Comput Optim Appl (2008) 41: 151–183
DOI 10.1007/s10589-007-9096-y

Implementation of warm-start strategies in interior-point methods for linear programming in fixed dimension

Elizabeth John · E. Alper Yıldırım

Received: 11 May 2006 / Revised: 12 December 2006 / Published online: 9 November 2007

Abstract We implement several warm-start strategies in interior-point methods for linear programming (LP). We study the situation in which both the original LP instance and the perturbed one have exactly the same dimensions. We consider different types of perturbations of data components of the original instance and different sizes of each type of perturbation. We modify the state-of-the-art interior-point solver PCx in our implementation. We evaluate the effectiveness of each warm-start strategy based on the number of iterations and the computation time in comparison with "cold start" on the NETLIB test suite. Our experiments reveal that each of the warm-start strategies leads to a reduction in the number of interior-point iterations especially for smaller perturbations and for perturbations of fewer data components in comparison with cold start. On the other hand, only one of the warm-start strategies exhibits better performance than cold start in terms of computation time. Based on the insight gained from the computational results, we discuss several potential improvements to enhance the performances of such warm-start strategies.

Keywords Linear programming · Interior-point methods · Warm-start strategies · Reoptimization

This research was supported in part by NSF through CAREER grant DMI-0237415.

E. John
Automatic Data Processing, Inc., Edgewood, NY 11717, USA
e-mail: lizjohn1@gmail.com

E.A. Yıldırım (✉)
Department of Industrial Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey
e-mail: yildirim@bilkent.edu.tr

1 Introduction

Having solved an optimization problem, the computational effort of solving another closely related optimization problem can in general be reduced if one can properly take advantage of the information gained during the course of the solution of the original problem. The techniques aimed at identifying an advanced starting point for the solution of a nearby optimization problem using the information gained from the original one are referred to as "warm-start strategies." Many optimization algorithms such as sequential linear/quadratic programming, branch-and-bound, and decomposition methods require the solution of a sequence of closely related optimization problems. Therefore, the development of effective warm-start strategies is essential in order to reduce the computational cost of such widely used sequential algorithms.

Since Karmarkar's pathbreaking work [19], interior-point methods (IPMs) have dominated research in continuous optimization in the last two decades. These methods have proved to be effective in solving a rather large class of convex optimization problems both in theory and in practice. Despite the fact that IPMs are well-understood in the broad context of convex optimization (see, e.g., [25, 28]), the development of warm-start strategies is still an active area of research.

Unlike the simplex method for linear programming (LP), IPMs generate a sequence of interior-points that converge to an optimal solution in the limit. An optimal basis of an LP problem usually serves as an excellent warm-start to resolve another closely related LP problem using the simplex method. However, IPMs work with interior-points and tend to generate much better search directions at points that are away from the boundary of the feasible region. Therefore, an optimal or a near-optimal solution of the original LP problem is in general not a very good candidate to be used as a warm-start for the solution of a nearby problem. This major difference between the simplex method and IPMs makes the development of effective warm-start strategies in IPMs a nontrivial problem.

For LP, research on warm-start strategies in IPMs has focused on two cases. In the first case, a nearby LP problem is obtained by adding constraints and/or variables to a given LP problem. This situation arises, for instance, in cutting plane schemes (see, e.g., [15, 22–24]) and in the context of branch-and-bound methods [6]. In addition, similar warm-start strategies have been developed for analytic center cutting plane methods in the case of central and deep cuts (see, e.g., [12, 13] and the references therein).

In the second case, the nearby LP problem has exactly the same number of constraints and variables as the original problem but the data is perturbed. This situation arises, for instance, in the sequential linear programming algorithm for nonlinear optimization and in the branch-and-bound method for integer programming for two sibling subproblems. This case has been studied in [2, 10, 11, 16, 17, 20, 27, 34]. Furthermore, the reader is referred to [7, 8] for warm-start strategies for convex multicriteria optimization problems and to [3, 9] for more general nonlinear optimization problems.

Several different approaches are proposed in the existing literature on warm-start strategies and reoptimization in the context of IPMs. Each of the suggested methods relies on having a solution or a set of solutions for the original optimization

problem that satisfies a set of desirable properties such as near-feasibility, near-optimality, and/or proximity to the central path. Typically, after perturbing the original LP problem, the previous solution fails to satisfy primal feasibility or dual feasibility or both. The existing methods propose different ways to handle this situation. For LP, while some strategies rely on computing an adjustment or several adjustments to the previously stored iterate to regain feasibility for the perturbed problem (see, e.g., [6, 15–17, 22–24, 34]), other methods modify the perturbed problem using judiciously chosen penalty or barrier parameters so that the stored iterate can be used as is (see, e.g., [2, 10, 11, 20, 27]). Computational results indicate varying degrees of success of some of these methods (see, e.g., [2, 16, 17, 22, 23]). The question of how to perform reoptimization in the context of IPMs does not seem to have been settled at the time of writing this manuscript. As a result, warm-start strategies in IPMs still remain an exciting avenue of research.

In this paper, we focus on the implementation of warm-start strategies in IPMs for LP and mainly rely on the theoretical framework developed by Yıldırım and Wright [34]. These strategies can be applied in the case in which the perturbed LP problem has the same dimensions as the original one. In their setting, the original LP problem is solved using a feasible primal-dual path-following IPM and a subset of the iterates generated during the course of the solution is stored. Given the perturbed LP problem, the proposed warm-start strategies are based on computing an adjustment at an iterate of the original problem so that the adjusted iterate is strictly feasible for the perturbed problem and is relatively well-centered. The procedure is started from the last stored iterate in an attempt to obtain an advanced starting iterate for the perturbed problem with a small duality measure. If the computed adjustment fails to produce an acceptable starting point for the perturbed problem, one retreats to an earlier iterate in the sequence of stored iterates and repeats the same procedure. If none of the stored iterates yields an acceptable starting point, the perturbed problem then is solved from scratch (i.e., "cold start"). In [34], two adjustments are proposed, namely a least-squares adjustment and a Newton step adjustment. The authors establish sufficient conditions on the size of the perturbation as a function of the problem data and certain algorithmic parameters in order for the adjusted iterate to be used as an acceptable starting point for the perturbed problem. These sufficient conditions lead to improved iteration complexity estimates to solve the perturbed LP problem using an IPM starting from the computed warm-start for small perturbations. As one would expect, these theoretical results indicate that warm-start strategies have a greater potential for reduced computational effort for smaller perturbations. To the best of our knowledge, this study presents one of the first complexity results for reoptimization using warm-start strategies. However, the theoretical results of this study are not accompanied with computational experiments.

It is well-known that neither of the assumptions of feasibility and proximity to the central path is enforced in any of the existing interior-point LP solvers. Therefore, the main objective of this paper is to evaluate the performance of these warm-start strategies in a more practical and realistic setting in an attempt to gain insight into the practical implications of the favorable theoretical results of [34] established under more restrictive assumptions.

In addition to the two adjustments suggested in [34], we consider and experiment with several other adjustments in this paper. We use state-of-the-art interior-point

code PCx [5] in our implementation. The warm-start strategies are tested on the standard testbed of NETLIB problems.[1] Our extensive computational results indicate that warm-start strategies are indeed effective in reducing the number of iterations for re-optimization of the perturbed problem especially for smaller perturbations and for perturbations of fewer data components. In terms of the computation time, our results reveal that warm-start strategies that can quickly identify an acceptable starting point lead to the most significant savings in comparison with cold start.

This paper is organized as follows. We define our notation in Sect. 1.1 and give a general overview of warm-start strategies in Sect. 2. The details of the implementation of warm-start strategies are presented in Sect. 3. Section 4 is devoted to the presentation and the discussion of the computational results. Finally, we conclude the paper with some future research directions in Sect. 5.

1.1 Notation

We reserve upper case Roman letters for matrices. For a vector $u \in \mathbb{R}^n$, $\|u\|$ is the Euclidean norm of u, u_i is the ith component of u, and U denotes the diagonal matrix whose entries are given by the components of u. The members of a sequence are identified using superscripts. We use e to denote the vector of ones in the appropriate dimension.

2 An overview of warm-start strategies

Consider an LP problem in standard form:

$$(\mathcal{P}) \quad \min_x \ c^T x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0,$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$ are given and $x \in \mathbb{R}^n$ is the decision variable. We assume that the matrix A has full row rank without loss of generality. The associated dual LP problem is given by

$$(\mathcal{D}) \quad \max_{y,s} \ b^T y \quad \text{s.t.} \quad A^T y + s = c, \quad s \geq 0,$$

where $y \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$ are the corresponding decision variables.

We use $d = (A, b, c)$ to denote the data of the original (unperturbed) LP problem. Note that d completely specifies an instance of a primal and dual pair of LP problems in standard form. The perturbed instance is denoted by $d + \Delta d$, where $\Delta d := (\Delta A, \Delta b, \Delta c)$ satisfies $\Delta A \in \mathbb{R}^{m \times n}$, $\Delta b \in \mathbb{R}^m$, and $\Delta c \in \mathbb{R}^n$. This implies that the original and the perturbed primal (dual) LP problems have precisely the same number of constraints and variables. We assume that the coefficient matrix $A + \Delta A$ continues to have full row rank.

[1] http://www.netlib.org/lp/index.html.

2.1 An overview of infeasible path-following methods

The most effective variant of IPMs in practice are the infeasible primal-dual path-following methods. These methods generate iterates $(x^k, y^k, s^k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$, $k = 0, 1, \ldots$ with $x^k > 0$ and $s^k > 0$ that somewhat loosely follow the so-called central path $\mathcal{C}$, which is defined as the set of solutions $(x(\mu), y(\mu), s(\mu)) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ to the following nonlinear system of equations and inequalities parametrized by the scalar $\mu > 0$:

$$Ax = b, \qquad A^T y + s = c, \qquad XSe = \mu e, \quad x > 0,\ s > 0. \tag{1}$$

Under the assumption that both $(\mathcal{P})$ and $(\mathcal{D})$ have feasible solutions that strictly satisfy the nonnegativity constraints (such solutions are called *strictly feasible*), it is well-known that the central path is well-defined and converges to an optimal solution of $(\mathcal{P})$ and $(\mathcal{D})$ as μ decreases to zero.

Infeasible primal-dual path-following IPMs generate iterates $(x^k, y^k, s^k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ with $x^k > 0$ and $s^k > 0$ that are not necessarily feasible for the primal or dual problems. As such, they offer greater flexibility as the issue of computing a feasible primal-dual solution is circumvented. Instead, the central path is used to guide the iterates towards feasibility and optimality simultaneously. For an iterate (x^k, y^k, s^k), the corresponding duality measure μ^k is defined by

$$\mu^k := ((x^k)^T s^k)/n, \quad k = 0, 1, \ldots.$$

A typical interior-point iteration at $(x, y, s) := (x^k, y^k, s^k)$ consists of taking a Newton step towards a point on the central path whose duality measure is not greater than that of (x, y, s). This amounts to solving the following Newton system:

$$A\Delta x = r_b, \tag{2a}$$

$$A^T \Delta y + \Delta s = r_c, \tag{2b}$$

$$X\Delta s + S\Delta x = -XSe + \sigma\mu e, \tag{2c}$$

where μ is the duality measure of (x, y, s), $\sigma \in [0, 1]$, and r_b and r_c are respectively the primal and dual infeasibility residuals given by

$$r_b := b - Ax, \qquad r_c := c - A^T y - s. \tag{3}$$

The Newton system (2) is most commonly solved by eliminating Δs and Δx from the system using (2c) and (2a), respectively, which leads to the following so-called normal equations form:

$$ADA^T \Delta y = r_b + A(Dr_c + x - \sigma\mu S^{-1} e), \tag{4}$$

where $D = XS^{-1}$. Once Δy is computed using a Cholesky factorization of ADA^T, Δs and Δx can be computed using (2b) and (2c), respectively:

$$\Delta s = r_c - A^T \Delta y,$$

$$\Delta x = -x + \sigma\mu S^{-1} e - D\Delta s.$$

Finally, a step length $\beta \in (0, 1]$ is chosen to ensure that $x + \beta\Delta x > 0$ and $s + \beta\Delta s > 0$. The reader is referred to the book by Wright [30] for a comprehensive treatment of IPMs.

The major computational effort in an interior-point iteration is the computation and the factorization of the $m \times m$ positive definite matrix ADA^T in (4). The performance of an interior-point solver highly depends on how effectively linear algebra subroutines can handle special structures such as sparsity and dense columns arising in normal equations.

2.2 A generic warm-start algorithm

In this subsection, we present the generic warm-start framework developed in [34]. Since our goal in this paper is to assess the practical performance of these warm-start strategies, we shall exclusively use the phrase "warm-start strategies" to refer to those proposed in [34]. As stated in Sect. 1, we stress that there exist several other warm-start strategies that handle warm-starts in different ways.

Suppose that the original instance d is solved using a primal-dual path-following IPM. Let $\{(x^k, y^k, s^k) : k = 0, \ldots, l\}$ denote the set of iterates generated during the course of the solution of d and let $\mathcal{T} \subseteq \{0, \ldots, l\}$ be a subset of the indices of these IPM iterates. The generic warm-start algorithm proposed in [34] is outlined in Algorithm 2.1.

Algorithm 2.1 Generic warm-start algorithm

Require: $d, \Delta d, l, \mathcal{T} \subseteq \{0, \ldots, l\}$
(beginning of the search stage)
flag $\leftarrow$ false.
For $k = l$ downto 0, $k \in \mathcal{T}$ do
loop
  Compute an adjustment $(\Delta x^k, \Delta y^k, \Delta s^k)$ as a function of (x^k, y^k, s^k) and Δd.
  if $(x^k, y^k, s^k) + (\Delta x^k, \Delta y^k, \Delta s^k)$ is an "acceptable" starting point for $d + \Delta d$ **then**
    flag $\leftarrow$ true.
    $t \leftarrow k$.
    Break.
  end if
end loop
(end of the search stage)
(beginning of the reoptimization stage)
if flag = true **then**
  Solve $d + \Delta d$ starting with $(x^t, y^t, s^t) + (\Delta x^t, \Delta y^t, \Delta s^t)$.
else
  Solve $d + \Delta d$ using cold start.
end if
(end of the reoptimization stage)

We now describe Algorithm 2.1 in detail. Given an LP instance d, a perturbation Δd, and a subset $\mathcal{T}$ of the indices of the iterates generated during the course of the solution of d, the algorithm starts with the most advanced iterate (x^k, y^k, s^k), $k \in \mathcal{T}$ and computes an adjustment $(\Delta x^k, \Delta y^k, \Delta s^k)$, which depends on $d + \Delta d$ and may also depend on (x^k, y^k, s^k). Then, if the adjusted iterate $(x^k, y^k, s^k) + (\Delta x^k, \Delta y^k, \Delta s^k)$ is an "acceptable" starting point for $d + \Delta d$, the Boolean variable "flag" is updated to indicate that a successful warm-start has been computed and the perturbed instance $d + \Delta d$ is solved with an IPM starting from this warm-start. Otherwise, the algorithm retreats to the next most advanced iterate among the subset of stored iterates and repeats the same procedure. If none of the stored iterates yields an acceptable warm-start, then the algorithm simply reverts to cold start to solve $d + \Delta d$.

The main motivation in developing a warm-start strategy is the expectation that two closely related optimization problems should in general share similar characteristics. Note that the description of Algorithm 2.1 is mainly driven by this observation. In particular, Algorithm 2.1 relies on the set of stored iterates of the original instance d in an attempt to compute an acceptable warm-start for the perturbed instance $d + \Delta d$. This is in contrast with the classical reoptimization approach using the simplex method or with several other warm-start strategies using IPMs which store only a single, advanced (or optimal) iterate of the original instance d and allow for several adjustments. Since Algorithm 2.1 computes a single adjustment for each stored iterate of the original instance d, having a set of stored iterates in general increases the likelihood of successfully computing an acceptable warm-start for the perturbed instance. Using a single adjustment as opposed to several adjustments enables one to prove sufficient conditions on the perturbation Δd so that a successful warm-start can be computed under certain assumptions [34]. At the same time, Algorithm 2.1 offers the flexibility of retreating to an earlier iterate in an attempt to absorb the infeasibility arising from a larger perturbation in a single adjustment. Therefore, Algorithm 2.1 is in general designed to deal with general perturbations and does not require any prior information about the perturbation as long as the dimension of the perturbed instance coincides with that of the original one.

Note that we have intentionally used the ambiguous adjective "acceptable" in the description of the warm-start algorithm. An acceptable starting point may be defined in various ways. At the very least, the adjusted iterate $(x + \Delta x, y + \Delta y, s + \Delta s)$ should satisfy $x + \Delta x > 0$ and $s + \Delta s > 0$ since it will be used as the initial point to solve the perturbed instance using an IPM. For instance, if $d + \Delta d$ is known to have strictly feasible primal-dual solutions, one may insist on obtaining such a starting point. Furthermore, one may even require that the starting point lie in some neighborhood of the central path for $d + \Delta d$ in an attempt to obtain a well-centered point. Note that complexity analyses of IPMs are carried out under the assumption that iterates lie in some well-defined neighborhood of the central path. In fact, in the theoretical framework of [34], it is assumed that both d and $d + \Delta d$ have strictly feasible solutions and that d is solved using a feasible IPM with a central path neighborhood restriction. Under these assumptions, sufficient conditions on the size of Δd and the duality measure of the iterate of the original instance are established to ensure that Algorithm 2.1 will succeed in computing a well-centered strictly feasible iterate for

$d + \Delta d$ using specific adjustments. Furthermore, solving the perturbed instance starting from an advanced iterate obtained in this manner leads to improved iteration complexity in comparison with cold start [34]. We describe how we define an acceptable iterate for the purposes of our implementation in the following subsection.

2.2.1 Acceptable starting points

The most effective interior-point solvers use infeasible path-following IPMs and they usually do not impose any central path neighborhood restrictions. As outlined in Sect. 2.1, such methods allow for infeasible iterates and work towards feasibility and optimality simultaneously. In contrast with the theoretical framework of [34], we do not make any assumptions on the instances d and $d + \Delta d$. Therefore, it may be the case that neither of the instances d and $d + \Delta d$ may have strictly feasible solutions. In the case that the perturbed instance $d + \Delta d$ does not possess a strictly feasible point or is primal and/or dual infeasible, insisting on having a strictly feasible starting point for $d + \Delta d$ will necessarily force Algorithm 2.1 to evaluate each of the stored iterates of the original instance d in an attempt to compute a feasible solution of $d + \Delta d$. Clearly, the search stage will fail to produce a warm-start for $d + \Delta d$, in which case, $d + \Delta d$ will be resolved using cold start and the search stage of Algorithm 2.1 will be a waste of computational effort. Therefore, in practice, one needs to define an "acceptable starting point" in a more realistic and less restrictive fashion. For instance, it may be reasonable to deem a computed starting point "acceptable" even if it has a small infeasibility residual with respect to the perturbed instance $d + \Delta d$.

Infeasible path-following IPMs generally achieve feasibility relatively quickly and then work towards optimality. It follows that advanced iterates of a primal-dual feasible original instance d usually have small infeasibility residuals (e.g., 10^{-6} or smaller). For small perturbations Δd, it may therefore be quite reasonable to accept the same infeasibility residual at a starting point of $d + \Delta d$. This amounts to computing an adjustment based only on Δd while ignoring the infeasibility of an iterate with respect to the original problem d. More precisely, given an interior-point iterate (x, y, s) of the original instance d, the computed adjustment satisfies

$$\bar{A}\Delta x = \Delta b - \Delta A x, \tag{5a}$$

$$\bar{A}^T \Delta y + \Delta s = \Delta c - \Delta A^T y, \tag{5b}$$

where $\bar{A} = A + \Delta A$. It follows from (5) that the primal and dual infeasibility residuals of the original iterate are identical to those of the candidate warm-start since $r_p := b - Ax = b + \Delta b - \bar{A}(x + \Delta x)$ and $r_d := c - A^T y - s = c + \Delta c - \bar{A}^T (y + \Delta y) - (s + \Delta s)$, respectively. In our implementation, the infeasibility residuals of the original iterate therefore are passed directly into the candidate warm-start. In conclusion, at a stored iterate (x, y, s) of d, the computed adjustment $(\Delta x, \Delta y, \Delta s)$ satisfies (5) and the resulting iterate $(x + \Delta x, y + \Delta y, s + \Delta s)$ is deemed acceptable if $x + \Delta x > 0$ and $s + \Delta s > 0$.

Another reason for ignoring the infeasibilities of the original iterate is the expectation that the warm-start strategies may have the potential to be useful in detecting infeasibility of the perturbed instance $d + \Delta d$ in fewer iterations in comparison with cold start. Detecting infeasibility in IPMs is an important problem both in theory and in practice. The reader is referred to [29] for theoretical results on this issue.

2.2.2 Adjustments

We now describe various adjustments that can be incorporated into Algorithm 2.1. Our choices are motivated by the theoretical foundation developed in [34]. In particular, Yıldırım and Wright propose a least-squares adjustment and a Newton step adjustment, both of which shall be explained in detail below.

Family of least-squares adjustments Let (x, y, s) be an iterate generated by an IPM during the course of the solution of the instance d. For the perturbed instance $d + \Delta d$, the family of least-squares adjustments is given by the optimal solutions of

$$(\mathcal{PA}) \quad \min_{\Delta x} \|\Sigma \Delta x\| \quad \text{s.t.} \quad \bar{A}\Delta x = \Delta b - \Delta A x,$$

$$(\mathcal{DA}) \quad \min_{\Delta y, \Delta s} \|\Lambda \Delta s\| \quad \text{s.t.} \quad \bar{A}^T \Delta y + \Delta s = \Delta c - \Delta A^T y,$$

where Σ and Λ are positive diagonal matrices in $\mathbb{R}^{n\times n}$ and $\bar{A} := A + \Delta A$. Note that the constraints of the optimization problems $(\mathcal{PA})$ and $(\mathcal{DA})$ ensure that $(\Delta x, \Delta y, \Delta s)$ satisfies (5a) and (5b), respectively, i.e., primal and dual infeasibility residuals of the original iterate (x, y, s) are transferred to the computed iterate $(x + \Delta x, y + \Delta y, s + \Delta s)$. If, in addition, $x + \Delta x > 0$ and $s + \Delta s > 0$, then the resulting iterate is deemed an acceptable starting solution for $d + \Delta d$.

Since $(\mathcal{PA})$ and $(\mathcal{DA})$ are least-squares problems, they have closed form solutions given by

$$\Delta x_\Sigma = \Sigma^{-2}\bar{A}^T(\bar{A}\Sigma^{-2}\bar{A}^T)^{-1}[\Delta b - \Delta A x], \tag{6a}$$

$$\Delta y_\Lambda = (\bar{A}\Lambda^2\bar{A}^T)^{-1}\bar{A}\Lambda^2(\Delta c - \Delta A^T y), \tag{6b}$$

$$\Delta s_\Lambda = \Delta c - \Delta A^T y - \bar{A}^T \Delta y_\Lambda. \tag{6c}$$

There are several choices for the diagonal scaling matrices Σ and Λ. Yıldırım and Wright [34] propose and study the plain least-squares adjustment (PLSA) given by $\Sigma = \Lambda = I$, the identity matrix.

In addition, the diagonal scaling matrices can be chosen as a function of the current iterate (x, y, s). For instance, reasonable choices of Σ include $X^{-1}, X^{-1/2}S^{1/2}, X^{-2}, X^{-1}S, \ldots$ and Λ can similarly be set to $S^{-1}, X^{1/2}S^{-1/2}, S^{-2}, XS^{-1}, \ldots$. In this paper, we will mainly focus on two pairs of choices. The weighted least-squares adjustment (WLSA) is given by $\Sigma = X^{-1}$ and $\Lambda = S^{-1}$. The choices of $\Sigma = X^{-1/2}S^{1/2}$ and $\Lambda = X^{1/2}S^{-1/2}$ give rise to the jointly weighted least-squares adjustment (JWLSA).

Newton step adjustment Given an iterate (x, y, s) generated during the solution of d, the Newton step adjustment arises from taking a Newton step towards a point $(\tilde{x}, \tilde{y}, \tilde{s})$ that satisfies $\tilde{X}\tilde{S}e = XSe$. Therefore, this adjustment is given by the solution $(\Delta x, \Delta y, \Delta s)$ of the following Newton system:

$$\begin{aligned} \bar{A}\Delta x &= \Delta b - \Delta A x, \\ \bar{A}^T \Delta y + \Delta s &= \Delta c - \Delta A^T y, \\ X\Delta s + S\Delta x &= 0, \end{aligned}$$

where $\bar{A} := A + \Delta A$. Similarly to the family of least-squares adjustments, the first two equations ensure that $(\Delta x, \Delta y, \Delta s)$ satisfies (5a) and (5b), respectively. The third equation is obtained by linearizing the nonlinear equation $(X + \Delta X)(S + \Delta S)e = XSe$. If, in addition, $x + \Delta x > 0$ and $s + \Delta s > 0$, then the resulting iterate is deemed an acceptable starting solution for $d + \Delta d$.

This choice was originally proposed by Yıldırım and Todd [32], who developed an interior-point approach to sensitivity analysis in linear and semidefinite programming. We refer the reader to [31–33] for the relationship of the Newton step adjustment to the optimal partition approach to sensitivity analysis in nondegenerate LP problems, degenerate LP problems, and semidefinite programming problems, respectively.

The solution of the Newton step adjustment is given by

$$\Delta y = (\bar{A} D \bar{A}^T)^{-1}(\bar{A} D[\Delta c - \Delta A^T y] + \Delta b - \Delta A x), \tag{7a}$$

$$\Delta s = \Delta c - \Delta A^T y - \bar{A}^T \Delta y, \tag{7b}$$

$$\Delta x = -D \Delta s, \tag{7c}$$

where $D := XS^{-1}$.

2.3 Properties of the specific adjustments

In this subsection, we aim to motivate the specific choices of adjustments outlined in Sect. 2.2.2. We first describe several properties that need to be satisfied by an effective adjustment in the context of Algorithm 2.1. We then evaluate our specific choices with respect to these properties.

An effective adjustment in the context of Algorithm 2.1 should ideally have the following capabilities:

1. Given d and Δd, an effective adjustment should have the property that the number of times the main loop in Algorithm 2.1 is executed should decrease for smaller perturbations Δd. This implies that a fairly advanced iterate of the instance d can be used to compute an acceptable iterate for $d + \Delta d$ for a small perturbation Δd.
2. If an advanced iterate of d yields an acceptable iterate for $d + \Delta d$, then the resulting iterate should also be a relatively advanced point, which can, for instance, be quantified using the duality measure and infeasibility residuals. Roughly speaking, obtaining an advanced iterate would eliminate the computational effort that would be required to generate earlier iterates leading to a similar advanced point if $d + \Delta d$ were to be solved with cold start. Usually, the more advanced the warm-start is, the faster the perturbed instance $d + \Delta d$ can be solved.
3. In addition to obtaining an advanced iterate for $d + \Delta d$, it is also important that the resulting iterate be well-centered. IPMs may make very slow progress at an iterate (x, y, s) whose x and/or s components are close to the boundary of the nonnegative orthant since the barrier function rapidly blows up towards the boundary.
4. The computational cost of the adjustment should not be excessive. If a warm-start strategy succeeds in computing an advanced iterate for $d + \Delta d$, the reduction in the computational effort for reoptimization would be given by the number of IPM

iterations saved due to the warm-start strategy as opposed to cold start. In order for a warm-start strategy to be effective overall, the cost of computing a warm-start should not outweigh the computational gain resulting from the number of IPM iterations saved.

The question of finding an adjustment that would satisfy each of the four properties above is a nontrivial one. Consequently, developing effective warm-start strategies in IPMs is still an active area of research.

We stress that the properties outlined above are specific to the generic framework of Algorithm 2.1. In particular, this algorithm strives to find a near-feasible solution for the perturbed problem by computing a *single* adjustment to a near-feasible solution of the original problem. In other words, it aims to absorb the infeasibility arising from the perturbation Δd using only one adjustment. There exist several other warm-start strategies in which several adjustments (e.g., multiple centrality correctors) are allowed to obtain a near-feasible solution of the perturbed instance (see, e.g. [16]). Since our objective in this paper is to assess the practical performance of warm-start strategies developed in [34], we restrict our discussion to this particular framework.

2.3.1 Family of least-squares adjustments

The family of least-squares adjustments is driven by minimizing a certain norm of the Δx and Δs components of the adjustment $(\Delta x, \Delta y, \Delta s)$ while satisfying (5). Note that certain components of x and s go to zero as (x, y, s) tends to an optimal solution of the original instance d. Therefore, an advanced iterate (x, y, s) of d necessarily has the property that the x and s components are close to the boundary of the positive orthant in $\mathbb{R}^n$. In view of the first property above, it then makes sense to try to control the Δx and Δs components of the adjustment $(\Delta x, \Delta y, \Delta s)$ as the resulting iterate should satisfy $x + \Delta x > 0$ and $s + \Delta s > 0$ in order for it to be acceptable. This is precisely the motivation behind the family of least-squares adjustments.

We now consider several members of this family and evaluate them in terms of the properties outlined above. The plain least-squares adjustment (PLSA) is given by $\Sigma = \Lambda = I$, the identity matrix. This adjustment simply uses the Euclidean norm to measure the sizes of the Δx and Δs components of the adjustment $(\Delta x, \Delta y, \Delta s)$. For these choices of the scaling matrices Σ and Λ, we have $\bar{A}\Sigma^{-2}\bar{A}^T = \bar{A}\Lambda^2\bar{A}^T = \bar{A}\bar{A}^T$, where $\bar{A} := A + \Delta A$. It follows from (6) that it suffices to form and factorize $\bar{A}\bar{A}^T$ only once to compute the corresponding adjustment $(\Delta x, \Delta y, \Delta s)$. Furthermore, if the current adjustment fails to yield an acceptable solution of $d + \Delta d$, then the same factorization of $\bar{A}\bar{A}^T$ can be stored and reused to compute the adjustment corresponding to an earlier iterate of d. Therefore, the computational cost of the PLSA is given by the computation and factorization of a single $m \times m$ positive definite matrix and each adjustment in turn can be computed by a few matrix-vector multiplications. This is a major advantage of the PLSA in view of the fourth property outlined above.

On the other hand, the PLSA assigns an equal weight to each component of Δx and Δs. Since an advanced iterate (x, y, s) of the instance d necessarily has the property that some components of x and s are very close to zero, the PLSA is unlikely to yield an acceptable solution of $d + \Delta d$ for such iterates especially for larger perturbations Δd. In particular, the PLSA does not necessarily yield an adjustment

$(\Delta x, \Delta y, \Delta s)$ with the property that the $(\Delta x, \Delta s)$ components of the adjustment are comparable in size to those of (x, s). Therefore, using this adjustment, it may be necessary to retreat to a considerably earlier iterate to be able to compute an iterate satisfying $x + \Delta x > 0$ and $s + \Delta s > 0$ using a single adjustment, which may adversely affect the potential benefit of using a warm-start strategy. Therefore, the PLSA may not necessarily satisfy the first property above.

The weighted least-squares adjustment (WLSA) given by $\Sigma = X^{-1}$ and $\Lambda = S^{-1}$ and the jointly weighted least-squares adjustment (JWLSA) given by $\Sigma = X^{-1/2}S^{1/2}$ and $\Lambda = X^{1/2}S^{-1/2}$ are primarily considered in an attempt to circumvent this drawback of the PLSA. While the WLSA separately uses only the primal information in the computation of Δx and only the dual information in Δs, the JWLSA combines the primal and dual information in computing the adjustment. For each of these two members, note that the diagonal scaling matrices Σ and Λ are chosen as a function of the current iterate (x, y, s). Instead of using the usual Euclidean norm, both of these members rely on an ellipsoidal norm to measure the sizes of the Δx and Δs components of the adjustment $(\Delta x, \Delta y, \Delta s)$. Indeed, an advanced iterate (x, y, s) of d has the property that certain components of x (s) are bounded away from zero while the corresponding components of s (x) tend to zero. The diagonal scaling matrices are chosen in order to ensure that both of these adjustments penalize large components of Δx and Δs corresponding to the small components of x and s, respectively. Therefore, these adjustments are more likely to satisfy the first property above in comparison with the PLSA.

In contrast with the PLSA, the computation of the adjustment based on the current iterate (x, y, s) has the major disadvantage of having to compute and factorize $\bar{A}\Sigma^{-2}\bar{A}^T$ and $\bar{A}\Lambda^2\bar{A}^T$ anew for each iterate. For the WLSA, one needs to compute and factorize two $m \times m$ matrices. On the other hand, since $\bar{A}\Sigma^{-2}\bar{A}^T = \bar{A}\Lambda^2\bar{A}^T = \bar{A}XS^{-1}\bar{A}^T$ for the JWLSA, it suffices to compute and factorize only one $m \times m$ positive definite matrix. Therefore, the computational cost of the WLSA is roughly twice the cost of the JWLSA for each adjustment. This suggests that both of these adjustments may fail to satisfy the fourth property above especially for larger perturbations in comparison with the PLSA as they may require the computation of these factorizations for several iterates of the original instance d. This observation indicates one of the difficulties of designing an adjustment satisfying all of the four desirable properties above.

Under the assumption that the original iterate (x, y, s) is feasible and well-centered, it is possible to obtain upper bounds on the duality measure and on the proximity to the central path of the iterate arising from an adjustment in this family based on the duality measure and on the proximity to the central path of the original iterate especially for smaller perturbations [34]. However, since neither of these assumptions is enforced in practice, one usually has no control over how well-centered or how advanced the resulting iterate will be. Therefore, this family of adjustments in general may not necessarily satisfy the second and third properties above.

2.3.2 Newton step adjustment

We now consider the Newton step adjustment given by (7). As stated in Sect. 2.2.2, starting with an iterate (x, y, s) of the original instance d, the Newton step adjustment

arises from taking a Newton step towards a point $(\tilde{x}, \tilde{y}, \tilde{s})$ that satisfies $\tilde{X}\tilde{S}e = XSe$. Developed originally in the context of sensitivity analysis by Yıldırım and Todd [32], this adjustment is driven by the following observation. An advanced feasible iterate (x, y, s) of d has the property that the componentwise products of x and s are close to zero. By aiming towards a point $(\tilde{x}, \tilde{y}, \tilde{s})$ satisfying the same componentwise products of $\tilde{x}$ and $\tilde{s}$, one intends to compute a near-optimal point for $d + \Delta d$ starting from a near-optimal point of d since $\mu := x^T s/n = \tilde{x}^T \tilde{s}/n$. In fact, simple necessary and sufficient conditions on Δd have been established in order for the resulting point to satisfy $x + \Delta x > 0$ and $s + \Delta s > 0$ [32, 34]. Furthermore, if the Newton step adjustment yields an acceptable point for $d + \Delta d$, it has the appealing property that the duality measure of the resulting iterate is bounded above by that of the original one [32, 34]. Therefore, in contrast with the family of least-squares adjustments, the Newton step adjustment necessarily satisfies the second property above. This is one of the main motivations to consider such an adjustment in the context of warm-start strategies.

It follows from (6) and (7) that the Newton step adjustment is somewhat related to the jointly weighted least-squares adjustment (JWLSA). Both of the adjustments require the computation and factorization of the same $m \times m$ matrix $\bar{A}XS^{-1}\bar{A}^T$. While the JWLSA computes the primal adjustment Δx using only ΔA and Δb and the dual adjustment $(\Delta y, \Delta s)$ using only ΔA and Δc, each component of the Newton step adjustment is a function of the entire perturbation Δd. In fact, for each of the two strategies, the dual adjustments $(\Delta y, \Delta s)$ coincide if $\Delta A = 0$ and $\Delta b = 0$ and the primal adjustments Δx are identical if $\Delta A = 0$ and $\Delta c = 0$ for a given iterate (x, y, s) of d. The computational cost of the Newton step adjustment is therefore similar to that of the JWLSA. Therefore, similar remarks apply in terms of the fourth property.

We also remark that a similar Newton system is employed in [16]. In contrast with the Newton step adjustment, their approach ignores the dual (primal) feasibility in the computation of the primal (dual) adjustment. It then follows from the observation in the preceding paragraph that the adjustment of [16] precisely coincides with the JWLSA for perturbations with $\Delta A = 0$.

In contrast with the JWLSA, the Newton step adjustment may not necessarily satisfy the first property since the primal and dual adjustments are no longer decoupled. In fact, it follows from the results of [32, 33] that the first property is satisfied if and only if the optimal partition of the perturbed instance $d + \Delta d$ coincides with that of d. Clearly, for general perturbations, this assumption may not hold.

Similarly to the family of least-squares adjustments, one can obtain upper bounds on the proximity to the central path of the resulting iterate in terms of the proximity to the central path of the original iterate if it is feasible and well-centered [34]. Since these assumptions may not necessarily be satisfied in practice, the Newton step adjustment does not have any guarantees on the proximity of the resulting iterate to the central path of the perturbed instance. Therefore, this adjustment may not necessarily satisfy the third property.

In conclusion, our choices of specific adjustments seem to satisfy certain subsets of the four desirable properties an effective adjustment is expected to have. By comparing the performances of these specific adjustments, our computational results in

this paper will help to shed more light into the significance of the role played by each of the aforementioned four properties.

3 Implementation

3.1 An overview of PCx

We used PCx to implement Algorithm 2.1 using the adjustments described in Sect. 2.2.2. PCx is an infeasible primal-dual path-following interior-point solver developed by Czyzyk, Mehrotra, Wagner, and Wright [5]. It implements Mehrotra's predictor-corrector algorithm [21] and the higher-order correction strategy of Gondzio [14]. Most of the code is written in C and the solution of the normal equations arising at each IPM iteration is obtained by a call to the Cholesky factorization package of Ng and Peyton [26], which is written in Fortran77. The source code of PCx and the linear algebra routines of Ng and Peyton are freely available for research use at the PCx web site.[2]

PCx accepts as input any LP problem that can be specified in the MPS format. Given an instance d in this format, PCx reduces it to a standard, simpler formulation and sends it to the presolver, which employs the techniques proposed by Andersen and Andersen [1]. PCx next applies the row and column scaling technique of Curtis and Reid [4] to minimize the variation of the nonzero elements in the coefficient matrix. These steps ensure that an equivalent but simpler form of the LP problem is passed to the solver.

After preprocessing and scaling operations, the reduced LP problem, which is stored in `ReducedLPtype`, contains equality constraints and nonnegative variables in addition to variables with finite positive upper bounds. Since the warm-start strategies are specified for LP problems in standard form given by $(\mathcal{P})$ and $(\mathcal{D})$, we have absorbed the bound constraints into the coefficient matrix by introducing slack variables. Note that this operation may considerably enlarge the coefficient matrix. However, the new coefficient matrix has the special structure that each of the new rows has only two nonzero entries and that each new column has only one nonzero entry. The linear algebra routines employed in PCx are capable of exploiting this special structure to aid in the factorization of the matrices arising in the normal equations. This modification allows us to universally apply the warm-start strategies to any LP problem. The resulting formulation is stored in `ReducedLPtype_NB`, which is then sent to the solver.

Apart from simple other modifications required by our experiments, we used the default parameters of PCx in our computational experiments. We chose the software package PCx to implement our warm-start strategies because it offers a simple interface to the solver, a modular structure that is easy to modify for our purposes, and compatibility with various platforms.

[2] http://www-fp.mcs.anl.gov/otc/Tools/PCx/.

3.2 Preserving the dimension

Note that the warm-start strategies described in this paper apply to the case in which the perturbed LP problem has precisely the same dimension as the original one. All LP solvers use preprocessing to simplify a given LP problem before it is sent to the solver. Among other advantages, the preprocessing stage helps to detect infeasibility, eliminates redundancy in the problem, and is used to feed the solver an LP problem in a certain, prespecified form which streamlines the code by eliminating the need to write different solvers for problems in different forms.

In general, preprocessing leads to addition of new constraints and/or variables and deletion of some of the original constraints and/or variables. Therefore, the simplified LP problem usually has a different dimension from that of the original one. If the user inputs an LP problem and a slightly perturbed version of it into an LP solver, it is likely that the simplified versions that are sent to the solver may not only look quite different from one another but may even have different dimensions. Such a situation may arise, for instance, if the original instance has redundant constraints. It may happen that the corresponding constraints in the perturbed problem are no longer redundant. In such a case, our warm-start strategies are not applicable.

One way to get around this problem is to turn off preprocessing. Our experiments indicated that this operation adversely affects the performance of the code by causing numerical instabilities. Therefore, given an LP problem, we treated the fully reduced version of it stored in `ReducedLPtype_NB` as the original instance. The perturbed instance was obtained by perturbing the data of this reduced form. We have therefore ensured that both the original and the perturbed instances have precisely the same dimensions.

We stress that the LP instance obtained by perturbing the fully reduced version stored in `ReducedLPtype_NB` may look entirely different from the reduced version of a perturbation of the original LP problem. Therefore, our modification does not necessarily yield a general-purpose code that can effectively implement a warm-start strategy for an arbitrary perturbation of an LP problem except for the case that the original dimension is preserved. In fact, such a general-purpose code should also contain warm-start strategies for the case in which the dimension of the perturbed LP problem may differ from that of the original one. Rather than writing a general purpose warm-start code, our main objective in this paper is to experimentally assess the practical performance of warm-start strategies in the framework of Algorithm 2.1. Therefore, we are content with perturbing the reduced version of the LP problem for the purposes of our computational experiments in order to ensure that the original and perturbed LP problems both have the same dimensions.

3.3 Generating perturbed instances

We have considered four types of perturbations in our experiments: (i) b only, (ii) c only, (iii) b and c only, and (iv) A, b, and c.

Given an LP instance, we treated its reduced version stored in `ReducedLP-type_NB` as the original instance d as explained in Sect. 3.2. For each component κ of the original instance d to be perturbed, we generated a random number γ distributed uniformly in $[-1, 1]$ and the corresponding component of Δd was set to $\gamma|\kappa|$.

This scheme enabled us to allow perturbations that are comparable in size to the original problem data. In our experiments, only nonzero components of d are perturbed, which ensures that both the original LP problem and the perturbed one have identical sparsity patterns. In order to evaluate the performance of warm-start strategies with respect to the size of the perturbations, we considered a family of perturbed instances given by $d + \alpha \Delta d$. In our experiments, we used $\alpha \in \{.01, .1, 1, 10\}$. We have not taken any care to ensure feasibility of the perturbed instances.

3.4 Methods of comparison

We have used two performance measures to assess the effectiveness of our warm-start strategies. The first measure is the number of interior-point iterations. For each perturbed instance, we compare the number of iterations required by cold start versus that required in the reoptimization stage of Algorithm 2.1 after successfully computing a warm-start. This performance measure provides information about the savings in interior-point iterations in reoptimization due to the use of a warm-start strategy. Note that we only consider the interior-point iterations in the reoptimization stage of Algorithm 2.1 while ignoring the number of times the main loop is executed in the search stage. This part is taken into account in computing the CPU time as explained in the following paragraph.

The second performance measure is the CPU time. Note that our warm-start strategies consist of two stages, namely the search stage and the reoptimization stage (cf. Algorithm 2.1). The new timer functions integrated into PCx provide us with separate timing information for each of these two components.

We have exercised care to ensure a fair and meaningful timing comparison between warm-start and cold start. When PCx solves an LP instance using cold start, it uses Mehrotra's heuristic [21] to compute a starting point. In computing this point, the code performs various operations and factorizations on the coefficient matrix A such as column reordering. This information is stored and then passed to the rest of the code along with the starting point. In our experiments, we measured the solution time of cold start starting precisely at this stage. Incidentally, our warm-start strategies also require similar operations on the coefficient matrix during the search of a warm-start. Therefore, this information is also passed to the rest of the code along with the warm-start in our implementation. Similarly, the solution time of the warm-start was measured starting at this stage. As a result, neither method was required to compute any more factorizations than the other.

3.5 Further details

In our implementation of Algorithm 2.1, we set $\mathcal{T} = \{0, 1, \ldots, l\}$, i.e., we stored *all* iterates generated during the course of the solution of the original instance d. While this choice may significantly increase the search time for a warm-start for the perturbed instance, we aimed to identify the most advanced iterate of d that would yield a successful warm-start. Moreover, this strategy enabled us to gain insight into the relationship between the size of the perturbation and the order of the index of the particular iterate that leads to a successful warm-start. We believe that this relationship is important in order to assess the quality of the theoretical bounds provided in [34].

We used the linear algebra routines of Ng and Peyton [26] in PCx to perform the computations (6) and (7). All experiments were carried out on a 1.33 GHz Pentium M processor with 512 MB RAM running Windows XP.

4 Computational results

In this section, we report and discuss our computational results. Each of the 93 LP instances in the NETLIB suite was initially solved using PCx. After preprocessing, the instance was converted into standard form after eliminating the upper bounds. The sizes of the reduced instances vary from (27/51) for `afiro` to (10505/21024) for `fit2d`, where $(\cdot/\cdot)$ denotes the number of rows and columns, respectively. The solution time ranges from the fraction of a second for `afiro` (27/51) to about 1100 seconds for `dfl001` (6084/12243). These "reduced" instances were treated as the "unperturbed" or "original" LP instances. For each such instance d, four different types of perturbations given by $\Delta d_1 = (0, \Delta b, 0)$, $\Delta d_2 = (0, 0, \Delta c)$, $\Delta d_3 = (0, \Delta b, \Delta c)$, and $\Delta d_4 = (\Delta A, \Delta b, \Delta c)$ were generated. Next, each such perturbation was scaled by $\alpha = .01, .1, 1, 10$. Therefore, for each original instance, 16 different perturbed instances were generated. On each perturbed instance, we implemented each of the four warm-start strategies. We also solved each perturbed instance using cold start (i.e., the default initial iterate given by Mehrotra's heuristic in PCx). This experimental setting allowed us to compare the number of iterations and the computation time using our warm-start strategies versus cold start.

Since we have not exercised any care to ensure the feasibility of perturbed LP instances, the phrase "solving the perturbed instance" is used to refer to either computing an optimal solution or detecting unboundedness or infeasibility. By not ensuring feasibility of the perturbed instance, we aimed to gain insight into whether warm-start strategies can also be used to effectively detect infeasibility of the perturbed instance in comparison with cold start.

4.1 Iteration comparison

We first compare the number of iterations needed to resolve the perturbed LP instance using our warm-start strategies versus cold start. The results are presented in Table 1, which is divided into two parts. The upper part reports the results of perturbations of b only and of c only and the lower part contains the results of perturbations of b and c only and of A, b, and c. Each part consists of four sets of rows corresponding to four different warm-start strategies. Table 1 is also divided into four sets of columns. The first column lists the particular warm-start strategy employed. We use PLSA for the plain least-squares adjustment, WLSA for the weighted least-squares adjustment, JWLSA for the jointly weighted least-squares adjustment, and NSA for the Newton step adjustment. The second column presents the outcome of the comparison of number of iterations. To this end, we define ρ_i to be the ratio of the number of interior-point iterations in the reoptimization stage of Algorithm 2.1 using a warm-start strategy to the number of iterations using cold start. Each row corresponds to an interval into which the value of this ratio ρ_i falls. We used three critical values of .5, 1,

Table 1 Iteration comparison of four warm-start strategies on four different types of perturbations

WS strategy	Iter. comp.	Perturbations of b				Perturbations of c			
		$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	$\rho_i \leq .5$	17.20	5.38	2.15	1.08	27.96	18.28	11.83	11.83
	$.5 < \rho_i \leq 1$	82.80	92.47	83.87	94.62	72.04	78.49	81.72	87.10
	$1 < \rho_i \leq 1.5$	0	2.15	12.90	4.30	0	3.23	6.45	1.08
	$1.5 < \rho_i$	0	0	1.08	0	0	0	0	0
WLSA	$\rho_i \leq .5$	80.65	45.16	6.45	2.15	83.87	50.54	25.81	13.98
	$.5 < \rho_i \leq 1$	17.20	54.84	87.10	95.70	16.13	47.31	73.12	84.95
	$1 < \rho_i \leq 1.5$	0	0	4.30	2.15	0	2.15	1.08	1.08
	$1.5 < \rho_i$	2.15	0	2.15	0	0	0	0	0
JWLSA	$\rho_i \leq .5$	81.72	43.01	7.53	2.15	81.72	53.76	26.88	13.98
	$.5 < \rho_i \leq 1$	16.13	54.84	86.02	95.70	18.28	46.24	69.89	84.95
	$1 < \rho_i \leq 1.5$	0	1.08	5.38	2.15	0	0	3.23	1.08
	$1.5 < \rho_i$	2.15	1.08	1.08	0	0	0	0	0
NSA	$\rho_i \leq .5$	77.42	36.56	5.38	2.15	75.27	46.24	22.58	11.83
	$.5 < \rho_i \leq 1$	16.13	59.14	88.17	96.77	24.73	52.69	75.27	88.17
	$1 < \rho_i \leq 1.5$	1.08	0	4.30	1.08	0	1.08	2.15	0
	$1.5 < \rho_i$	5.38	4.30	2.15	0	0	0	0	0
WS strategy	Iter. comp.	Perturbations of b and c				Perturbations of A, b, and c			
		$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	$\rho_i \leq .5$	15.05	4.30	1.08	0	0	0	0	0
	$.5 < \rho_i \leq 1$	82.80	92.47	91.40	98.92	97.85	98.92	100	100
	$1 < \rho_i \leq 1.5$	1.08	2.15	7.53	1.08	2.15	1.08	0	0
	$1.5 < \rho_i$	1.08	1.08	0	0	0	0	0	0
WLSA	$\rho_i \leq .5$	69.89	35.48	2.15	0	39.78	7.53	0	0
	$.5 < \rho_i \leq 1$	29.03	64.52	96.77	98.92	60.22	90.32	100	100
	$1 < \rho_i \leq 1.5$	0	0	1.08	1.08	0	2.15	0	0
	$1.5 < \rho_i$	1.08	0	0	0	0	0	0	0
JWLSA	$\rho_i \leq .5$	66.67	29.03	3.23	0	38.71	6.45	0	0
	$.5 < \rho_i \leq 1$	33.33	70.97	93.55	100	60.22	92.47	98.92	100
	$1 < \rho_i \leq 1.5$	0	0	3.23	0	1.08	1.08	1.08	0
	$1.5 < \rho_i$	0	0	0	0	0	0	0	0
NSA	$\rho_i \leq .5$	62.37	21.51	0	0	34.41	5.38	0	0
	$.5 < \rho_i \leq 1$	33.33	74.19	91.40	100	64.52	92.47	100	100
	$1 < \rho_i \leq 1.5$	0	1.08	7.53	0	1.08	2.15	0	0
	$1.5 < \rho_i$	4.30	3.23	1.08	0	0	0	0	0

and 1.5. For each warm-start strategy and each perturbation type, we computed the percentage of 93 LP instances for which $\rho_i \leq .5$ (warm-start is "much better" than cold start), $.5 < \rho_i \leq 1$ (warm-start is "better" than cold start), $1 < \rho_i \leq 1.5$ (warm-start is "worse" than cold start), and $1.5 < \rho_i$ (warm-start is "much worse" than cold start). We reported these percentages in the corresponding rows. The third and fourth sets of columns present the results for different values of the scaling factor α used to compute the perturbed instance for each of the four types of perturbations. For example, for perturbations of b with $\alpha = .01$, the plain least-squares adjustment was "much better" than cold start on 17.20% of the instances and "better" on the remaining 82.80% of the instances.

A careful examination of Table 1 reveals that each of the four warm-start strategies usually performed at least as well as cold start for all four types of perturbations and for all four values of the scaling factor α. More specifically, the percentages reported in the last two rows of each warm-start strategy are either small or equal to zero.

For a fixed warm-start strategy and a fixed perturbation type, Table 1 illustrates that the performance of the warm-start strategy usually degrades for larger values of the scaling factor α. This is indicated by the fact that the percentages in each set of rows tend to shift from the first row ("much better") to the third and fourth rows ("worse" and "much worse") as α increases from .01 to 10. This is an expected behavior as larger perturbations lead to an increased distance between the original instance and the perturbed one. In such situations, the advantages of warm-start strategies are less pronounced.

For a fixed warm-start strategy and a fixed value of the scaling factor α, Table 1 indicates that the performance of a warm-start strategy usually degrades as more data components are perturbed. For instance, while the jointly weighted least-squares adjustment is much better than cold start on 81.72% of the instances for perturbations of b and of c with $\alpha = .01$, this percentage reduces to 66.67% for perturbations of b and c and to 38.71% for perturbations of A, b, and c.

In Table 2, we report the cumulative iteration comparison of the warm-start strategies. For each warm-start strategy, we report the ratio of the total number of interior-point iterations in the reoptimization stage of all the perturbed instances using that particular strategy to the total number of iterations using cold start. Therefore, Table 2 summarizes overall savings in terms of the number of iterations as a result of using warm-start strategies. For instance, the JWLSA requires only 32% of the number of iterations generated by cold start for perturbations of b with $\alpha = .01$, which translates into a 68% reduction in the number of iterations. The results presented in Table 2 also support our previous observations. Generally, for each warm-start strategy, the savings diminish as more data components are perturbed and as the scaling factor α increases. Comparing the different warm-start strategies for a fixed perturbation type and a fixed value of the scaling factor α, we see that the WLSA and the JWLSA usually yield the largest savings. The NSA has a slightly worse performance than these two strategies. The PLSA usually results in the smallest savings among the warm-start strategies. These results are in support of our previous observations in Sect. 2.3. In computing the adjustment, the PLSA does not distinguish between small and large components of an iterate of the original instance while all the other three strategies somehow take this disparity into account. Therefore, the PLSA usually computes a successful warm-start at a rather early iterate of the original instance

Table 2 Cumulative iteration comparison of four warm-start strategies on four different types of perturbations

WS strategy	Perturbations of b				Perturbations of c			
	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	.67	.84	.93	.95	.59	.73	.84	.84
WLSA	.33	.50	.84	.93	.28	.46	.66	.81
JWLSA	.32	.52	.83	.93	.29	.45	.65	.82
NSA	.38	.60	.89	.93	.33	.51	.71	.84
WS strategy	Perturbations of b and c				Perturbations of A, b, and c			
	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	.74	.89	.97	1.00	.98	1.00	1.00	1.00
WLSA	.41	.60	.90	.98	.64	.86	1.00	1.00
JWLSA	.41	.60	.91	.98	.67	.87	1.00	1.00
NSA	.48	.71	.97	1.00	.69	.88	1.00	1.00

and therefore requires a larger number of iterations in the reoptimization stage in comparison with the other three adjustments on a given perturbed instance.

4.2 Performance of the search stage

Note that the generic warm-start algorithm has two stages (cf. Algorithm 2.1). In the search stage, the algorithm searches for an acceptable starting iterate for the perturbed instance by computing adjustments to iterates of the original instance. In this subsection, we analyze the performances of each of the four adjustments in the search stage in an attempt to gain insight into the relationship between the number of the iterations in the reoptimization stage and the order of the iterate of the original instance that yields a successful warm-start in the search stage.

Ideally, a warm-start strategy should use a relatively advanced iterate of the unperturbed problem to compute a successful warm-start especially for smaller perturbations and then reoptimize the perturbed instance in a relatively small number of iterations in comparison with cold start. In the theoretical framework of [34], sufficient conditions are established on the duality measure of an iterate of the original instance to ensure that a specific adjustment yields a successful warm-start for a perturbed instance. These bounds indicate that a sufficiently advanced iterate can be used to generate a warm-start especially for smaller perturbations. Moreover, resolving the perturbed problem starting from such a warm-start leads to reduced iteration complexity. These results are established under the assumptions that the original iterates are feasible and somewhat well-centered. In our experimental setting, we enforce neither of these two assumptions. Therefore, we would like to investigate whether a similar relationship continues to hold without these assumptions.

PCx indexes the iterates generated during the solution of an LP instance starting from zero, which corresponds to the initial point generated using Mehrotra's heuristic (i.e., cold start). Therefore, if the instance is solved in l iterations, the iterates are numbered $0, \ldots, l$. If the search stage of Algorithm 2.1 succeeds in computing a warm-start for a perturbed instance, we then let t denote the index of the iterate of the original instance yielding this warm-start, where $0 \leq t \leq l$. In this case, we define the ratio $\rho_a = t/l \in [0, 1]$ in order to measure how advanced the iterate t is. We say that the search stage identifies an "advanced" iterate of the original instance to generate a successful warm-start if $.75 \leq \rho_a \leq 1$, a "fairly advanced" iterate if $.5 \leq \rho_a < .75$, an "intermediate" iterate if $.25 \leq \rho_a < .5$, and an "early" iterate if $0 \leq \rho_a < .25$. Otherwise, the search stage fails to identify a successful warm-start and Algorithm 2.1 reverts to cold start to solve the perturbed instance.

Using this classification scheme, we report the performances of the search stage of each of the four warm-start strategies in Table 3. We use precisely the same reporting scheme as in Table 1. For instance, the search stage of the PLSA identifies an advanced iterate on 2.15% of the instances, a fairly advanced iterate on 13.98% of the instances, an intermediate iterate on 54.84% of the instances, an early iterate on 29.03% of the instances, and reverts to cold start on none of the instances for perturbations of b with $\alpha = .01$.

A careful examination of Table 3 reveals similar observations to those arising from the analysis of Table 1 and Table 2. For a fixed warm-start strategy and a fixed perturbation type, the performance of the warm-start strategy in the search stage degrades for larger values of the perturbation parameter α as indicated by the percentages shifting towards last two rows. Similarly, for a fixed warm-start strategy and a fixed perturbation parameter α, the performance of the search stage deteriorates as more data components are perturbed. We remark that these observations are in line with the theoretical findings in [34] despite the fact that the assumptions of feasibility and proximity to the central path are not necessarily satisfied in our setting.

Table 3 clearly indicates that the advantage of using warm-start strategies tends to disappear for larger perturbations and for perturbations of more data components. For instance, all strategies revert to cold start on the vast majority of instances for perturbations of A, b, and c with $\alpha = 10$.

In terms of the capability of identifying an advanced iterate of the original instance to generate a successful warm-start, we conclude that the WLSA and JWLSA usually exhibit the best performance. The NSA has a slightly worse performance. The PLSA exhibits the least success among the other strategies. Note that these observations are very similar to those arising from Table 2.

We now stress the strong correlation between the results of Table 3 and those of Table 1. For each strategy, if we consider the sum of the percentages in the rows corresponding to the "advanced" and "fairly advanced" rows in Table 3 for a given perturbation type and a perturbation parameter α, we see that this sum is fairly close to the corresponding percentage in the first row of Table 1. For instance, Table 3 indicates that the PLSA identifies an advanced iterate on about 2% of the instances and a fairly advanced iterate on about 14% of the instances for perturbations of b and $\alpha = .01$, whose sum is about 16%. This implies that the PLSA identifies an iterate from the second half of the iteration sequence on about 16% of the instances.

Table 3 Comparison of the performances of the four warm-start strategies in the search stage on four different types of perturbations

WS strategy	Iter. used	Perturbations of b				Perturbations of c			
		$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	Advanced	2.15	2.15	1.08	1.08	13.98	12.90	11.83	11.83
	Fairly adv.	13.98	2.15	2.15	0	11.83	4.30	0	0
	Intermediate	54.84	7.53	0	1.08	41.94	10.75	3.23	0
	Early	29.03	88.17	88.17	4.30	32.26	69.89	38.71	0
	Cold start	0	0	8.60	93.55	0	2.15	46.24	88.17
WLSA	Advanced	47.31	25.81	5.38	2.15	46.24	29.03	23.66	13.98
	Fairly adv.	30.11	18.28	1.08	0	36.56	21.51	1.08	1.08
	Intermediate	18.28	30.11	8.60	0	15.05	30.11	15.05	1.08
	Early	4.30	25.81	74.19	6.45	2.15	19.35	25.81	2.15
	Cold start	0	0	10.75	91.40	0	0	34.41	81.72
JWLSA	Advanced	48.39	25.81	5.38	2.15	45.16	29.03	24.73	13.98
	Fairly adv.	29.03	16.13	1.08	0	36.56	21.51	1.08	1.08
	Intermediate	18.28	31.18	9.68	0	16.13	31.18	18.28	1.08
	Early	4.30	26.88	73.12	5.38	2.15	18.28	21.51	0
	Cold start	0	0	10.75	92.47	0	0	34.41	83.87
NSA	Advanced	46.24	18.28	4.30	2.15	41.94	24.73	20.43	11.83
	Fairly adv.	29.03	19.35	0	0	33.33	19.35	2.15	0
	Intermediate	19.35	35.48	10.75	0	20.43	31.18	11.83	0
	Early	5.38	26.88	70.97	3.23	4.30	23.66	24.73	0
	Cold start	0	0	13.98	94.62	0	1.08	40.86	88.17
WS strategy	Iter. used	Perturbations of b and c				Perturbations of A, b, and c			
		$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	Advanced	0	0	0	0	0	0	0	0
	Fairly adv.	11.83	2.15	1.08	0	1.08	0	0	0
	Intermediate	41.94	4.30	0	0	1.08	0	0	0
	Early	46.24	91.40	46.24	1.08	24.73	11.83	1.08	1.08
	Cold start	0	2.15	52.69	98.92	73.12	88.17	98.92	98.92
WLSA	Advanced	30.11	17.20	2.15	0	13.98	1.08	0	0
	Fairly adv.	38.71	13.98	1.08	0	25.81	3.23	0	0
	Intermediate	24.73	35.48	3.23	1.08	30.11	22.58	0	0
	Early	6.45	33.33	50.54	2.15	18.28	38.71	1.08	1.08
	Cold start	0	0	43.01	96.77	11.83	34.41	98.92	98.92
JWLSA	Advanced	31.18	17.20	2.15	0	12.90	1.08	0	0
	Fairly adv.	35.48	12.90	1.08	0	24.73	4.30	0	0
	Intermediate	26.88	36.56	3.23	1.08	32.26	20.43	0	0
	Early	6.45	33.33	48.39	1.08	18.28	40.86	2.15	1.08
	Cold start	0	0	45.16	97.85	11.83	33.33	97.85	98.92

Table 3 (*Continued*)

WS strategy	Iter. used	Perturbations of b and c				Perturbations of A, b, and c			
		$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
NSA	Advanced	29.03	8.60	0	0	8.60	0	0	0
	Fairly adv.	31.18	12.90	0	0	24.73	3.23	0	0
	Intermediate	30.11	38.71	4.30	0	31.18	16.13	0	0
	Early	9.68	38.71	45.16	1.08	22.58	44.09	1.08	1.08
	Cold start	0	1.08	50.54	98.92	12.90	36.56	98.92	98.92

On the other hand, the PLSA is "much better" than cold start on about 17% of the instances (cf. Table 1), i.e., the PLSA solves the perturbed instance in less than half of the number of iterations required by cold start. This observation reveals the close connection between the ability to identify an advanced warm-start and the reduction in the number of iterations to reoptimize the perturbed instance in comparison with cold start.

Finally, we point out the connection between Table 2 and Table 3. Clearly, if the search stage fails to compute a successful warm-start, then the number of iterations in the reoptimization stage is equal to the number of iterations using cold start. The cells in Table 2 with a value of 1.00 usually correspond to the cases in which the strategy reverted to cold start on the majority of the instances. Note that this case usually happens for larger values of the scaling factor α and for perturbations of more data components.

4.3 Time comparison

We next compare warm-start strategies and cold start in terms of the computation time. Recall that the generic warm-start algorithm has two stages (cf. Algorithm 2.1). In the search stage, the algorithm searches for an appropriate starting iterate for the perturbed instance by computing adjustments to iterates of the original instance. Therefore, each warm-start strategy requires some time to identify an appropriate starting iterate for the perturbed instance. We refer to this as the "search time." Once such an iterate has been identified, the perturbed instance is solved starting from it. The time spent in the reoptimization stage is referred to as the "reoptimization time." Therefore, the overall computation time of a warm-start strategy is obtained by summing up these two components.

In Table 4, we report the timing comparison using the same reporting scheme as in Table 1. We use ρ_t to denote the ratio of the total computation time (i.e., the sum of the search time and the reoptimization time) required by a warm-start strategy to the solution time of the perturbed instance using cold start. Note that the solution time of cold start only includes the actual solution stage and excludes pre- and post-processing. We employ the same threshold values of .5, 1, and 1.5 for ρ_t. The results are tabulated in percentages. For example, the PLSA is "much better" than cold start on about 12% of the instances for perturbations of b using $\alpha = .01$.

The observations arising from a careful analysis of Table 4 are in general similar to those resulting from Table 1. For a fixed warm-start strategy, the performance

Table 4 Timing comparison of four warm-start strategies on four different types of perturbations

WS strategy	Time comp.	Perturbations of b				Perturbations of c			
		$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	$\rho_t \leq .5$	11.83	3.23	2.15	3.23	23.66	15.05	11.83	10.75
	$.5 < \rho_t \leq 1$	77.42	78.49	59.14	26.88	75.27	69.89	45.16	18.28
	$1 < \rho_t \leq 1.5$	9.68	15.05	30.11	64.52	1.08	12.90	40.86	64.52
	$1.5 < \rho_t$	1.08	3.23	8.60	5.38	0	2.15	2.15	6.45
WLSA	$\rho_t \leq .5$	35.48	19.35	4.30	2.15	33.33	24.73	19.35	10.75
	$.5 < \rho_t \leq 1$	24.73	12.90	5.38	5.38	35.48	16.13	4.30	4.30
	$1 < \rho_t \leq 1.5$	23.66	45.16	4.30	3.23	25.81	30.11	11.83	3.23
	$1.5 < \rho_t$	16.13	22.58	86.02	89.25	5.38	29.03	64.52	81.72
JWLSA	$\rho_t \leq .5$	39.78	19.35	5.38	2.15	41.94	27.96	21.51	11.83
	$.5 < \rho_t \leq 1$	38.71	25.81	5.38	6.45	44.09	33.33	10.75	3.23
	$1 < \rho_t \leq 1.5$	15.05	35.48	22.58	3.23	11.83	33.33	29.03	7.53
	$1.5 < \rho_t$	6.45	19.35	66.67	88.17	2.15	5.38	38.71	77.42
NSA	$\rho_t \leq .5$	39.78	13.98	3.23	3.23	37.63	24.73	17.20	10.75
	$.5 < \rho_t \leq 1$	39.78	34.41	10.75	5.38	41.94	33.33	11.83	4.30
	$1 < \rho_t \leq 1.5$	13.98	35.48	19.35	5.38	17.20	34.41	29.03	7.53
	$1.5 < \rho_t$	6.45	16.13	66.67	86.02	3.23	7.53	41.94	77.42
WS strategy	Time comp.	Perturbations of b and c				Perturbations of A, b, and c			
		$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	$\rho_t \leq .5$	10.75	1.08	0	1.08	0	1.08	1.08	2.15
	$.5 < \rho_t \leq 1$	77.42	73.12	38.71	26.88	19.35	16.13	11.83	17.20
	$1 < \rho_t \leq 1.5$	9.68	20.43	55.91	68.82	74.19	77.42	72.04	48.39
	$1.5 < \rho_t$	2.15	5.38	5.38	3.23	6.45	5.38	15.05	32.26
WLSA	$\rho_t \leq .5$	19.35	10.75	0	0	6.45	2.15	0	1.08
	$.5 < \rho_t \leq 1$	33.33	12.90	8.60	3.23	24.73	6.45	4.30	4.30
	$1 < \rho_t \leq 1.5$	26.88	24.73	3.23	3.23	25.81	16.13	3.23	2.15
	$1.5 < \rho_t$	20.43	51.61	88.17	93.55	43.01	75.27	92.47	92.47
JWLSA	$\rho_t \leq .5$	27.96	10.75	0	0	11.83	1.08	0	1.08
	$.5 < \rho_t \leq 1$	43.01	24.73	10.75	5.38	32.26	12.90	6.45	5.38
	$1 < \rho_t \leq 1.5$	22.58	39.78	11.83	5.38	35.48	41.94	20.43	13.98
	$1.5 < \rho_t$	6.45	24.73	77.42	89.25	20.43	44.09	73.12	79.57
NSA	$\rho_t \leq .5$	23.66	10.75	0	0	9.68	3.23	1.08	1.08
	$.5 < \rho_t \leq 1$	49.46	17.20	8.60	6.45	36.56	13.98	4.30	6.45
	$1 < \rho_t \leq 1.5$	20.43	47.31	13.98	2.15	34.41	40.86	25.81	13.98
	$1.5 < \rho_t$	6.45	24.73	77.42	91.40	19.35	41.94	68.82	78.49

degrades for larger values of the scaling factor α and also for perturbations of more data components. Table 4 also indicates that the PLSA usually results in the largest savings in terms of time followed by the JWLSA and the NSA, whose performances are somewhat similar. The WLSA almost always has the largest percentage in the "much worse" row in comparison with the other three strategies.

A comparison of Table 4 and Table 1 reveals that the savings in the computation time in general are not as significant as the savings in the iteration count. For instance, while the WLSA is better or much better than cold start on about 99% of the instances in terms of iteration count for perturbations of b and c using $\alpha = .01$, the corresponding percentage reduces to about 42% in terms of the computation time. In order to understand this discrepancy, we report the cumulative timing comparison in Table 5, which presents the timing comparison in a similar manner to that of Table 2. For each strategy, we compute the ratio of the total computation time it requires to the total computation time using cold start. In order to accurately assess the contribution of the search time and the reoptimization time for each warm-start strategy, we also present the overall ratio in terms of the sum of the two ratios. "Search" refers to the ratio of the overall search time of the warm-start strategy to the total computation time using cold start. "Reopt" denotes the ratio of the overall reoptimization time of the warm-start strategy to the total computation time using cold start. Finally, "Total" indicates the sum of these two ratios, i.e., it corresponds to the ratio of the total computation time required by the warm-start strategy to that of cold start. For instance, the overall search time and the overall reoptimization of the PLSA are about 17% and 50% of the overall computation time of cold start, respectively, for perturbations of b with $\alpha = .01$, which implies that the total time taken by the PLSA is about 67% of the total time required by cold start, i.e., the PLSA reduces the overall computation time by about 33%.

Table 5 sheds some light into why the savings in computation time are not as significant as those in iteration count. For instance, Table 5 reveals that each of the WLSA, JWLSA, and NSA requires significant search times. In order to have a better understanding of this phenomenon, it is useful to consider these results in conjunction with those of Table 3, which presents the percentages of perturbed instances solved by cold start. In particular, for each warm-start strategy, the performance of the search stage degrades with larger perturbations and with perturbations of more data components. If a particular strategy reverts to cold start, this implies that each of the iterates of the original instance has been checked without successfully producing an acceptable warm-start. In terms of the computation time of the search stage, this brings significant overhead especially for the WLSA, JWLSA, and NSA, each of which spends considerable time for each iterate of the original instance. In contrast, the PLSA continues to maintain significantly smaller search times on such perturbed instances in comparison with the other three strategies since the computation of the adjustment is much cheaper.

Table 5 indicates that the overall search time in some cases considerably exceeds the total computation time of cold start. For instance, the overall search time of the WLSA is about 8.5 times larger than the total computation time of cold start for perturbations of b using $\alpha = 10$. A careful examination of our experimental results indicates that this unexpected behavior is largely due to a few large LP instances in

Table 5 Cumulative timing comparison of four warm-start strategies on four different types of perturbations

WS strategy	Time comp.	Perturbations of b				Perturbations of c			
		$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	Search	.17	.18	.21	.31	.07	.05	.06	.06
	Reopt	.50	.77	.82	.87	.75	.88	.92	1.04
	Total	.67	.95	1.03	1.18	.82	.93	.98	1.10
WLSA	Search	4.03	4.92	5.76	8.56	.70	.75	1.43	1.52
	Reopt	.35	.55	.86	.85	.38	.67	.68	1.05
	Total	4.38	5.47	6.62	9.41	1.08	1.42	2.11	2.57
JWLSA	Search	2.07	2.54	2.93	4.42	.36	.40	.69	.79
	Reopt	.36	.59	.78	.84	.46	.59	.84	1.05
	Total	2.43	3.13	3.71	5.26	.82	.99	1.53	1.84
NSA	Search	2.05	1.11	2.95	4.35	.37	.40	.72	.79
	Reopt	1.68	.83	2.23	.89	.35	.46	.84	1.00
	Total	3.73	1.94	5.18	5.24	.72	.86	1.56	1.79
WS strategy	Time comp.	Perturbations of b and c				Perturbations of A, b, and c			
		$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	Search	.18	.19	.19	.27	.16	.14	.14	.24
	Reopt	.53	.79	.85	.89	.95	.96	.85	.83
	Total	.71	.98	1.04	1.16	1.11	1.10	.99	1.07
WLSA	Search	4.35	4.54	5.15	7.43	3.03	2.54	3.01	5.50
	Reopt	.43	.56	.84	.82	.71	.72	.95	.89
	Total	4.78	5.10	5.99	8.25	3.74	3.26	3.96	6.39
JWLSA	Search	2.24	2.21	2.62	3.80	1.54	1.63	1.79	2.33
	Reopt	.39	.59	.83	.89	.72	.84	.84	.82
	Total	2.63	2.80	3.45	4.69	2.26	2.47	2.63	3.15
NSA	Search	2.00	2.24	2.62	3.77	1.23	1.31	1.49	2.79
	Reopt	1.44	1.83	2.05	.88	.61	.74	.88	.93
	Total	3.44	4.07	4.67	4.65	1.84	2.05	2.37	3.72

the NETLIB suite. It is well-known that most of the LP instances in this suite are "nasty" in the sense that small perturbations may considerably change the characteristics of the original instance. Therefore, on such perturbed instances, Algorithm 2.1 will almost always necessarily go through all the iterates of the original instance to identify a "near-feasible" starting point for the perturbed instance and will finally re-

vert to cold start. The problem becomes even more significant if the original instance is solved after a large number of interior-point iterations. This is the case especially on larger instances such as `dfl001` (6084/12243), `greenbeb` (227/4453), `pilot` (2443/5618), and `pilot87` (3586/7997), where $(\cdot/\cdot)$ denotes the number of rows and columns, respectively. For instance, PCx detects that the original instance `dfl001` is infeasible after 43 iterations, which takes about 1100 seconds. Our experiments indicate that the instance obtained by perturbing the right-hand side of `dfl001` with $\alpha = 10$ is infeasible and cold start, which forms a basis for time comparison of the warm-start strategies, detects infeasibility of this perturbed instance in only one iteration, which takes about 52 seconds. On this perturbed instance, each of the four warm-start strategies checks each of these 43 iterates before reverting to cold start. The WLSA spends as much as 2265 seconds on this instance in search of a warm start. The search time of each of the JWLSA and NSA on the same instance is about 1100 seconds whereas that of the PLSA is only about 50 seconds. As illustrated by this example, the search time of WLSA may be as much as twice the solution time of the original instance just to identify a starting iterate, at which point it will revert to cold start (cf. Algorithm 2.1). The factor two comes from the fact that the WLSA needs to compute two different factorizations for each iterate of the original instance. The search times of the JWLSA and NSA on such instances are roughly the same as the computation time of the original instance since they each require only one factorization per iterate of the original instance (cf. Sect. 2.3.1). On the other hand, the PLSA has a significant advantage in comparison with the other three warm-start strategies since it requires only one factorization for each original instance. A comparison of the overall "Search" ratios corresponding to different warm-start strategies presented in Table 5 justifies our observation.

We believe that these computational results exemplify potential drawbacks of the generic framework of Algorithm 2.1. The PLSA enjoys the advantage of fairly small search time in comparison with the other three strategies. The JWLSA and the NSA both have similar performances and offer much less savings in terms of the overall computation time. The WLSA exhibits the worst performance as the overall search time is roughly doubled in comparison with the JWLSA and the NSA.

Table 5 also reveals that the "Reopt" ratios for each of the four warm-start strategies are usually reasonably small, which indicates that our warm-start strategies in general succeed in reducing the reoptimization time over cold start. However, for the warm-start strategies requiring excessive search time, this reduction in the reoptimization time may be far outweighed and the advantage of using a warm-start strategy may quickly disappear.

4.4 Detecting infeasibility

In this subsection, we investigate whether warm-start strategies are effective in terms of detecting infeasibility of a perturbed instance in comparison with cold start.

Given an LP instance, PCx terminates with one of four possibilities: `optimal`, `infeasible`, `unknown`, and `suboptimal`. The `unknown` status indicates an uncorrelated convergence towards feasibility and optimality. If the iteration limit is reached, a status of `suboptimal` is returned. In Table 6, we report the outcomes

Table 6 Summary of the outcomes of perturbed instances

Status	Perturbations of b				Perturbations of c			
	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
Optimal	87.10	78.49	41.94	9.68	90.32	92.47	90.32	60.22
Infeasible	8.60	16.13	58.06	90.32	2.15	2.15	3.23	35.48
Unknown	4.30	5.38	0	0	7.53	5.38	6.45	4.30
Status	Perturbations of b and c				Perturbations of A, b, and c			
	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
Optimal	82.80	73.12	41.94	4.30	73.12	68.82	38.71	0
Infeasible	9.68	17.20	56.99	95.70	16.13	23.66	52.69	97.85
Unknown	7.53	9.68	1.08	0	10.75	7.53	8.60	2.15

Table 7 Cumulative iteration comparison of four warm-start strategies on infeasible perturbed instances

WS strategy	Perturbations of b				Perturbations of c			
	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	.56	.85	.97	1.01	.72	.98	.97	1.00
WLSA	.31	.68	.88	1.00	.38	.52	.73	1.00
JWLSA	.32	.69	.86	1.00	.38	.42	.64	1.00
NSA	.64	.91	.99	.99	.45	.76	.66	1.00
WS strategy	Perturbations of b and c				Perturbations of A, b, and c			
	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$	$\alpha = .01$	$\alpha = .1$	$\alpha = 1$	$\alpha = 10$
PLSA	.72	.88	.97	1.00	1.00	1.00	1.00	1.00
WLSA	.50	.75	.91	1.00	.79	.94	1.00	1.00
JWLSA	.53	.72	.92	1.00	.86	.93	1.00	1.00
NSA	.73	.99	1.01	1.00	.86	.94	1.00	1.00

of the perturbed instances in percentages for each of the four types of perturbations. Note that the percentage of infeasible instances usually increases with larger perturbations and with perturbations of more data components.

We use the iteration count in an attempt to assess whether warm-start strategies are effective in detecting infeasibility of a perturbed instance in comparison with cold start. As in Table 2, we report the ratio of the total number of iterations in the re-optimization stage for each warm-start strategy to the total number of iterations generated with cold start in Table 7. In contrast with Table 2, we only restrict ourselves to the perturbed instances terminating with a status of "infeasible" or "unknown." For

instance, this ratio is .56 for the PLSA for perturbations of b with $\alpha = .01$, which translates into a savings of 44% over cold start on such perturbed instances.

Table 7 reveals that warm-start strategies can lead to faster detection of infeasibility over cold start especially for smaller perturbations. For each warm-start strategy, the ratio increases with larger perturbations and with perturbations of more data components. The WLSA and JWLSA usually seem to offer the largest reductions in iteration count. The PLSA and NSA usually result in smaller savings.

4.5 Discussion

In this subsection, we discuss our computational results with an emphasis on the significance of the role played by each of the four desirable properties outlined in Sect. 2.3.

The first desirable property in Sect. 2.3 is related to the capability of a warm-start strategy to quickly identify a warm-start for a perturbed instance from among the iterates of the original instance. Table 3 provides statistics about this property for each of the four strategies. In particular, the WLSA and JWLSA are more capable of quickly identifying a warm-start. The NSA has a slightly worse performance. The PLSA seems to retreat to considerably earlier iterates in comparison with its counterparts.

The second desirable property pertains to the performance of a warm-start strategy in the reoptimization stage. Table 2 indicates that the WLSA and JWLSA exhibit the best performance in terms of the reduction of the number of iterations over cold start, followed closely by the NSA. The PLSA seems to offer the least advantage in this respect. Note that these observations are in close connection with those in the preceding paragraph. Therefore, the interplay between these two properties is clearly demonstrated by our computational results, i.e., sooner identification of a warm-start usually leads to faster reoptimization of the perturbed instance.

The third desirable property is related to the proximity of the computed warm-start to the central path of the perturbed instance. Note that none of our warm-start strategies enforces this property. Nevertheless, our computational results indicate that each strategy is still capable of reducing the computational work in the reoptimization stage. Therefore, we conclude that this property is relatively less important in comparison with the other desirable properties. We remark that the proximity to the central path is not enforced in any of the interior-point LP solvers. On the contrary, the solvers usually make very aggressive choices in computing step lengths towards the boundary. This is in consistence with our observations.

The last desirable property is concerned with the computational cost of the search stage of a warm-start strategy. The relevant statistics are provided in Table 5, which indicates that the PLSA has a significant advantage due to fairly small computation time per each iterate of the original instance. The JWLSA and NSA spend considerably larger times during the search stage since the cost of computing each adjustment roughly equals that of a single interior-point iteration. The search stage of the WLSA takes the largest amount of time since the computational cost of each adjustment is roughly twice that of an interior-point iteration.

It follows from our computational results that the first two desirable properties largely dictate the performance of a warm-start strategy in the reoptimization stage.

On the other hand, the last desirable property determines the efficiency of the search stage. In general, each of our warm-start strategies is usually capable of reducing the number of interior-point iterations and therefore the computation time during the reoptimization stage. These reductions are more pronounced for smaller perturbations and for perturbations of fewer number of data components. In terms of the iteration count, the WLSA and JWLSA have the best performances. The NSA leads to a slightly worse performance than these two strategies. The PLSA results in the smallest savings. On the other hand, the PLSA is a clear winner in terms of the overall time. The JWLSA and NSA both exhibit significantly worse performance than the PLSA. The WLSA offers the least advantage in terms of the overall computation time. Finally, it follows from the results of Table 7 that warm-start strategies can be effective in reducing the computational cost of the reoptimization stage even if the perturbed instance is infeasible. This is more significant especially for smaller perturbations.

We remark that the observations in the preceding paragraph are specific to the experimental setup used in this study. In particular, our implementation stores *all* the interior-point iterates generated during the course of the solution of the unperturbed instance. As indicated by our experimental results, this choice may cause severe problems especially for larger perturbations and for the adjustments requiring a significant amount of search time. We state several ways to alleviate this problem in Sect. 5. As before, we justify the choice of saving all the iterates by the desire to identify the most advanced iterate of the original instance that yields a successful warm-start for the perturbed instance. In addition, it allows us to determine the significance of the role played by the first desirable property.

Based on our computational results, we conclude that the PLSA has a reasonably good performance in the general scheme of Algorithm 2.1 if all the iterates of the original LP instance are stored and tested in search of a starting point for the perturbed instance. The JWLSA is a strong candidate to be the second followed closely by the NSA. The WLSA seems to lose its distinctive advantage in iteration count by the excessive overall search time.

5 Concluding remarks

We have implemented different warm-start strategies in interior-point methods for linear programming. We have included three members from the family of least-squares adjustments and the Newton step adjustment in our experiments.

Our extensive computational results on the LP instances in the NETLIB suite indicate that each of the warm-start strategies is generally effective in reducing the computational effort during the reoptimization stage. The performance of a warm-start strategy usually degrades with larger perturbations and with perturbations of more data components. This behavior is expected as the effectiveness of a warm-start strategy is largely dictated by the proximity between the original LP instance and the perturbed one. Among the warm-start strategies tested, we observe that the PLSA seems to have a distinctive overall advantage in terms of reducing the overall computation time in comparison with cold start despite the fact that it usually generates a successful warm-start at a considerably earlier iterate in comparison with the other three strategies.

Experimental results indicate that none of our warm-start strategies in general possesses all of the desirable properties outlined in Sect. 2.3 simultaneously. Especially, in light of the first, second, and fourth properties, we intend to investigate further warm-start strategies that can offer more distinctive advantages.

Our study also reveals several potential drawbacks of the generic warm-start algorithm given by Algorithm 2.1, which can be utilized in order to enhance the effectiveness of a warm-start strategy. In our experiments, we stored and tested each of the iterates of the original LP instance in searching for a starting iterate for the perturbed instance. The computational results indicate that this scheme can lead to excessive search times for some strategies especially with the increased distance between the original instance and the perturbed one. There are several potential remedies for this problem. For instance, instead of storing all iterates of the original instance, one can store only a subset. The selection of such a subset can be based on duality measure, which may be used to ensure that no two original iterates will have close duality measures. Furthermore, one can use binary search on the subset of stored iterates in order to decrease the search time. This approach assumes monotonicity in computing a successful warm-start, i.e., if an iterate of the original instance yields a successful warm-start, then so will the earlier iterates. This is justified by the theoretical results of [34] as long as the iterates are feasible and somewhat well-centered. Another remedy to reduce the search time is to impose an upper limit on the number of iterates that will be tested. If all such trials fail to produce an acceptable starting point, then one can revert to cold start. Such a scheme may be effective in preventing excessive search time.

As illustrated by our computational results, warm-start strategies offer little or no advantage for large perturbations. This suggests that one of the most important ingredients in the effectiveness of a warm-start strategy is the ability to correctly measure the relative distance between an original LP instance and a perturbed one. If this distance is above a certain threshold, then warm-start can be deemed to offer no advantage, in which case the perturbed instance can simply be solved using cold start. In such a scheme, the search stage can be completely bypassed. We intend to work on such distance measures that can be incorporated into our warm-start strategies in the near future.

Another interesting direction is to extend warm-start strategies to incorporate changes in the dimension of an LP problem. Such an extension would make warm-start strategies universally applicable. The branch-and-bound algorithm for integer programming is an ideal setting since both kinds of perturbations naturally arise in a branch-and-bound tree. We expect that a new implementation of this algorithm with warm-starts could potentially lead to significant savings in overall computation time.

We conclude this paper with a few remarks about the test problems used in this study. We used the well-known NETLIB test suite in our experiments. The fact that this collection consists of challenging LP instances was one of the deciding factors in our choice. We reasoned that the effectiveness of our warm-start strategies presumably will not be worse on LP problems that arise naturally in practice. Therefore, any positive result on the NETLIB suite would potentially translate into more significant savings in general. In fact, our experimental results on randomly generated transportation problems are generally in favor of this observation [18]. We have not included these results in this paper in order to maintain a reasonable length.

However, it is not entirely clear how one can define meaningful perturbations for the LP instances in the NETLIB suite. Therefore, we echo the request from researchers and practitioners in [2] for a meaningful data set for reoptimization.

Acknowledgements We are deeply grateful to Michael Wagner and Stephen J. Wright for their assistance with the PCx code in various stages of this study. We also thank the anonymous referees and the Associate Editor for their numerous helpful and perceptive suggestions, which considerably improved our presentation.

References

1. Andersen, E.D., Andersen, K.D.: Presolving in linear programming. Math. Program. **71**(2), 221–245 (1995)
2. Benson, H.Y., Shanno, D.F.: An exact primal-dual penalty method approach to warmstarting interior-point methods for linear programming. Comput. Optim. Appl. (2007, in press). doi:10.1007/s10589-007-9048-6
3. Benson, H.Y., Shanno, D.F.: Interior-point methods for nonconvex nonlinear programming: Regularization and warmstarts. Comput. Optim. Appl. (2007, in press). doi:10.1007/s10589-007-9089-x
4. Curtis, A.R., Reid, J.K.: On the automatic scaling of matrices for Gaussian elimination. J. Inst. Math. Appl. **10**, 118–124 (1972)
5. Czyzyk, J., Mehrotra, S., Wagner, M., Wright, S.J.: PCx: An interior-point code for linear programming. Optim. Methods Softw. **11–2**(1–4), 397–430 (1999)
6. Elhedhli, S., Goffin, J.L.: The integration of an interior-point cutting plane method within a branch-and-price algorithm. Math. Program. **100**(2), 267–294 (2004)
7. Fliege, J.: An efficient interior-point method for convex multicriteria optimization problems. Math. Oper. Res. **31**, 825–845 (2006)
8. Fliege, J., Heseler, A.: Constructing approximations to the efficient set of convex quadratic multiobjective problems. Technical report, Dortmund University, Dortmund, Germany (2002)
9. Forsgren, A.: On warmstarts for interior methods. In: Ceragioli, F., Dontchev, A., Furuta, H., Marti, K., Pandolfi, L. (eds.) System Modeling and Optimization. IFIP International Federation for Information Processing, vol. 199, pp. 51–66. Springer, Boston (2006)
10. Freund, R.M.: A potential function reduction algorithm for solving a linear program directly from an infeasible "warm start". Math. Program. **52**, 441–466 (1991)
11. Freund, R.M.: Theoretical efficiency of a shifted-barrier-function algorithm for linear programming. Linear Algebra Appl. **152**(1), 19–41 (1991)
12. Goffin, J.-L., Haurie, A., Vial, J.-P.: Decomposition and nondifferentiable optimization with the projective algorithm. Manag. Sci. **38**, 284–302 (1992)
13. Goffin, J.L., Vial, J.P.: Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method. Optim. Methods Softw. **17**(5), 805–867 (2002)
14. Gondzio, J.: Multiple centrality corrections in a primal-dual method for linear programming. Comput. Optim. Appl. **6**, 137–156 (1996)
15. Gondzio, J.: Warm start of the primal-dual method applied in the cutting-plane scheme. Math. Program. **83**, 125–143 (1998)
16. Gondzio, J., Grothey, A.: Re-optimization with the primal-dual interior point method. SIAM J. Optim. **13**(3), 842–864 (2003)
17. Gondzio, J., Vial, J.-Ph.: Warm start and epsilon-subgradients in the cutting plane scheme for block-angular linear programs. Comput. Optim. Appl. **14**(1), 17–36 (1999)
18. John, E.: Implementation of warm-start strategies in interior-point methods for linear programming. PhD thesis, Department of Applied Mathematics and Statistics, Stony Brook University (2005)
19. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica **4**, 373–395 (1984)
20. Lustig, I.J., Marsten, R.E., Shanno, D.F.: Interior point methods for linear programming: Computational state of the art. ORSA J. Comput. **6**, 1–14 (1994)
21. Mehrotra, S.: On the implementation of a primal-dual interior point method. SIAM J. Optim. **2**, 575–601 (1992)

22. Mitchell, J.E.: Computational experience with an interior-point cutting plane algorithm. SIAM J. Optim. **10**(4), 1212–1227 (2000)
23. Mitchell, J.E., Borchers, B.: Solving real-world linear ordering problems using a primal-dual interior point cutting plane method. Ann. Oper. Res. **62**, 253–276 (1996)
24. Mitchell, J.E., Todd, M.J.: Solving combinatorial optimization problems using Karmarkar's algorithm. Math. Program. **56**, 245–284 (1992)
25. Nesterov, Y.E., Nemirovskii, A.S.: Interior Point Polynomial Methods in Convex Programming. SIAM, Philadelphia (1994)
26. Ng, E., Peyton, B.W.: Block sparse Cholesky algorithms on advanced uniprocessor computers. SIAM J. Sci. Comput. **14**, 1034–1056 (1993)
27. Polyak, R.: Modified barrier functions (theory and methods). Math. Program. **54**, 177–222 (1992)
28. Renegar, J.: A Mathematical View of Interior-Point Methods in Convex Optimization. MPS/SIAM Series on Optimization, vol. 3. SIAM, Philadelphia (2001)
29. Todd, M.J.: Detecting infeasibility in infeasible-interior-point methods for optimization. In: Cucker, F., De Vore, R., Olver, P. (eds.) Foundations of Computational Mathematics, pp. 157–192. Cambridge University Press, Cambridge (2004)
30. Wright, S.J.: Primal-Dual Interior-Point Methods. SIAM, Philadelphia (1997)
31. Yıldırım, E.A.: An interior-point perspective on sensitivity analysis in semidefinite programming. Math. Oper. Res. **28**(4), 649–676 (2003)
32. Yıldırım, E.A., Todd, M.J.: Sensitivity analysis in linear programming and semidefinite programming using interior-point methods. Math. Program. **90**(2), 229–261 (2001)
33. Yıldırım, E.A., Todd, M.J.: An interior-point approach to sensitivity analysis in degenerate linear programs. SIAM J. Optim. **12**(3), 692–714 (2002)
34. Yıldırım, E.A., Wright, S.J.: Warm-start strategies in interior-point methods for linear programming. SIAM J. Optim. **12**(3), 782–810 (2002)