subproblems

We first summarize the statement of DASA as given (for example) in [12] for problems of the form:

$$\max_{\boldsymbol{\lambda}} \ \min_{\mathbf{x}\geq\mathbf{0}} \ \mathcal{L}(\boldsymbol{\lambda},\mathbf{x}), \tag{2.1}$$

where $\mathcal{L} : \mathbf{R}^m \times \mathbf{R}^n \to \mathbf{R}$. We assume that $\mathcal{L}(\boldsymbol{\lambda},\mathbf{x})$ is concave in $\boldsymbol{\lambda}$ for each fixed $\mathbf{x} \in \mathbf{R}^n$, uniformly strongly convex in $\mathbf{x}$ for each fixed $\boldsymbol{\lambda} \in \mathbf{R}^m$, and continuously differentiable. By strong convexity in $\mathbf{x}$, we mean that there exists a constant $\alpha > 0$ such that

$$\left(\nabla_x\mathcal{L}(\boldsymbol{\lambda},\mathbf{y}) - \nabla_x\mathcal{L}(\boldsymbol{\lambda},\mathbf{x})\right)(\mathbf{y}-\mathbf{x}) \geq \alpha\|\mathbf{y}-\mathbf{x}\|^2,$$

where α is independent of $\boldsymbol{\lambda}$, $\mathbf{x}$, and $\mathbf{y}$, and $\|\cdot\|$ denotes the Euclidean norm.

If $B \subset \{1,2,\ldots,n\}$, let $\mathbf{x}_B$ be the subvector of $\mathbf{x}$ consisting of those components x_i associated with $i \in B$. Two different functions enter into the statement of the DASA:

$$\mathcal{L}_B(\boldsymbol{\lambda}) = \min_{\mathbf{x}_B\geq\mathbf{0}} \mathcal{L}(\boldsymbol{\lambda},\mathbf{x}) \quad \text{and} \quad \mathcal{L}_B^0(\boldsymbol{\lambda}) = \min_{\mathbf{x}_B=\mathbf{0}} \mathcal{L}(\boldsymbol{\lambda},\mathbf{x}). \tag{2.2}$$

In carrying out the minimizations in (2.2), the components of $\mathbf{x}$ corresponding to indices in the complement of B are unconstrained. By the strong convexity of $\mathcal{L}(\boldsymbol{\lambda},\cdot)$, there exists a unique minimizer $\mathbf{x}(\boldsymbol{\lambda},B)$ over the set $\mathbf{x}_B \geq \mathbf{0}$, for each choice of $\boldsymbol{\lambda}$ and B. Since $\mathcal{L}(\boldsymbol{\lambda},\mathbf{x})$ is strongly convex in $\mathbf{x}$ and continuously differentiable, $\mathbf{x}(\boldsymbol{\lambda},B)$ depends continuously on $\boldsymbol{\lambda}$ [12]. The unique minimizer of (2.1) corresponding to $B = \{1,2,\ldots,n\}$ in (2.2) is denoted $\mathbf{x}(\boldsymbol{\lambda})$. We let $\mathcal{L}(\boldsymbol{\lambda})$ denote the dual function $\mathcal{L}(\boldsymbol{\lambda},\mathbf{x}(\boldsymbol{\lambda}))$:

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathcal{L}(\boldsymbol{\lambda},\mathbf{x}(\boldsymbol{\lambda})) = \min_{\mathbf{x}\geq\mathbf{0}} \ \mathcal{L}(\boldsymbol{\lambda},\mathbf{x}).$$

In the DASA, we start from an arbitrary $\boldsymbol{\lambda}_0$ and generate a finite sequence of iterates. If $\boldsymbol{\lambda}_k$ denotes the current iterate (initially $k = 0$), then either $\boldsymbol{\lambda}_k$ maximizes the dual function and we stop, or we move to the next iterate $\boldsymbol{\lambda}_{k+1}$ using a finite sequence of subiterates $\boldsymbol{\nu}_0 = \boldsymbol{\lambda}_k$, $\boldsymbol{\nu}_1$, $\boldsymbol{\nu}_2, \ldots$. The algorithmic steps are the following:

Dual Active Set Algorithm (with line search)

- **Convergence test:** If $\boldsymbol{\lambda}_k$ maximizes the dual function, then stop.
- **Dual initialization:** Set $j = 0$, $\boldsymbol{\nu}_0 = \boldsymbol{\lambda}_k$, $B_0 = \{i : x_i(\boldsymbol{\lambda}_k) = 0\}$.
- **Dual subiteration:**

$$\boldsymbol{\omega}_j \in \arg \max_{\boldsymbol{\lambda}} \mathcal{L}^0_{B_j}(\boldsymbol{\lambda}) \quad \text{and} \quad \boldsymbol{\nu}_{j+1} \in \arg \max_{\boldsymbol{\lambda} \in [\boldsymbol{\nu}_j, \boldsymbol{\omega}_j]} \mathcal{L}_{B_j}(\boldsymbol{\lambda}).$$

  If there are multiple maxima on the line segment $[\boldsymbol{\nu}_j, \boldsymbol{\omega}_j]$ connecting $\boldsymbol{\nu}_j$ and $\boldsymbol{\omega}_j$, then $\boldsymbol{\nu}_{j+1}$ should be the point closest to $\boldsymbol{\nu}_j$.
- **Constraint deletion:**

$$B_{j+1} = \{i \in B_j : y_i = 0,\ (\nabla_x \mathcal{L}(\boldsymbol{\nu}_{j+1}, \mathbf{y}))_i > 0\} \ \text{ where } \mathbf{y} = \mathbf{x}(\boldsymbol{\nu}_{j+1}, B_j).$$

- **Stopping criterion:** If $\mathcal{L}_{B_j}(\boldsymbol{\nu}_{j+1}) = \mathcal{L}^0_{B_j}(\boldsymbol{\omega}_j)$, then increment k, set $\boldsymbol{\lambda}_k = \boldsymbol{\nu}_{j+1}$, and go to convergence test. Otherwise, increment j and continue the dual subiteration.

In [12] we prove the following:

Theorem 2.1 *Assume that $\mathcal{L}(\boldsymbol{\lambda}, \mathbf{x})$ is uniformly strongly convex in $\mathbf{x}$ for each fixed $\boldsymbol{\lambda} \in \mathbf{R}^m$, concave in $\boldsymbol{\lambda}$ for each fixed $\mathbf{x} \in \mathbf{R}^n$, and continuously differentiable. If in each step of the DASA, a maximizer $\boldsymbol{\omega}_j$ in the dual subiteration exists, then the DASA generates a solution of* (2.1) *in a finite number of iterations.*

The proof in [12] reveals that for each j, B_{j+1} is strictly contained in B_j until the stopping criterion is satisfied.

3 DASA with inexact subproblems

In the previous section, we assumed the subproblems in the DASA were solved exactly, and we obtained convergence in a finite number of steps. In an iterative implementation, the solution of the subproblems are approximated, and convergence occurs in the limit (possibly an infinite number of steps). One approach for achieving guaranteed (monotone) convergence is to begin the dual subiteration with a steepest ascent step, and in subsequent subiterations, compute a maximizer of $\mathcal{L}^0_{B_j}(\boldsymbol{\lambda})$ with sufficient accuracy to ensure ascent. More precisely, if $\mathcal{R}(\boldsymbol{\lambda}, \mathbf{g})$ denotes the ray emanating from $\boldsymbol{\lambda}$ in the direction $\mathbf{g}$:

$$\mathcal{R}(\boldsymbol{\lambda}, \mathbf{g}) = \{\boldsymbol{\lambda} + \alpha \mathbf{g} : \alpha \ge 0\},$$

then the definitions of $\boldsymbol{\omega}_j$ and $\boldsymbol{\nu}_{j+1}$ in the dual subiteration should be revised to the following:

$$\text{Case } j = 0: \qquad \boldsymbol{\nu}_1 \in \arg \max_{\boldsymbol{\lambda} \in \mathcal{R}(\boldsymbol{\lambda}_k, \nabla \mathcal{L}(\boldsymbol{\lambda}_k))} \mathcal{L}_{B_0}(\boldsymbol{\lambda}) \tag{3.1}$$

$$\text{Case } j > 0: \qquad \text{Find } \boldsymbol{\omega}_j \text{ such that } \mathcal{L}^0_{B_j}(\boldsymbol{\omega}_j) \ge \mathcal{L}^0_{B_j}(\boldsymbol{\nu}_j)$$

$$\text{and choose} \quad \boldsymbol{\nu}_{j+1} \in \arg \max_{\boldsymbol{\lambda} \in [\boldsymbol{\nu}_j, \boldsymbol{\omega}_j]} \mathcal{L}_{B_j}(\boldsymbol{\lambda}) \tag{3.2}$$

By [4, Theorem 2.1], $\nabla \mathcal{L}(\boldsymbol{\lambda}_k) = \nabla_\lambda \mathcal{L}(\boldsymbol{\lambda}_k, \mathbf{x})$ evaluated at $\mathbf{x} = \mathbf{x}(\boldsymbol{\lambda}_k)$.

The stopping criterion in the DASA is expressed in terms of the maximizer $\boldsymbol{\omega}_j$ of $\mathcal{L}_{B_j}$. In the iterative version of DASA, we only approximate $\boldsymbol{\omega}_j$; since $\boldsymbol{\omega}_j$ is not available, a different stopping criterion is needed. We have found that the following criterion works well in practice: Given a fixed scalar $\theta \in (0,1)$, stop when

$$\|\nabla \mathcal{L}_{B_j}(\boldsymbol{\nu}_{j+1})\| \le \theta \|\nabla \mathcal{L}(\boldsymbol{\nu}_{j+1})\|. \tag{3.3}$$

By the "iterative DASA" we mean DASA with the dual subiteration replaced by (3.1)–(3.2) and with stopping criterion replaced by (3.3). In more detail, the algorithm is the following:

Iterative Dual Active Set Algorithm (with line search)

- **Convergence test:** If $\|\nabla \mathcal{L}(\boldsymbol{\lambda}_k)\|$ is sufficiently small, then stop.
- **Dual initialization:** Set $j = 0$, $\boldsymbol{\nu}_0 = \boldsymbol{\lambda}_k$, $B_0 = \{i : x_i(\boldsymbol{\lambda}_k) = 0\}$.
- **Dual subiteration:**

$$\text{Case } j = 0: \qquad \boldsymbol{\nu}_1 \in \arg \max_{\boldsymbol{\lambda} \in \mathcal{R}(\boldsymbol{\lambda}_k, \nabla \mathcal{L}(\boldsymbol{\lambda}_k))} \mathcal{L}_{B_0}(\boldsymbol{\lambda})$$

$$\text{Case } j > 0: \qquad \text{Find } \boldsymbol{\omega}_j \text{ such that } \mathcal{L}^0_{B_j}(\boldsymbol{\omega}_j) \ge \mathcal{L}^0_{B_j}(\boldsymbol{\nu}_j)$$

$$\text{and choose} \quad \boldsymbol{\nu}_{j+1} \in \arg \max_{\boldsymbol{\lambda} \in [\boldsymbol{\nu}_j, \boldsymbol{\omega}_j]} \mathcal{L}_{B_j}(\boldsymbol{\lambda})$$

- **Constraint deletion:**

$$B_{j+1} = \{i \in B_j : y_i = 0,\ (\nabla_x \mathcal{L}(\boldsymbol{\nu}_{j+1}, \mathbf{y}))_i > 0\} \text{ where } \mathbf{y} = \mathbf{x}(\boldsymbol{\nu}_{j+1}, B_j).$$

- **Stopping criterion:** If $\|\nabla \mathcal{L}_{B_j}(\boldsymbol{\nu}_{j+1})\| \le \theta \|\nabla \mathcal{L}(\boldsymbol{\nu}_{j+1})\|$, go to convergence test. Otherwise, increment j and continue the dual subiteration.

If the $\boldsymbol{\omega}_j$ in the dual subiteration are generated by steepest ascent or conjugate gradients or preconditioned conjugate gradients, the gradients $\nabla \mathcal{L}_{B_j}(\boldsymbol{\nu}_j)$ tend to zero, as j tends to infinity, and the stopping criterion is eventually satisfied (except when $\nabla \mathcal{L}(\boldsymbol{\nu}_j)$ tends to zero, in which case the iterates approach a maximizer of $\mathcal{L}$). We assume that the iterative method used in the dual subiteration has the property that the gradients $\nabla \mathcal{L}_{B_j}(\boldsymbol{\nu}_j)$ tend to zero as j tends to infinity.

Theorem 3.1 *Assume that $\mathcal{L}(\boldsymbol{\lambda}, \mathbf{x})$ is uniformly strongly convex in $\mathbf{x}$ for each fixed $\boldsymbol{\lambda} \in \mathbf{R}^m$, concave in $\boldsymbol{\lambda}$ for each fixed $\mathbf{x} \in \mathbf{R}^n$, and twice continuously differentiable in $\boldsymbol{\lambda}$ and $\mathbf{x}$. If for some iterate $\boldsymbol{\lambda}_k$, the associated level set $\mathcal{S}$ of the dual function is compact, then all the succeeding iterates of the iterative DASA are contained in $\mathcal{S}$ and each convergent subsequence converges to a solution of* (2.1).

Proof Without loss of generality, suppose that $\mathcal{S}$ is the level set associated with the starting guess $\boldsymbol{\lambda}_0$. We divide the proof into 8 steps.

I. *Boundedness of* $\mathbf{x}(\boldsymbol{\lambda})$, $\boldsymbol{\lambda} \in \mathcal{S}$:
Given $\boldsymbol{\pi} \in \mathbf{R}^n$, let $\mathbf{z} = \mathbf{x}(\boldsymbol{\pi})$ denote the associated minimizer in the dual function:

$$\mathcal{L}(\boldsymbol{\pi}, \mathbf{z}) = \min_{\mathbf{x} \ge \mathbf{0}} \mathcal{L}(\boldsymbol{\pi}, \mathbf{x}),$$

and let $\mathbf{x}_0 = \mathbf{x}(\boldsymbol{\lambda}_0)$ denote the dual function minimizer associated with $\boldsymbol{\lambda}_0$. Since the constraint $\mathbf{x} \ge \mathbf{0}$ defines a convex set, the first-order optimality conditions for $\mathbf{z}$ and $\mathbf{x}_0$ yield:

$$\nabla_x \mathcal{L}(\boldsymbol{\pi}, \mathbf{z})(\mathbf{x}_0 - \mathbf{z}) \ge 0 \quad \text{and} \quad \nabla_x \mathcal{L}(\boldsymbol{\lambda}_0, \mathbf{x}_0)(\mathbf{z} - \mathbf{x}_0) \ge 0$$

After adding and rearranging these inequalities, and utilizing the strong convexity assumption, we obtain:

$$\alpha\|\mathbf{z}-\mathbf{x}_0\| \le \|\nabla_x\mathcal{L}(\boldsymbol{\pi},\mathbf{x}_0)-\nabla_x\mathcal{L}(\boldsymbol{\lambda}_0,\mathbf{x}_0)\|. \tag{3.4}$$

Since $\nabla_x\mathcal{L}(\boldsymbol{\lambda},\mathbf{x}_0)$ is a continuous function of $\boldsymbol{\lambda}$, it follows from (3.4) that $\mathbf{z}$ is uniformly bounded, independent of the choice of $\boldsymbol{\pi}\in\mathcal{S}$.

II. *Continuity of the dual function and Lipschitz continuity on $\mathcal{S}$*:
Given $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2\in\mathcal{S}$, let $\mathbf{z}_i=\mathbf{x}(\boldsymbol{\pi}_i)$, $i=1,2$, denote the associated minimizers in the dual function. Similar to (3.4), we have

$$\alpha\|\mathbf{z}_1-\mathbf{z}_2\| \le \|\nabla_x\mathcal{L}(\boldsymbol{\pi}_1,\mathbf{z}_2)-\nabla_x\mathcal{L}(\boldsymbol{\pi}_2,\mathbf{z}_2)\|. \tag{3.5}$$

Since $\mathcal{L}$ is twice continuously differentiable, it follows that the minimizer $\mathbf{x}(\boldsymbol{\pi})$ is a continuous function of $\boldsymbol{\pi}$, and hence, the dual function $\mathcal{L}(\boldsymbol{\pi})=\mathcal{L}(\boldsymbol{\pi},\mathbf{x}(\boldsymbol{\pi}))$ is a continuous function of $\boldsymbol{\pi}$. If $\boldsymbol{\pi}\in\mathcal{S}$, then $\mathbf{z}_2$ lies in a bounded set by step I; consequently, by (3.5) there exists a constant c, independent of $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2\in\mathcal{S}$, such that

$$\|\mathbf{z}_1-\mathbf{z}_2\| \le c\|\boldsymbol{\pi}_1-\boldsymbol{\pi}_2\|. \tag{3.6}$$

III. *Lipschitz continuity of the dual gradient*:
Since $\nabla\mathcal{L}(\boldsymbol{\lambda})=\nabla_\lambda\mathcal{L}(\boldsymbol{\lambda},\mathbf{x})$ evaluated at $\mathbf{x}=\mathbf{x}(\boldsymbol{\lambda})$ [4, Theorem 2.1], we have

$$\nabla\mathcal{L}(\boldsymbol{\pi}_1)-\nabla\mathcal{L}(\boldsymbol{\pi}_2)=\nabla_\lambda\mathcal{L}(\boldsymbol{\pi}_1,\mathbf{z}_1)-\nabla_\lambda\mathcal{L}(\boldsymbol{\pi}_2,\mathbf{z}_2).$$

Utilizing the Lipschitz estimate (3.6) and the fact that $\mathcal{L}(\boldsymbol{\lambda},\mathbf{x})$ is twice continuously differentiable in $\boldsymbol{\lambda}$ and $\mathbf{x}$, we conclude that there exists a constant c, independent of $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2\in\mathcal{S}$, such that

$$\|\nabla\mathcal{L}(\boldsymbol{\pi}_1)-\nabla\mathcal{L}(\boldsymbol{\pi}_2)\| \le c\|\boldsymbol{\pi}_1-\boldsymbol{\pi}_2\|.$$

In a similar fashion, $\nabla\mathcal{L}_B$ is Lipschitz continuous over $\mathcal{S}$ for any choice of B. Since B is a subset of a finite set, there are a finite number of choices for B, and a constant c_0 can be chosen so that

$$\|\nabla\mathcal{L}_B(\boldsymbol{\pi}_1)-\nabla\mathcal{L}_B(\boldsymbol{\pi}_2)\| \le c_0\|\boldsymbol{\pi}_1-\boldsymbol{\pi}_2\|$$

for all $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2\in\mathcal{S}$, and for all choices of $B\subset\{1,2,\ldots,n\}$.

IV. *Monotonicity of the iterates*:
This property follows by exactly the same analysis used to establish monotonicity in the DASA itself. The analysis is repeated for completeness: Recall that $\mathbf{x}_k=\mathbf{x}(\boldsymbol{\lambda}_k)$ is the solution to the problem

$$\min_{\mathbf{x}\ge\mathbf{0}}\ \mathcal{L}(\boldsymbol{\lambda}_k,\ \mathbf{x}).$$

Since the first-order optimality conditions are both necessary and sufficient for optimality when $\mathcal{L}(\boldsymbol{\lambda},\cdot)$ is convex (see [17, Chap. 7]), we have

$$\mathcal{L}(\boldsymbol{\lambda}_k)=\mathcal{L}(\boldsymbol{\lambda}_k,\mathbf{x}_k)=\mathcal{L}_{B_0}(\boldsymbol{\lambda}_k)=\mathcal{L}_{B_0}(\boldsymbol{\nu}_0). \tag{3.7}$$

For the same reason, we have

$$\mathcal{L}_{B_j}(\boldsymbol{\nu}_{j+1})=\mathcal{L}_{B_{j+1}}(\boldsymbol{\nu}_{j+1}) \tag{3.8}$$

for each $j\ge 0$. Since $\boldsymbol{\nu}_{j+1}$ is obtained from a line search, $\mathcal{L}_{B_j}(\boldsymbol{\nu}_j)\le\mathcal{L}_{B_j}(\boldsymbol{\nu}_{j+1})$. Combining this with (3.7) and (3.8) gives

$$\mathcal{L}(\boldsymbol{\lambda}_k)\le\mathcal{L}_{B_j}(\boldsymbol{\nu}_j)\le\mathcal{L}_{B_j}(\boldsymbol{\nu}_{j+1})=\mathcal{L}_{B_{j+1}}(\boldsymbol{\nu}_{j+1}) \tag{3.9}$$

for each $j \geq 0$. This implies that

$$\mathcal{L}(\boldsymbol{\lambda}_k) \leq \mathcal{L}_{C_k}(\boldsymbol{\lambda}_{k+1}) \leq \mathcal{L}(\boldsymbol{\lambda}_{k+1}), \tag{3.10}$$

where C_k is the final set B_j in the subiteration. The final inequality in (3.10) is due to the fact that the optimization problem associated with the evaluation of $\mathcal{L}(\boldsymbol{\lambda}_{k+1})$ involves more constraints than the corresponding optimization problem for $\mathcal{L}_{C_k}(\boldsymbol{\lambda}_{k+1})$.

V. *Convexity of $\mathcal{S}$ and concavity of the dual function*:
Since $\mathcal{L}(\boldsymbol{\lambda}, \mathbf{x})$ is concave in $\boldsymbol{\lambda}$ for each fixed $\mathbf{x}$, we have, for each $\theta \in [0,1]$ and $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2 \in \mathbf{R}^m$,

$$\begin{aligned} \mathcal{L}(\theta\boldsymbol{\pi}_1 + (1-\theta)\boldsymbol{\pi}_2) &= \mathcal{L}(\theta\boldsymbol{\pi}_1 + (1-\theta)\boldsymbol{\pi}_2, \mathbf{x}_\theta) \\ &\geq \theta\mathcal{L}(\boldsymbol{\pi}_1, \mathbf{x}_\theta) + (1-\theta)\mathcal{L}(\boldsymbol{\pi}_2, \mathbf{x}_\theta), \end{aligned} \tag{3.11}$$

where $\mathbf{x}_\theta = \mathbf{x}(\theta\boldsymbol{\pi}_1 + (1-\theta)\boldsymbol{\pi}_2)$. Since $\mathbf{x} = \mathbf{x}(\boldsymbol{\pi})$ is the minimizer of the Lagrangian $\mathcal{L}(\boldsymbol{\pi}, \mathbf{x})$ over $\mathbf{x} \geq \mathbf{0}$, it follows that

$$\mathcal{L}(\boldsymbol{\pi}_i, \mathbf{x}_\theta) \geq \mathcal{L}(\boldsymbol{\pi}_i, \mathbf{x}(\boldsymbol{\pi}_i)) = \mathcal{L}(\boldsymbol{\pi}_i), \quad i = 1, 2.$$

Combining this with (3.11) gives

$$\mathcal{L}(\theta\boldsymbol{\pi}_1 + (1-\theta)\boldsymbol{\pi}_2) \geq \theta\mathcal{L}(\boldsymbol{\pi}_1) + (1-\theta)\mathcal{L}(\boldsymbol{\pi}_2)$$

for each $\theta \in [0,1]$. Thus the dual function is concave, and the level set $\mathcal{S}$ is convex. In the same fashion, the function $\mathcal{L}_B$ is concave for any choice of the index set B.

VI. *Strict ascent of the dual function*:
Define $\mathbf{g} = \nabla\mathcal{L}(\boldsymbol{\lambda}_k)$. Since

$$\left.\frac{d}{ds}\mathcal{L}(\boldsymbol{\lambda}_k + s\mathbf{g})\right|_{s=0} = \nabla\mathcal{L}(\boldsymbol{\lambda}_k)\mathbf{g} = \|\nabla\mathcal{L}(\boldsymbol{\lambda}_k)\|^2,$$

it follows that $\mathcal{L}(\boldsymbol{\lambda}_k + s\mathbf{g}) \geq \mathcal{L}(\boldsymbol{\lambda}_k)$ for $s > 0$ sufficiently small. Let $\bar{s}$ be the largest s for which $\boldsymbol{\lambda}_k + s\mathbf{g} \in \mathcal{S}$ for all $s \in [0, \bar{s}]$. We now show that $\bar{s} \geq 1/c_0$. The proof is by contradiction. Suppose $\bar{s} < 1/c_0$. By the fundamental theorem of calculus,

$$\mathcal{L}(\boldsymbol{\lambda}_k + s\mathbf{g}) = \mathcal{L}(\boldsymbol{\lambda}_k) + \int_0^s \nabla\mathcal{L}(\boldsymbol{\lambda}_k + t\mathbf{g})\mathbf{g}dt.$$

Adding and subtracting $\mathbf{g} = \nabla\mathcal{L}(\boldsymbol{\lambda}_k)$ from the $\nabla\mathcal{L}(\boldsymbol{\lambda}_k + t\mathbf{g})$ factor gives

$$\mathcal{L}(\boldsymbol{\lambda}_k + s\mathbf{g}) = \mathcal{L}(\boldsymbol{\lambda}_k) + s\|\nabla\mathcal{L}(\boldsymbol{\lambda}_k)\|^2 + \int_0^s [\nabla\mathcal{L}(\boldsymbol{\lambda}_k + t\mathbf{g}) - \nabla\mathcal{L}(\boldsymbol{\lambda}_k)]\mathbf{g}dt. \tag{3.12}$$

Since $\boldsymbol{\lambda}_k + t\mathbf{g} \in \mathcal{S}$ for $t \in [0, \bar{s}]$, and $\nabla\mathcal{L}$ is Lipschitz continuous on $\mathcal{S}$ (see III), we have

$$\begin{aligned} \int_0^s [\nabla\mathcal{L}(\boldsymbol{\lambda}_k + t\mathbf{g}) - \nabla\mathcal{L}(\boldsymbol{\lambda}_k)]\mathbf{g}dt &\leq \int_0^s \|\nabla\mathcal{L}(\boldsymbol{\lambda}_k + t\mathbf{g}) - \nabla\mathcal{L}(\boldsymbol{\lambda}_k)\|\|\mathbf{g}\|dt \\ &\leq \int_0^s c_0 t\|\mathbf{g}\|^2 dt = \frac{1}{2}c_0 s^2\|\nabla\mathcal{L}(\boldsymbol{\lambda}_k)\|^2 \end{aligned} \tag{3.13}$$

for each $s \in [0, \bar{s}]$. Combining (3.12) and (3.13) gives

$$\mathcal{L}(\boldsymbol{\lambda}_k + s\mathbf{g}) \geq \mathcal{L}(\boldsymbol{\lambda}_k) + (s - \frac{1}{2}c_0 s^2)\|\nabla\mathcal{L}(\boldsymbol{\lambda}_k)\|^2 \tag{3.14}$$

for each $s \in [0, \bar{s}]$. If $\bar{s} < 1/c_0$, then (3.14) implies that

$$\mathcal{L}(\boldsymbol{\lambda}_k + s\mathbf{g}) \geq \mathcal{L}(\boldsymbol{\lambda}_k) + \frac{s}{2}\|\nabla\mathcal{L}(\boldsymbol{\lambda}_k)\|^2 \tag{3.15}$$

for each $s \in [0, \bar{s}]$.

Due to the continuity of $\mathcal{L}$ (see II), (3.15) yields $\mathcal{L}(\boldsymbol{\lambda}_k + s\mathbf{g}) \geq \mathcal{L}(\boldsymbol{\lambda}_k)$ for s near $\bar{s}$. This violates the fact that $\bar{s}$ was the largest s such that $\boldsymbol{\lambda}_k + s\mathbf{g} \in \mathcal{S}$. Hence, we have a contraction and our original supposition $\bar{s} < 1/c_0$ cannot hold. In other words, $\bar{s} \geq 1/c_0$. Hence, (3.15) holds for all $s \in [0, 1/c_0]$.

VII. *Strict ascent of* $\mathcal{L}_{B_0}$:
We now derive a similar lower bound for $\mathcal{L}_{B_0}$. In IV we observe that $\mathcal{L}(\boldsymbol{\lambda}_k) = \mathcal{L}_{B_0}(\boldsymbol{\lambda}_k)$. Again, by [4, Theorem 2.1] we have $\nabla\mathcal{L}_{B_0}(\boldsymbol{\lambda}_k) = \nabla_{\lambda}\mathcal{L}(\boldsymbol{\lambda}_k, \mathbf{y})$ where $\mathbf{y}$ is the solution to

$$\min_{\mathbf{x}_{B_0} \geq \mathbf{0}} \mathcal{L}(\boldsymbol{\lambda}_k, \mathbf{x}). \tag{3.16}$$

As observed in (3.7), $\mathbf{x}_k = \mathbf{x}(\boldsymbol{\lambda}_k)$ achieves the minimum in (3.16). Consequently, $\mathbf{y} = \mathbf{x}_k$ and $\nabla\mathcal{L}(\boldsymbol{\lambda}_k) = \nabla\mathcal{L}_{B_0}(\boldsymbol{\lambda}_k)$. Combining the fact that $\nabla\mathcal{L}_B$ is Lipschitz continuous over $\mathcal{S}$ with Lipschitz constant c_0, that $\mathcal{L}(\boldsymbol{\lambda}_k) = \mathcal{L}_{B_0}(\boldsymbol{\lambda}_k)$, that $\nabla\mathcal{L}(\boldsymbol{\lambda}_k) = \nabla\mathcal{L}_{B_0}(\boldsymbol{\lambda}_k)$, and that $\mathcal{L}_{B_0}(\boldsymbol{\lambda}) \leq \mathcal{L}(\boldsymbol{\lambda})$ for each $\boldsymbol{\lambda} \in \mathbf{R}^m$, we conclude that $\mathcal{L}_{B_0}$ has a lower bound analogous to that of $\mathcal{L}$ in (3.15):

$$\mathcal{L}_{B_0}(\boldsymbol{\lambda}_k + s\nabla\mathcal{L}(\boldsymbol{\lambda}_k)) \geq \mathcal{L}(\boldsymbol{\lambda}_k) + \frac{s}{2}\|\nabla\mathcal{L}(\boldsymbol{\lambda}_k)\|^2 \tag{3.17}$$

whenever $s \in [0, 1/c_0]$. In the dual subiteration, $\boldsymbol{\nu}_1$ maximizes $\mathcal{L}_{B_0}$ along the ray $\mathcal{R}(\boldsymbol{\lambda}_k, \nabla\mathcal{L}(\boldsymbol{\lambda}_k))$. Hence, taking $s = 1/c_0$ in (3.17) yields the lower bound

$$\mathcal{L}_{B_0}(\boldsymbol{\nu}_1) \geq \mathcal{L}(\boldsymbol{\lambda}_k) + \frac{1}{2c_0}\|\nabla\mathcal{L}(\boldsymbol{\lambda}_k)\|^2. \tag{3.18}$$

VIII. *Convergence*:
Utilizing the lower bound (3.18) in the monotonicity inequalities IV gives:

$$\mathcal{L}(\boldsymbol{\lambda}_{k+1}) \geq \mathcal{L}(\boldsymbol{\lambda}_k) + \frac{1}{2c_0}\|\nabla\mathcal{L}(\boldsymbol{\lambda}_k)\|^2.$$

Summing this inequality over k, we have

$$\mathcal{L}(\boldsymbol{\lambda}_{k+1}) \geq \mathcal{L}(\boldsymbol{\lambda}_0) + \frac{1}{2c_0}\sum_{j=0}^{k}\|\nabla\mathcal{L}(\boldsymbol{\lambda}_j)\|^2. \tag{3.19}$$

Since $\mathcal{S}$ is compact and $\mathcal{L}$ is continuous, $\mathcal{L}$ attains a finite-valued maximum on $\mathcal{S}$. Hence, the gradients $\nabla\mathcal{L}(\boldsymbol{\lambda}_j)$ in (3.19) tend to zero. By the concavity of $\mathcal{L}$, the maximizers of $\mathcal{L}$ coincide with the points where the gradient vanishes. Hence, by the continuity of the gradients of $\mathcal{L}$, any convergent subsequence of the iterates approaches a maximizer of $\mathcal{L}$, and the proof is complete. □

4 LPs and proximal approximations

Now let us consider the linear programming problem

$$\min \ \mathbf{c}^{\mathsf{T}}\mathbf{x} \ \text{ subject to } \ \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}, \tag{4.1}$$

where $\mathbf{b} \in \mathbf{R}^m$, $\mathbf{c} \in \mathbf{R}^n$, and $\mathbf{A}$ is an m by n matrix. In this case, the associated Lagrangian has the following special form:

$$\mathcal{L}(\boldsymbol{\lambda}, \mathbf{x}) = \mathbf{c}^{\mathsf{T}}\mathbf{x} + \boldsymbol{\lambda}^{\mathsf{T}}(\mathbf{b} - \mathbf{A}\mathbf{x}).$$

In this linear setting, $\mathcal{L}$ does not satisfy the strong convexity assumption used in the previous section. One way to deal with the lack of strict convexity is to examine the DASA in the limit, as a strong convexity parameter tends to zero. This limit analysis leads to the LPDASA, as presented in [12], in which the minimization in the DASA subiteration is replaced by a projection. A related approach for handling the lack of strong convexity in the Lagrangian, which we discuss here, involves a proximal regularization. In the proximal approach, additional "small" terms are added to the Lagrangian to achieve suitable strong convexity. Let δ and ϵ denote small positive parameters, and let $\mathbf{y} \in \mathbf{R}^n$ and $\boldsymbol{\mu} \in \mathbf{R}^m$ denote the proximal "shift" vectors. Typically, the shift vectors are zero before any estimate of the primal or dual solutions are known. After approximations to the primal or dual solutions have been determined, $\mathbf{y}$ or $\boldsymbol{\mu}$ might be replaced by these approximations. There is a substantial body of research on proximal point methods. References include [7, 16, 19, 20, 22, 23, 24, 26].

The proximal Lagrangian $\mathcal{L}_p$ corresponding to (4.1) is

$$\mathcal{L}_p(\boldsymbol{\lambda}, \mathbf{x}) = \frac{\epsilon}{2}\|\mathbf{x} - \mathbf{y}\|^2 - \frac{\delta}{2}\|\boldsymbol{\lambda} - \boldsymbol{\mu}\|^2 + \mathbf{c}^\mathsf{T}\mathbf{x} + \boldsymbol{\lambda}^\mathsf{T}(\mathbf{b} - \mathbf{A}\mathbf{x}), \tag{4.2}$$

and the proximal dual is

$$\mathcal{L}_p(\boldsymbol{\lambda}) = \min_{\mathbf{x} \ge \mathbf{0}} \ \mathcal{L}_p(\boldsymbol{\lambda}, \mathbf{x}).$$

If $\delta = \epsilon = 0$, then the proximal Lagrangian coincides with the original Lagrangian. If $\mathbf{y}$ and $\boldsymbol{\mu}$ are chosen to be a solution $\mathbf{x}^*$ and a dual multiplier $\boldsymbol{\lambda}^*$ for the original LP (4.1), then for any choice of δ and ϵ, $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$ maximizes the proximal dual function and $\mathbf{x} = \mathbf{x}^*$ is the associated minimizer in the proximal Lagrangian.

Observe that the DASA can be applied to the proximal Lagrangian (4.2) since $\mathcal{L}_p(\boldsymbol{\lambda}, \mathbf{x})$ is concave in $\boldsymbol{\lambda}$ and strongly convex in $\mathbf{x}$. In this LP setting, $\boldsymbol{\omega}_j$ in the DASA subiteration is the solution $\boldsymbol{\lambda}$ of the following linear equation:

$$(\mathbf{A}_F\mathbf{A}_F^\mathsf{T} + \sigma\mathbf{I})\boldsymbol{\lambda} = \sigma\boldsymbol{\mu} + \mathbf{A}_F\mathbf{c}_F + \epsilon(\mathbf{b} - \mathbf{A}_F\mathbf{y}_F), \tag{4.3}$$

where $\sigma = \delta\epsilon$, F (the free set) is the complement of B_j (the bound set), and $\mathbf{A}_F$ is the submatrix of $\mathbf{A}$ corresponding to the column indices in F. We now show that the inverse of the matrix in (4.3) is approximately a projection matrix; consequently, the proximal iterates generated by the solution of (4.3) track, in a sense, the dual subiterates generated in [12] by a projection.

Without loss of generality, we can assume that the columns of $\mathbf{A}_F$ are linearly independent; that is, there exists a matrix $\mathbf{L}$ with linearly independent columns such that $\mathbf{L}\mathbf{L}^\mathsf{T} = \mathbf{A}_F\mathbf{A}_F^\mathsf{T}$ where $\mathbf{A}_F^\mathsf{T} = \mathbf{Q}\mathbf{L}^\mathsf{T}$ is the factorization of $\mathbf{A}_F^\mathsf{T}$ into the product of a matrix with orthonormal columns and a triangular matrix with nonzero row. The nonzero rows of $\mathbf{L}^\mathsf{T}$, or equivalently, the nonzero columns of $\mathbf{L}$ are trivially independent due to the triangular structure of $\mathbf{L}$.

Returning to the matrix in (4.3), the matrix modification formula [8] gives

$$(\mathbf{I} + \sigma^{-1}\mathbf{A}_F\mathbf{A}_F^\mathsf{T})^{-1} = \mathbf{I} - \sigma^{-1}\mathbf{A}_F(\mathbf{I} + \sigma^{-1}\mathbf{A}_F^\mathsf{T}\mathbf{A}_F)^{-1}\mathbf{A}_F^\mathsf{T}. \tag{4.4}$$

Recall the geometric series expansion

$$(\mathbf{I} + \mathbf{C})^{-1} = \mathbf{C}^{-1} - \mathbf{C}^{-2} + \mathbf{C}^{-3} - \ldots,$$

which is valid when $\mathbf{C}$ is invertible and the spectral radius of $\mathbf{C}^{-1}$ is less than one. Using this in (4.4) with $\mathbf{C} = \sigma^{-1}\mathbf{A}_F^\mathsf{T}\mathbf{A}_F$, we see that when σ is sufficiently small,

$$(\mathbf{I} + \sigma^{-1}\mathbf{A}_F\mathbf{A}_F^\mathsf{T})^{-1} = \mathbf{I} - \mathbf{A}_F[(\mathbf{A}_F^\mathsf{T}\mathbf{A}_F)^{-1} - \sigma(\mathbf{A}_F^\mathsf{T}\mathbf{A}_F)^{-2} + \sigma^2(\mathbf{A}_F^\mathsf{T}\mathbf{A}_F)^{-3} - \ldots]\mathbf{A}_F^\mathsf{T}.$$

Since the matrix $\mathbf{P} = \mathbf{I} - \mathbf{A}_F(\mathbf{A}_F^\mathsf{T}\mathbf{A}_F)^{-1}\mathbf{A}_F^\mathsf{T}$ projects a vector into the space orthogonal to the columns of $\mathbf{A}_F$, and since σ is typically a small number, it follows

Table 1 LPs associated with the quadratic assignment problem

Problem	Rows	Columns	Nnz
qap08	912	1632	7296
qap12	3192	8856	38304
qap15	6330	22275	94,950
nug05	210	225	1050
nug06	372	486	2232
nug07	602	931	4214
nug08	912	1632	7296
nug12	3192	8856	38,304
nug15	6330	22275	94,950
nug20	15240	72600	304,800
nug30	52260	379350	1,567,800

that

$$(\mathbf{A}_F\mathbf{A}_F^{\mathsf{T}} + \sigma\mathbf{I})^{-1} \approx \frac{1}{\sigma}\mathbf{P}.$$

Thus the solution of (4.3) is nearly proportional to the projection of the right side into the space orthogonal to the columns of $\mathbf{A}_F$.

5 Numerical results

We compare CPLEX (V7.0.0) with our iterative implementation of the LPDASA using the SSOR preconditioned conjugate gradient scheme of Björck and Elfving [3] to solve (4.3), or equivalently, to ascend the function $\mathcal{L}_{B_j}^0$ in the dual subiteration. The test problems, derived from lower bounds for the quadratic assignment problem, are available at several web sites including:

- `http://www.research.att.com/~mgc`
- `http://www.netlib.org/lp`
- `http://www.cise.ufl.edu/~davis`
- `http://www.math.ufl.edu/~coap`

A description of the matrices in these problems is given in Table 1. The column labeled Nnz gives the number of nonzeros in $\mathbf{A}$. The problems qap08, qap12, and qap15, found at the netlib site, appear to be permutations of the problems nug08, nug12, and nug15 found at Rescende's site.

Run time statistics for CPLEX dual simplex and CPLEX barrier routines, on an IBM RS6000 computer, appear in Table 2. There was insufficient memory to solve the nug30 problem using either the barrier method or the simplex method. The barrier method also ran out of memory on nug20. The statistics for the dual simplex method on nug20, flagged by *, were estimated in the following way: After 974,000 seconds and 418,000 iterations, CPLEX dual simplex achieved the relative error .15 in the cost. In the CPLEX run for nug15, 97.6% of the run time was spent after the relative cost error reached .15. By using the same constant of proportionality in nug20, we arrived at the estimates for time and iterations given in Table 2.

Run time statistics for iterative LPDASA appear in Table 3. In these experiments, we solved the problems to 3 and 7-digit accuracy. That is, when the primal

Table 2 CPLEX dual simplex and barrier: time (seconds) and iterations (its)

Problem Name	Simplex (secs)	Barrier (secs)	Simplex (its)	Barrier (its)
qap08	8.6	2.6	4,924	8
qap12	1,568	95	102,310	12
qap15	84,606	750	1,538,300	15
nug05	.03	.11	100	8
nug06	.19	.36	400	8
nug07	1.7	1.0	1,700	9
nug08	8.4	2.6	4,900	8
nug12	4,525	95	284,788	12
nug15	150,128	648	265,0179	11
nug20	40,248,000*	???	17,273,000*	???
nug30	???	???	???	???

Table 3 Iterative LPDASA: time, iterations, 3 and 7-digit accuracy

Problem Name	3-digit (secs)	7-digit (secs)	3-digit (its)	7-digit (its)
qap08	.7	6.3	542	6,000
qap12	56	427	12,600	103,300
qap15	187	1,339	17,400	173,200
nug05	.2	.5	900	2,400
nug06	.7	1.9	1,600	5,100
nug07	1.5	39.5	2,200	67,100
nug08	3.5	7.4	3,100	7,100
nug12	49	399	10,800	96,800
nug15	282	1,463	27,300	152,200
nug20	832	13,870	25,100	480,200
nug30	16,200	215,000	68,600	1,007,300

and dual solutions were normalized to be unit vectors, the Kuhn-Tucker conditions were satisfied with a relative error, in the L^∞ norm, of either 10^{-3} or 10^{-7}. In comparing Tables 2 and 3, we see that the simplex method is competitive only for the very smallest problems. Iterative LPDASA achieved the best computing time for 3-digit accuracy. For 7-digit accuracy and moderate size problems (up to 6330 rows), the barrier method had the lowest computing time. For larger problems, however, there was insufficient memory for the factorization used in the barrier method.

The solid line of Figure 1 plots the relative error versus computing time (seconds) for iterative LPDASA and test problem qap12 (3192 rows). Observe that a low accuracy solution is obtained relatively quickly: 2-digit accuracy in 13 seconds, 3-digit accuracy in 56 seconds. For between 4 and 8 digits, the cpu time increases more rapidly, and about 380 seconds are needed to boost the accuracy from 4 to

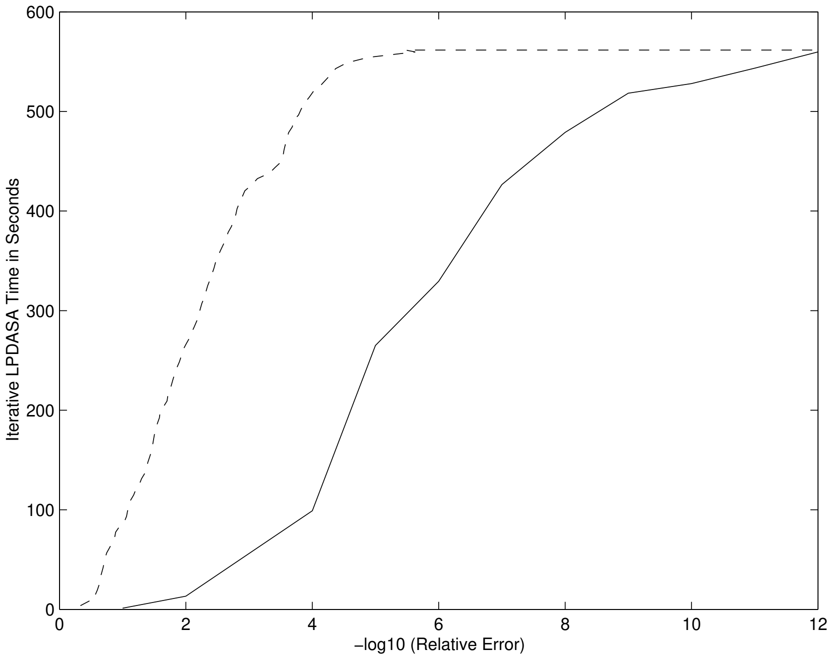

Figure 1 Iterative LPDASA time (solid) and simplex scaled time (dashed) versus relative error for qap12

8 digits. For more than 8 digits, the cpu time increases more slowly–about 81 seconds are needed to increase accuracy from 8 to 12 digits. Roughly, we observe the following behavior in the iterations: For between 4 and 8 significant digits, much of the cpu time is consumed identifying the active constraints. After 8 significant digits have been achieved, the active constraints are fairly well identified, and the iterative method is essentially solving a linear system with ever increasing accuracy.

The convergence of both the simplex and barrier methods is much different from the convergence of iterative LPDASA shown in Figure 1. For comparison, we scale the computing time in the CPLEX dual simplex run and plot with a dashed line in Figure 1 the relative cost error versus scaled time. Observe that 3-digit accuracy is achieved after 76% of the run, while in iterative LPDASA, 3-digit error is achieved after 10% of the run. With the simplex method, the convergence speed is relatively slow until the very end of the run when there is a huge improvement in accuracy. Iterative LPDASA exhibits steady convergence throughout the run.

The set of test problems used in this paper has the property that **A** is relatively sparse while factorizations of **A** or $\mathbf{AA}^{\mathsf{T}}$ are relatively dense. For example, **A** in qap15 is 99.93% zero, while if $\mathbf{AA}^{\mathsf{T}}$ is ordered using approximate minimum degree [1], then the Cholesky factor is only 53.36% zero. Hence, as the size of the problems increases, a point is reached where it is no longer possible to store the Cholesky factor. For problems like these with relatively dense factors, the time to factor the matrix grows proportional to m^3, the number of rows of **A** cubed. Hence, even if it were possible to store factors of the larger matrices in this test set, the computing time should become quite large. For comparison, with iterative LPDASA and 7-digit accuracy, the ratio of running times for nug30 and nug20 is 15.6, while the ratio of row dimensions is 3.4. This implies that running time grew proportional to $m^{2.2}$. Thus iterative LPDASA was able to exploit the sparsity of $\mathbf{A}$ in order to achieve a computing time which grew more slowly with m than CPLEX barrier; the number of nonzeros in $\mathbf{A}$ grows more slowly than m^3, which is reflected in a computing time that grows more slowly than m^3.

In these experiments with iterative LPDASA, the matrix $\mathbf{A}$ was stored in a standard sparse matrix format, the same format used in Malab. Note though that iterative schemes such as conjugate gradients, diagonal preconditioned conjugate gradients, or SSOR preconditioned conjugate gradients, applied to (4.3), only involves the product of submatrices of $\mathbf{A}$ and a vector. Hence, it is possible to implement iterative LPDASA without storing $\mathbf{A}$. In problems like qapxx and nugxx, where the product of submatrices of $\mathbf{A}$ with a vector amount to summing various components of the vector, these coded products should run much quicker than a product which is implemented by looking up coefficients stored in a sparse format.

In summary, iterative LPDASA is relatively efficient for certain classes of linear programming problems. In the test set investigated in this paper, iterative LDASA achieved the best computing time for up to 3 digit accuracy, when compared to CPLEX dual simplex and CPLEX barrier. Although the iterative approach could be used to compute high accuracy solutions, CPLEX barrier was more efficient for problems of size up to 6330 rows (whose factorization also fit in the available memory). With iterative LPDASA, we solved problems with up to 52260 rows. In the current implementation of our code, we need enough memory to store $\mathbf{A}$ itself (not a factorization of $\mathbf{A}$ or $\mathbf{A}\mathbf{A}^{\mathsf{T}}$). By coding the products of submatrices of $\mathbf{A}$ with vectors, this memory constraint could be removed, and it should be possible to solve problems in space proportional to that required by the primal and dual solutions.

References

[1] P. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM Journal on Matrix Analysis and Applications, **17** (1996), pp. 886–905.

[2] M. Bergounioux and K. Kunisch, *Primal-dual strategy for state-constrained optimal control problems*, to appear in Computational Optimization and Applications.

[3] Å. Björck and T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT, 19 (1979), pp. 145–163.

[4] F. H. Clarke, Generalized gradients and applications, *Transactions of the American Mathematical Society*, **205** (1975), pp. 247–262.

[5] R. De Leone and O. L. Mangasarian, *Serial and parallel solution of large scale linear programs by augmented Lagrangian successive overrelaxation*, In Optimization, Parallel Processing and Applications, A. Kurzhanski, K. Neumann, and D. Pallaschke, eds., Springer-Verlag, Berlin, Lecture Notes in Economics and Mathematical Systems, Vol. 304, 1988, pp. 103–124.

[6] Z. Drezner, *Lower bounds based on linear programming for the quadratic assignment problem*, Computational Optimization and Applications, **4** (1995), pp. 159–165.

[7] C. D. Ha, *A generalization of the proximal point algorithm*, SIAM Journal on Control and Optimization, **28** (1990), pp. 503–512.

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

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

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

[11] W. W. Hager, *Iterative methods for nearly singular linear systems*, SIAM Journal on Scientific Computing, **22** (2000), pp. 747–766

[12] W. W. Hager, *The dual active set algorithm and its application to linear programming*, Computational Optimization and Applications, **21** (2002), pp. 263–275.

[13] W. W. Hager and D. Hearn, *The dual active set algorithm applied to quadratic networks*, Computational Optimization and Applications, **1** (1993), pp. 349–373.

[14] W. W. Hager and G. Ianculescu, *Dual approximations in optimal control*, SIAM Journal on Control and Optimization, **22** (1984), pp. 423–465.

[15] Y. Li, P. M. Pardalos, K. G. Ramakrishnan, and M. G. C Rescende, *Lower bounds for the quadratic assignment problem*, Annals of Operations Research, **50** (1994), pp. 387–411.

[16] F. J. Luque, *Asymptotic convergence analysis of the proximal point algorithm*, SIAM Journal on Control and Optimization, **22** (1984), pp. 277–293.

[17] O. L. Mangasarian, *Nonlinear Programming*, McGraw-Hill, New York, 1969.

[18] O. L. Mangasarian, *Iterative solution of linear programs*, SIAM Journal Numerical Analysis, **18** (1981), pp. 606–614.

[19] B. Martinet, *Regularisation inèquations variationelles par approximations successives*, Revue Francaise Informatique et Recherche Opérationnelle, (1970), pp. 154–159.

[20] B. Martinet, *Determination approachée d'un point fixe d'une application pseudo-contractante*, Comptes Rendus des Séances de l'Académie des Sciences, Paris, **274** (1972), pp. 163–165.

[21] M. G. C. Rescende, K. G. Ramakrishnan, and Z. Drezner, *Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming*, Operations Research, **43** (1995), pp. 781–791.

[22] R. T. Rockafellar, *Monotone operators and the proximal point algorithm*, SIAM Journal on Control and Optimization, **14** (1976), pp. 877–898.

[23] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Mathematics of Operations Research, **2** (1976), pp. 97–116.

[24] J. E. Spingarn, *Submonotone mappings and the proximal point algorithm*, Numerical Functional Analysis and Optimization, **4** (1982), pp. 123–150.

[25] S. J. Wright, *Implementing proximal point methods for linear programming*, Journal of Optimization Theory and Applications, **65** (1990), pp. 531–554.

[26] C. Zhu, *Asymptotic convergence analysis of some inexact proximal point algorithms for minimization*, SIAM Journal on Optimization, **6** (1996), pp. 626–637.