Mathematical Programming 66 (1994) 79–101

Bundle-based decomposition for large-scale convex optimization: Error estimate and application to block-angular linear programs

Deepankar Medhi *

Computer Science Telecommunications Program, University of Missouri-Kansas City, 5100 Rockhill Road, Kansas City, Missouri 64110, USA

Received 17 August 1988; revised manuscript received 12 October 1993

Abstract

Robinson has proposed the bundle-based decomposition algorithm to solve a class of structured large-scale convex optimization problems. In this method, the original problem is transformed (by dualization) to an unconstrained nonsmooth concave optimization problem which is in turn solved by using a modified bundle method. In this paper, we give a posteriori error estimates on the approximate primal optimal solution and on the duality gap. We describe implementation and present computational experience with a special case of this class of problems, namely, block-angular linear programming problems. We observe that the method is efficient in obtaining the approximate optimal solution and compares favorably with MINOS and an advanced implementation of the Dantzig–Wolfe decomposition method.

Keywords: Large-scale optimization; Nonsmooth optimization; Bundle method; Decomposition; Block-angular linear programming

1. Introduction

Robinson [23] has proposed the bundle-based decomposition method to solve the following convex optimization problem:

$$\inf_{x_1,\dots,x_N} \sum_{i=1}^{N} f_i(x_i) \tag{1.1a}$$

subject to

* E-mail: dmedhi@cstp.umkc.edu

$$\sum_{i=1}^{N} A_i x_i = a\,, \tag{1.1b}$$

where $a \in \mathbb{R}^m$, and for $i = 1, \ldots, N$, $A_i \in \mathbb{R}^{m \times n_i}$, $x_i \in \mathbb{R}^{n_i}$, and each f_i is a closed proper convex function taking values in the extended real line $(-\infty, \infty]$. Note that each f_i is allowed to take the value $+\infty$. Typically, the number of coupling constraints, m, is much smaller than the dimension of the whole problem $(n_1 + \cdots + n_N)$.

An example of (1.1) is the well-known block-angular linear programming problem [1]. To illustrate this point, set

$$f_i(x_i) = \begin{cases} \langle c_i, x_i \rangle & \text{if } B_i x_i = b_i,\ x_i \geqslant 0; \\ +\infty & \text{otherwise,} \end{cases} \tag{1.2}$$

$(c_i \in \mathbb{R}^{n_i}$, $b_i \in \mathbb{R}^{m_i}$, $B_i \in \mathbb{R}^{m_i \times n_i}$, for $i = 1, \ldots, N)$ and assume that the feasible regions $\{x_i \in \mathbb{R}^{n_i} \mid B_i x_i = b_i,\ x_i \geqslant 0\}$ are nonempty and bounded. Then, clearly, each f_i is a closed proper convex function and the problem (1.1) is equivalent to

$$\min_{x_1, \ldots, x_N} \ \langle c_1, x_1 \rangle + \cdots + \langle c_N, x_N \rangle$$

subject to

$$\begin{array}{ccccl} B_1 x_1 & & & & = b_1 \\ & B_2 x_2 & & & = b_2 \\ & & \ddots & & \ \vdots \\ & & & B_N x_N & = b_N \\ A_1 x_1 + & \cdots & + & A_N x_N & = a \end{array} \tag{1.3}$$

$$x_i \geqslant 0, \quad i = 1, \ldots, N\,.$$

The first step of the bundle-based decomposition (BBD) method involves dualization of problem (1.1) to obtain a nonsmooth concave problem, which may be solved in various ways. For example, subgradient methods (such as [22,5]) may be applied. A possible disadvantage of these subgradient methods, however, is the difficulty of obtaining a solution that satisfies primal optimality conditions within a certain tolerance; if these methods terminate at a point where the subgradient is nonzero, there is no simple way to find a primal-feasible solution (see the remark in Section 2.1). In the BBD method, on the other hand, the nonsmooth problem is solved using the bundle method [14], which allows an approximate primal-feasible solution to be calculated easily; see Theorem 2.5 in Section 2.2. We note also that Ha [4] suggested an alternative approach based on applying the proximal point algorithm [27] to create a sequence of smooth dual problems that may be solved by, for example, the BFGS method ([3, p. 119]).

In this paper, we give a posteriori error estimates on the approximate primal optimal solution (see Theorem 2.5 in §2.2) and on the duality gap. (For a priori convergence, see [24].) We also describe implementation and present computational experience for (1.3) which indicates the method to be promising. We observe from computational experience that, in practice, the approximate primal optimal solution obtained by this method produces a duality gap on the order of a user-specified tolerance level (ϵ of the optimality condition (1.4) introduced at the end of this section). Based on our comparison with both MINOS 5.0 [20] and DECOMP [7,11], an advanced implementation of the Dantzig–Wolfe Decomposition method, we observe this method to be competitive.

The rest of the paper is organized as follows. In Section 2, we describe the bundle-based decomposition method and give results on error estimates on both the primal optimal solution and the duality gap. The implementation of the method is described in Section 3 for the problem (1.3). In Section 4, we give computational experience.

We briefly describe our notations and definitions here. The number of elements in a set $\mathscr{S}$ is denoted by $\#(\mathscr{S})$. The scalar product of two vectors x and y in $\mathbb{R}^n$ is denoted by $\langle x, y\rangle$. The 2-norm $\langle x, x\rangle^{1/2}$ of a vector x in $\mathbb{R}^n$ is denoted by $\|x\|$. Most of the definitions related to convex functions can be found in [25]. For clarity, we present some of them here. The conjugate of a convex function f, denoted by f^*, is given by

$$f^*(x) := \sup_z \{\langle x, z\rangle - f(z)\} .$$

The subdifferential of a concave function $g(\cdot)$ at $y \in \mathbb{R}^m$ is defined as the set

$$\partial g(y) := \{\pi \mid \forall z,\ g(z) \leqslant g(y) + \langle \pi, z-y\rangle\} .$$

An element of the subdifferential set will be called a subgradient. The ϵ-subdifferential ($\epsilon \geqslant 0$) of a concave function $g(\cdot)$ at $y \in \mathbb{R}^m$ is defined as the set

$$\partial_\epsilon g(y) := \{\pi \mid \forall z,\ g(z) \leqslant g(y) + \langle \pi, z-y\rangle + \epsilon\} .$$

An element of $\partial_\epsilon g(y)$ is called an ϵ-subgradient. We say that $\bar{y}$ is ϵ-optimal for the problem of maximizing $g(\cdot)$ if and only if $0 \in \partial_\epsilon g(\bar{y})$. We, further, define the following property of an approximate maximizer $\bar{y}$ of a concave function $g(\cdot)$:

$$\bar{y} \text{ is an } \epsilon\delta\text{-optimal solution if and only if for any } \bar{d} \in \partial_\epsilon g(\bar{y}), \text{ it holds that } \|\bar{d}\| < \delta . \tag{1.4}$$

2. Bundle-based decomposition

In this section, we describe the bundle-based decomposition (BBD) method and give a posteriori error estimates on the approximate primal optimal solution and on the duality gap. We start with the method.

2.1. The method

The Rockafellar dual [26] of the problem (1.1), obtained by perturbing the coupling constraints (1.1b), is the unconstrained problem:

$$\max_y g(y) \tag{2.1a}$$

where

$$g(y) := \langle a, y\rangle - \sum_{i=1}^{N} f_i^*(A_i^{\mathrm{T}} y) \tag{2.1b}$$

and where

$$f_i^*(A_i^{\mathrm{T}} y) := \sup_{x_i} \{\langle A_i^{\mathrm{T}} y, x_i\rangle - f_i(x_i)\}, \quad i=1, \ldots, N. \tag{2.1c}$$

Note that $g(\cdot)$ is a nonsmooth concave function. Under the constraint qualification that

$$a \in \sum_{i=1}^{N} A_i(\mathrm{ri}\, \mathrm{dom} f_i)\ ,$$

the infimum in problem (1.1) is equal to the maximum in (2.1). Here, ri stands for relative interior (see [25]), and

$$A_i(\mathrm{ri}\, \mathrm{dom} f_i) := \{A_i z \mid z \in \mathrm{ri}\, \mathrm{dom} f_i\}\ .$$

Furthermore, if the condition

$$\bigcap_{i=1}^{N} (\mathrm{im} A_i^{\mathrm{T}} \cap \mathrm{ri}\, \mathrm{dom} f_i^*) \neq \emptyset$$

(see [25]) is satisfied, then one has

$$\partial g(y) = a - \sum_{i=1}^{N} A_i \partial f_i^*(A_i^{\mathrm{T}} y) \tag{2.2a}$$

where

$$\partial f_i^*(A_i^{\mathrm{T}} y) = \operatorname*{argmin}_{x_i} \{f_i(x_i) - \langle A_i^{\mathrm{T}} y, x_i\rangle\}, \quad i=1, \ldots, N. \tag{2.2b}$$

Thus, for a given y, if one solves the N smaller subproblems (2.1c) to obtain solutions $x_i(y)$, i.e., $x_i(y) \in \partial f_i^*(A_i^{\mathrm{T}} y)$, for $i=1, \ldots, N$ then a subgradient $\pi(y) = a - \sum_{i=1}^{N} A_i x_i(y) \in \partial g(y)$ can be computed. Also, from the optimal objective function values of the subproblems, the dual objective $g(y)$ can be computed using (2.1b).

Remark. One may not obtain a primal optimal solution using a Polyak-type subgradient algorithm [22] to solve the nonsmooth dual problem (2.1). To see this, suppose we know a dual optimal solution $\bar{y}$. Then for this $\bar{y}$, the corresponding subgradient $\pi(\bar{y})$ may not be 0. Thus, corresponding solution $x_i(\bar{y})$ of the subproblems (2.1c) produce

$$a - \sum_{i=1}^{N} A_i x_i(\bar{y}) \neq 0\ . \tag{2.3}$$

Then, $x = (x_1(\bar{y}), \ldots, x_N(\bar{y}))$ would not be a feasible solution to the problem (1.1).

The BBD method solves the dual problem (2.1) by a modified bundle method. Instead of using a subgradient for a search direction, as in Polyak-type algorithms, this method keeps and uses a bundle of subgradients (and a corresponding primal bundle) to obtain an ascent direction, and to obtain an approximate primal optimal solution readily in the end. We shall use the following notation in the algorithm and subsequent results:

$$\alpha(y^k, y^j, \pi^j) := g(y^j) - g(y^k) + \langle \pi^j, y^k - y^j \rangle .$$

We present the BBD algorithm below:

Step 0. *Initialization.*
Select initialization parameters, ϵ, $\delta > 0$, an initial $\hat{\epsilon} \geqslant \epsilon$, and a number b for the maximum size of the bundle. Choose β, γ and μ such that

$$0 < \gamma < \beta < 1, \qquad \beta + \mu < 1, \qquad \mu > 0 .$$

Pick a starting dual point y^1. Set $k \leftarrow 1$, $\epsilon^1 \leftarrow \hat{\epsilon}$.

Step 1. *Compute the dual function and the subgradient for the starting point.*
For given y^1, solve the N subproblems (2.1c). Let x_i^1 be the computed optimal solution (i.e. $x_i^1 \in \partial f_i^*(A_i^{\mathrm{T}} y^1)$) and z_i^1 be the optimal objective function value for the ith subproblem (2.1c). Compute the dual function $g(y^1)$ and a subgradient $\pi^1 \in \partial g(y^1)$. Set

$$\mathscr{D} \leftarrow \{\pi^1\} ,$$

$$\mathscr{P} \leftarrow \{(x_1^1, \ldots, x_N^1)\} ,$$

$$\mathscr{K} \leftarrow \{1\} ,$$

$$\alpha_{11} \leftarrow 0 .$$

Step 2. *Compute a search direction.*
Update

$$\epsilon_k \leftarrow \max\{\epsilon, \min\{\epsilon_k, \hat{\epsilon}\}\} .$$

Solve the following quadratic programming (QP) problem

$$\min_{\lambda_j, j \in \mathscr{K}} \frac{1}{2} \| \sum_{j \in \mathscr{K}} \lambda_j \pi^j \|^2$$

subject to

$$\sum_{j \in \mathscr{K}} \lambda_j = 1 ,$$

$$\sum_{j \in \mathscr{K}} \alpha_{jk} \lambda_j \leqslant \epsilon_k ,$$

$$\lambda_j \geqslant 0, \quad j \in \mathscr{K} \tag{2.4}$$

to obtain the solution $\{\lambda_j^k, j \in \mathscr{K}\}$, the direction

$$d^k = \sum_{j \in \mathscr{K}} \lambda_j^k \pi^j \tag{2.5a}$$

and

$$v_k := \|d_k\|^2 + s_k \epsilon_k \tag{2.5b}$$

where s_k is the dual multiplier associated with the constraint $\sum_{j\in\mathscr{K}} \alpha_{jk}\lambda_j \leqslant \epsilon_k$.

Step 3. *Test for convergence.*
If $\|d^k\| < \delta$, and $\sum_{j\in\mathscr{K}} \alpha_{jk}\lambda_j^k \leqslant \epsilon$, then we are done. Compute an approximate primal optimal solution as follows:

$$\bar{x}_i = \sum_{j\in\mathscr{K}} \lambda_j^k x_i^j, \quad i = 1, \ldots, N, \tag{2.6}$$

and then compute the final objective function value. STOP. Else, if $\|d^k\| < \delta$ but $\sum_{j\in\mathscr{K}} \alpha_{jk}\lambda_j^k > \epsilon$, then $\hat{\epsilon} \leftarrow \mu\hat{\epsilon}$, and go to step 2. Else, go to step 4.

Step 4. *Reduction of bundle.*
If $\#(\mathscr{K}) = b$, then

$$\mathscr{K} \leftarrow \{j \in \mathscr{K} \mid \lambda_j^k > 0\},$$

$$\mathscr{D} \leftarrow \{\pi^j \mid j \in \mathscr{K}\},$$

$$\mathscr{P} \leftarrow \{(x_1^j, \ldots, x_N^j) \mid j \in \mathscr{K}\}.$$

If $\#(\mathscr{K}) = b$, then (still too large, reduce to a singleton set)

$$\pi^k \leftarrow d^k,$$

$$\mathscr{D} \leftarrow \{\pi^k\},$$

$$\mathscr{P} \leftarrow \{(x_1^k, \ldots, x_N^k) = \sum_{j\in\mathscr{K}} \lambda_j^k (x_1^j, \ldots, x_N^j)\},$$

$$\alpha_{kk} \leftarrow \sum_{j\in\mathscr{K}} \alpha_{jk}\lambda_j^k,$$

$$\mathscr{K} \leftarrow \{k\}.$$

Step 5. *Perform line search.*
Using the line search techniques of Lemaréchal [13], find $t_k > 0$ to compute a trial point $u^{k+1} = y^k + t_k d^k$, then solve the subproblems (2.1c) [i.e. obtain $x_i^{k+1} \in \partial f_i^*(A_i^{\mathrm{T}} u^{k+1})$] to obtain the dual objective $g(u^{k+1})$ and a subgradient π^{k+1} at u^{k+1} so that

$$\langle \pi^{k+1}, d^k \rangle \leqslant \beta v_k$$

and, either

$$g(u^{k+1}) \geqslant g(y^k) + \gamma t_k v_k, \qquad \text{(serious step)}$$

or

$$\alpha(y^k, u^{k+1}, \pi^{k+1}) \leqslant \mu\epsilon_k. \qquad \text{(null step)}$$

(Here, v_k is as defined in (2.5b)).
If it is a serious step, update

$$y^{k+1} \leftarrow u^{k+1},$$
$$\alpha_{j,k+1} \leftarrow \alpha_{jk} + g(y^k) - g(y^{k+1}) + \langle \pi^j, y^{k+1} - y^k \rangle, \quad j \in \mathscr{K},$$
$$\alpha_{k+1,k+1} \leftarrow 0,$$
$$\mathscr{K} \leftarrow \mathscr{K} \cup \{k+1\},$$
$$\mathscr{D} \leftarrow \mathscr{D} \cup \{\pi^{k+1}\},$$
$$\mathscr{P} \leftarrow \mathscr{P} \cup \{(x_1^{k+1}, \ldots, x_N^{k+1})\}.$$

If it is a null step, update

$$y^{k+1} \leftarrow y^k,$$
$$\alpha_{j,k+1} \leftarrow \alpha_{jk}, \quad j \in \mathscr{K},$$
$$\alpha_{k+1,k+1} \leftarrow \alpha(y^{k+1}, u^{k+1}, \pi^{k+1}),$$
$$\mathscr{K} \leftarrow \mathscr{K} \cup \{k+1\},$$
$$\mathscr{D} \leftarrow \mathscr{D} \cup \{\pi^{k+1}\},$$
$$\mathscr{P} \leftarrow \mathscr{P} \cup \{(x_1^{k+1}, \ldots, x_N^{k+1})\}.$$

Set k ← k + 1, and go to step 2.

Comments. (1) The feasible region of the QP (2.4) is an inner approximation of the set $\partial_{\epsilon_k} g(y^k)$.

(2) The line search method is due to Lemaréchal [13]. He has shown that the line search finds either a serious or a null step in a finite number of computation of the function and the subgradient.

(3) It is easy to show that the weights satisfy (for $j \in \mathscr{K}$)

$$\begin{aligned}\alpha_{jk} &\leftarrow \alpha_{j,k-1} + g(y^{k-1}) - g(y^k) + \langle \pi^j, y^k - y^{k-1} \rangle \\ &= \alpha_{jj} + g(y^j) - g(y^k) + \langle \pi^j, y^k - y^j \rangle.\end{aligned}$$

The advantage of the first expression is that it is recursive.

(4) At each iteration, the dual bundle $\mathscr{D}$ and the primal bundle $\mathscr{P}$ are preserved; while the dual bundle is used for computing the direction given in (2.5a), the primal bundle is used for computing the approximate primal optimal solution (2.6). The size of the bundle is not allowed to exceed b; thus, finite storage is required for the primal and the dual bundle.

2.2. *Error estimate*

In this section, we will show that by using this method, we can obtain an approximate primal optimal solution and an a posteriori error estimate on this solution. We also obtain an estimate of the duality gap for this solution. We need the following lemmas to obtain the results.

Lemma 2.1. *For $i=1, \ldots, N$ and $j \in \mathscr{K}$,*

$$x_i^j \in \partial_{\alpha_{jki}} f_i^*(A_i^{\mathrm{T}} y^k) ,$$

where

$$\alpha_{jki} := \alpha_{jji} + f_i^*(A_i^{\mathrm{T}} y^k) - f_i^*(A_i^{\mathrm{T}} y^j) - \langle A_i^{\mathrm{T}} y^k - A_i^{\mathrm{T}} y^j, x_i^j \rangle$$

and where

$$\alpha_{jji} = \begin{cases} 0 & \text{for serious step at } j, \\ -f_i^*(A_i^{\mathrm{T}} u^j) + f_i^*(A_i^{\mathrm{T}} y^j) - \langle A_i^{\mathrm{T}} y^j - A_i^{\mathrm{T}} u^j, x_i^j \rangle & \text{for null step at } j, \\ \sum_{\ell \in \mathscr{J}} \alpha_{\ell ji} \lambda_\ell^j & \text{if bundle reduces} \\ & \text{to a singleton element} \end{cases}$$

($\mathscr{J}$ is the appropriate set of iteration indices up to iteration j).

Proof. It suffices to show the result for a particular i and j.

Case I: for a serious step at j.

By definition,

$$x_i^j \in \partial f_i^*(A_i^{\mathrm{T}} y^j) .$$

This implies that for each $z_i \in \mathbb{R}^{n_i}$,

$$\begin{aligned} f_i^*(z_i) &\geq f_i^*(A_i^{\mathrm{T}} y^j) + \langle z_i - A_i^{\mathrm{T}} y^j, x_i^j \rangle \\ &= f_i^*(A_i^{\mathrm{T}} y^k) + \langle z_i - A_i^{\mathrm{T}} y^k, x_i^j \rangle - \{ f_i^*(A_i^{\mathrm{T}} y^k) \\ &\quad - f_i^*(A_i^{\mathrm{T}} y^j) - \langle A_i^{\mathrm{T}} y^k - A_i^{\mathrm{T}} y^j, x_i^j \rangle \} \\ &= f_i^*(A_i^{\mathrm{T}} y^k) + \langle z_i - A_i^{\mathrm{T}} y^k, x_i^j \rangle - \alpha_{jki} , \end{aligned}$$

which, in turn, implies the result.

Case II, for a null step at j.

By definition,

$$x_i^j \in \partial f_i^*(A_i^{\mathrm{T}} u^j) .$$

This implies that for each $z_i \in \mathbb{R}^{n_i}$,

$$\begin{aligned} f_i^*(z_i) &\geq f_i^*(A_i^{\mathrm{T}} u^j) + \langle z_i - A_i^{\mathrm{T}} u^j, x_i^j \rangle \\ &= f_i^*(A_i^{\mathrm{T}} y^j) + \langle z_i - A_i^{\mathrm{T}} y^j, x_i^j \rangle \\ &\quad - \{ (- f_i^*(A_i^{\mathrm{T}} u^j) + f_i^*(A_i^{\mathrm{T}} y^j) - \langle A_i^{\mathrm{T}} y^j - A_i^{\mathrm{T}} u^j, x_i^j \rangle \} \\ &= f_i^*(A_i^{\mathrm{T}} y^j) + \langle z_i - A_i^{\mathrm{T}} y^j, x_i^j \rangle - \alpha_{jji} . \end{aligned}$$

That is,

$$x_i^j \in \partial_{\alpha_{jji}} f_i^*(A_i^{\mathrm{T}} y^j) .$$

Continuing,

$$f_i^*(z_i) = f_i^*(A_i^T y^k) + \langle z_i - A_i^T y^k, x_i^j\rangle$$
$$- \{\alpha_{jji} + (f_i^*(A_i^T y^k) - f_i^*(A_i^T y^j) - \langle A_i^T y^k - A_i^T y^j, x_i^j\rangle)\}$$
$$= f_i^*(A_i^T y^k) + \langle z_i - A_i^T y^k, x_i^j\rangle - \alpha_{jki},$$

which, in turn, implies the result.

Case III: When each of the bundles, $\mathscr{D}$ and $\mathscr{P}$, is reduced to a singleton element at j.

By cases I and II above, we have (when iteration ℓ is serious or null),

$$x_i^\ell \in \partial_{\alpha_{\ell ji}} f_i^*(A_i^T y^j), \quad \ell \in \mathscr{J}.$$

This implies

$$f_i^*(z_i) \geqslant f_i^*(A_i^T y^j) + \langle z_i - A_i^T y^i, x_i^\ell\rangle - \alpha_{\ell ji}, \quad \ell \in \mathscr{J}.$$

Multiplying by the solution $\{\lambda_\ell^j, \ell \in \mathscr{J}\}$ of the QP (2.4) at iteration j, and summing over $\ell \in \mathscr{J}$, we obtain

$$f_i^*(z_i) \geqslant f_i^*(A_i^T y^j) + \langle z_i - A_i^T y^j, \sum_{\ell \in \mathscr{J}} \lambda_\ell^j x_i^\ell\rangle - \sum_{\ell \in \mathscr{J}} \alpha_{\ell ji} \lambda_\ell^j.$$

By definition, this implies

$$f_i^*(z_i) \geqslant f_i^*(A_i^T y^j) + \langle z_i - A_i^T y^j, x_i^j\rangle - \alpha_{jji}$$

which means

$$x_i^j \in \partial_{\alpha_{jji}} f_i^*(A_i^T y^j).$$

Using a similar argument as in case II above, we get

$$x_i^j \in \partial_{\alpha_{jki}} f_i^*(A_i^T y^k). \qquad \square$$

Following arguments similar to those in Lemma 2.1, we can show the following lemma (proof not given).

Lemma 2.2. *For* $j \in \mathscr{K}$,

$$\pi^j \in \partial_{\alpha_{jk}} g(y^k),$$

where

$$\alpha_{jk} = \alpha_{jj} + g(y^j) - g(y^k) + \langle \pi^j, y^k - y^j\rangle$$

and where

$$\alpha_{jj} = \begin{cases} 0 & \text{for serious step at } j, \\ g(u^j) - g(y^k) + \langle \pi^j, y^k - u^j\rangle & \text{for null step at } j, \\ \sum_{\ell \in \mathscr{J}} \alpha_{\ell j} \lambda_\ell^j & \text{if bundle reduces to a singleton element} \end{cases}$$

($\mathcal{J}$ is the appropriate set of iteration indices up to iteration j).

Lemma 2.3. *Let α_{jki} and α_{jk} be as defined in the previous lemmas. Then*

$$\sum_{i=1}^{N} \alpha_{jki} = \alpha_{jk} .$$

Proof. We will show the result for the case when iteration j is a serious step or a null step or when the bundle is reduced to a singleton set.

Case I: for a serious step at j, $\alpha_{jj}=0$, and thus

$$\begin{aligned}
\alpha_{jk} &= g(y^j) - g(y^k) + \langle y^k - y^j, \pi^j\rangle \\
&= \langle y^j, a\rangle - \sum_{i=1}^{N} f_i^*(A_i^{\mathrm{T}} y^j) - \langle y^k, a\rangle + \sum_{i=1}^{N} f_i^*(A_i^{\mathrm{T}} y^k) \\
&\quad + \langle y^k - y^j, a - \sum_{i=1}^{N} A_i x_i^j\rangle \qquad \text{(using the definitions of } g \text{ and } \pi^j) \\
&= \sum_{i=1}^{N} \{f_i^*(A_i^{\mathrm{T}} y^k) - f_i^*(A_i^{\mathrm{T}} y^j) - \langle A_i^{\mathrm{T}} y^k - A_i^{\mathrm{T}} y^j, x_i^j\rangle\} \\
&= \sum_{i=1}^{N} \alpha_{jki} .
\end{aligned}$$

Case II: for a null step at j,

$$\alpha_{jj} = \alpha(y^j, u^j, \pi^j) ,$$

and thus

$$\begin{aligned}
\alpha_{jk} &= \alpha_{jj} + g(y^j) - g(y^k) + \langle y^k - y^j, \pi^j\rangle \\
&= g(u^j) - g(y^k) + \langle y^k - u^j, \pi^j\rangle \\
&= \langle u^j, a\rangle - \sum_{i=1}^{N} f_i^*(A_i^{\mathrm{T}} u^j) - \langle y^k, a\rangle + \sum_{i=1}^{N} f_i^*(A_i^{\mathrm{T}} y^k) \\
&\quad + \left\langle y^k - u^j, a - \sum_{i=1}^{N} A_i x_i^j \right\rangle \\
&= \sum_{i=1}^{N} \{f_i^*(A_i^{\mathrm{T}} y^k) - f_i^*(A_i^{\mathrm{T}} y^j) \\
&\quad - \langle A_i^{\mathrm{T}} y^k - A_i^{\mathrm{T}} u^j, x_i^j\rangle\} \\
&= \sum_{i=1}^{N} a_{jki} .
\end{aligned}$$

Case III: If the bundle is reduced to a singleton set,

$$\alpha_{jj} = \sum_{\ell \in \mathscr{J}} \alpha_{\ell j} \lambda_\ell^j = \sum_{\ell \in \mathscr{J}} \sum_{i=1}^N \alpha_{\ell j i} \lambda_\ell^j = \sum_{i=1}^N \sum_{\ell \in \mathscr{J}} \alpha_{\ell j i} \lambda_\ell^j = \sum_{i=1}^N \alpha_{jji}\,.$$

Now,

$$\begin{aligned}\alpha_{jk} &= \alpha_{jj} + g(y^j) - g(y^k) + \langle y^k - y^j, \pi^j \rangle \\ &= \sum_{i=1}^N \alpha_{jji} + \sum_{i=1}^N \{f_i^*(A_i^{\mathrm{T}} y^k) - f_i^*(A_i^{\mathrm{T}} y^j) - \langle A_i^{\mathrm{T}} y^k - A_i^{\mathrm{T}} y^j, x_i^j \rangle\} \\ &= \sum_{i=1}^N \{\alpha_{jji} + f_i^*(A_i^{\mathrm{T}} y^k) - f_i^*(A_i^{\mathrm{T}} y^j) - \langle A_i^{\mathrm{T}} y^k - A_i^{\mathrm{T}} y^j, x_i^j \rangle\} \\ &= \sum_{i=1}^N \alpha_{jki}\,. \qquad \square\end{aligned}$$

First we present the result for dual optimality (due to [14]).

Theorem 2.4. *If at iteration k, the exact solution $\{\lambda_j^k, j \in \mathscr{K}\}$ of the* QP (2.4) *satisfies*

$$\sum_{j \in \mathscr{K}} \alpha_{jk} \lambda_j^k \leqslant \epsilon$$

and $\|d^k\| < \delta$, then y^k is $\epsilon\delta$-optimal as defined in (1.4).

Proof. From Lemma 2.2., we have

$$\pi^j \in \partial_{\alpha_{jk}} g(y^k), \quad j \in \mathscr{K}\,,$$

which implies that for each $y \in \mathbb{R}^m$,

$$g(y) \leqslant g(y^k) + \langle \pi^j, y - y^k \rangle + \alpha_{jk}, \quad j \in \mathscr{K}\,.$$

Multiplying by the solution λ_j^k of the QP (2.4), and summing over $j \in \mathscr{K}$, we get

$$\begin{aligned}g(y) &\leqslant g(y^k) + \langle \sum_{j \in \mathscr{K}} \lambda_j^k \pi^j, y - y^k \rangle + \sum_{j \in \mathscr{K}} \alpha_{jk} \lambda_j^k \\ &\leqslant g(y^k) + \langle d^k, y - y^k \rangle + \epsilon\,.\end{aligned}$$

This, by definition, implies that y^k is $\epsilon\delta$-optimal. $\square$

Now we are ready to present the main results. We assume that a dual iterate y^k has been obtained which satisfies the optimality condition as states in (1.4) and we write this approximate dual optimal solution y^k as $\bar{y}$.

Theorem 2.5. *Let α_{jki} and α_{jk} be as defined before. Let $\{\bar{\lambda}_j, j \in \mathscr{K}\}$ be the exact solution of the quadratic programming problem* (2.4), *and*

$$\sum_{j \in \mathscr{K}} \alpha_{jk} \bar{\lambda}_j \leqslant \epsilon_k \leqslant \epsilon .$$

Then $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_N)$, given by

$$\bar{x}_i = \sum_{j \in \mathscr{K}} \bar{\lambda}_j x_i^j, \quad i = 1, \ldots, N, \tag{2.7}$$

is an approximate primal optimal solution so that for every $(x_1, \ldots, x_N)$ with $\sum_{i=1}^N A_i x_i = a$,

$$\sum_{i=1}^N f_i(x_i) \geqslant \sum_{i=1}^N f_i(\bar{x}_i) + \langle \bar{d}, \bar{y} \rangle - \epsilon , \tag{2.8}$$

where $\bar{d} = a - \sum_{i=1}^N A_i \bar{x}_i = \sum_{j \in \mathscr{K}} \bar{\lambda}_j \pi^j$ with $\|\bar{d}\| < \delta$.

Proof. By Lemma 2.1, we have

$$x_i^j \in \partial_{\alpha_{jki}} f_i^*(A_i^{\mathrm{T}} \bar{y})$$

which says that for each $z_i \in \mathbb{R}^{n_i}$,

$$f_i^*(z_i) \geqslant f_i^*(A_i^{\mathrm{T}} \bar{y}) + \langle z_i - A_i^{\mathrm{T}} \bar{y}, x_i^j \rangle - \alpha_{jki} .$$

Multiplying by $\bar{\lambda}_j$, summing over $j \in \mathscr{K}$, and using (2.7), we obtain for each $z_i \in \mathbb{R}^{n_i}$,

$$f_i^*(z_i) \geqslant f_i^*(A_i^{\mathrm{T}} \bar{y}) + \langle z_i - A_i^{\mathrm{T}} \bar{y}, \bar{x}_i \rangle - \sum_{j \in \mathscr{K}} \bar{\lambda}_j \alpha_{jki} .$$

Setting $\nu_i = \sum_{j \in \mathscr{K}} \bar{\lambda}_j \alpha_{jki}$, we can rewrite this relation as

$$\bar{x}_i \in \partial_{\nu_i} f_i^*(A_i^{\mathrm{T}} \bar{y}) ,$$

which is equivalent to

$$A_i^{\mathrm{T}} \bar{y} \in \partial_{\nu_i} f_i(\bar{x}_i)$$

(see [6]). By definition, this means that for each x_i,

$$f_i(x_i) \geqslant f_i(\bar{x}_i) + \langle x_i - \bar{x}_i, A_i^{\mathrm{T}} \bar{y} \rangle - \nu_i .$$

Summing over $i = 1, \ldots, N$, we obtain that for each $x = (x_1, \ldots, x_n)$,

$$\sum_{i=1}^N f_i(x_i) \geqslant \sum_{i=1}^N f_i(\bar{x}_i) + \langle \sum_{i=1}^N A_i(x_i - \bar{x}_i), \bar{y} \rangle - \epsilon ,$$

since (using Lemma 2.3),

$$\sum_{i=1}^N \nu_i = \sum_{i=1}^N \sum_{j \in \mathscr{K}} \bar{\lambda}_j \alpha_{jki} = \sum_{j \in \mathscr{K}} \bar{\lambda}_j \sum_{i=1}^N \alpha_{jki} = \sum_{j \in \mathscr{K}} \bar{\lambda}_j \alpha_{jk} \leqslant \epsilon_k \leqslant \epsilon .$$

To obtain the final result consider all $x = (x_1, \ldots, x_N)$ such that $\sum_{i=1}^N A_i x_i = a$. Then we have

$$\sum_{i=1}^{N} f_i(x_i) \geqslant \sum_{i=1}^{N} f_i(\bar{x}_i) + \langle a - a + \bar{d}, \bar{y}\rangle - \epsilon$$
$$= \sum_{i=1}^{N} f_i(\bar{x}_i) + \langle \bar{d}, \bar{y}\rangle - \epsilon. \qquad \square$$

Note that the error bound given by (2.8) in Theorem 2.5 is computable a posteriori, since $\bar{d}$, $\bar{y}$, and ϵ will all be known. We refer to the approximate primal optimal solution given by (2.7) satisfying (2.8) as the $\epsilon\delta$-optimal primal solution. A less stringent inequality than (2.8), given directly in terms of δ, $\bar{y}$ and ϵ, is

$$\sum_{i=1}^{N} f_i(x_i) > \sum_{i=1}^{N} f_i(\bar{x}_i) - (\delta\|\bar{y}\| + \epsilon) .$$

Finally, we give the following result on the duality gap.

Theorem 2.6. *Let $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_N)$ and $\bar{y}$ be a pair of $\epsilon\delta$-optimal solutions to the primal and the dual satisfying the relation*

$$a - \sum_{i=1}^{N} A_i\bar{x}_i = \bar{d} \in \partial_{\epsilon} g(\bar{y}) \quad \text{with} \quad \|\bar{d}\| < \delta \text{ and } \sum_{j\in\mathscr{K}} \alpha_{jk}\bar{\lambda}_j \leqslant \epsilon .$$

Then,

$$\sum_{i=1}^{N} f_i(\bar{x}_i) - g(\bar{y}) \leqslant \epsilon - \langle \bar{y}, \bar{d}\rangle < \epsilon + \delta\|\bar{y}\| .$$

Proof. While proving Theorem 2.5, we obtained the result

$$A_i^{\mathrm{T}}\bar{y} \in \partial_{\nu_i} f_i(\bar{x}_i) .$$

This implies that

$$\nu_i \geqslant f_i(\bar{x}_i) + f_i^*(A_i^{\mathrm{T}}\bar{y}) - \langle A_i^{\mathrm{T}}\bar{y}, \bar{x}_i\rangle .$$

Summing over, $i = 1, \ldots, N$, we get

$$\sum_{i=1}^{N} \nu_i \geqslant \sum_{i=1}^{N} f_i(\bar{x}_i) + \sum_{i=1}^{N} f_i^*(A_i^{\mathrm{T}}\bar{y}) - \langle \sum_{i=1}^{N} A_i\bar{x}_i, \bar{y}\rangle .$$

Using the fact that $\sum_{i=1}^{N} \nu_i \leqslant \epsilon$ and that $\bar{d} = \mathrm{a} - \sum_{i=1}^{N} A_i\bar{x}_i$, we get

$$\epsilon \geqslant \sum_{i=1}^{N} f_i(\bar{x}_i) - \left[\langle a, \bar{y}\rangle - \sum_{i=1}^{N} f_i^*(A_i^{\mathrm{T}}\bar{y})\right] + \langle \bar{d}, \bar{y}\rangle .$$

This implies that

$$\sum_{i=1}^{N} f_i(\bar{x}_i) - g(\bar{y}) \leqslant \epsilon - \langle \bar{y}, \bar{d} \rangle < \epsilon + \delta \|\bar{y}\| . \qquad \square$$

Thus, we obtain an a posteriori bound on the duality gap.

3. Implementation of the algorithm

We have implemented the bundle-based decomposition algorithm in Fortran 77. Our implementation, the code BUNDECOMP, was specifically designed to solve the block-angular problem (1.3); BUNDECOMP was developed by modifying M1FC1, a code for unconstrained nonsmooth convex minimization due to Lemaréchal [15]. The line search in M1FC1 is due to Lemaréchal [13], and the quadratic programming problem (2.4) is solved by an efficient method due to Mifflin [19]. The objective function and the subgradient are provided through the subroutine SIMUL.

For block-angular linear programming problems (1.3), the subproblems (2.1c) take the special form

$$\max_{x_i} \{ \langle yA_i - c_i, x_i \rangle \,|\, B_i x_i = b_i,\ x_i \geqslant 0 \} . \tag{3.1}$$

These subproblems, which are smaller linear programs, must be solved to compute the dual objective function and subgradient. We solved the subproblems using the revised simplex routine ZX0LP from IMSL library [12], in which the needed matrices are stored in dense form. Note that from one value of y to another, only the objective function of (3.1) changes; the feasible region remains the same. This means that for the very first iterate y^1, problem (3.1) is required to be solved from scratch (cold start). After that to move from one dual iterate y^k to next iterate y^{k+1}, the procedure to solve (3.1) can start from the optimal basis obtained at y^k (warm start) and find a new optimal basis for the changed objective function; this can be accomplished using parametric linear programming [21, chapter 8]. Our parametric programming code to solve (3.1) at subsequent y's is also based on ZX0LP. From our initial computational experience, we have observed that by using parametric programming, the time to solve (3.1) at subsequent y's is reduced by a factor of about seven to ten compared to the case where (3.1) was solved from scratch at every dual iteration. Thus, in BUNDECOMP, we use a cold start version of ZX0LP for the very first iterate and the warm start parametric programming code for subsequent iterates to solve (3.1) (these two cases are provided within SIMUL).

The user is required to provide the following main parameters for using BUNDECOMP (as needed for M1FC1):

DX: Accuracy for successive y's,

DF1: A positive number which is used for expected change in the objective function at the first iteration. It is also used for initializing the step-size and the ϵ of the optimality condition (1.4),

EPS: the ϵ of the optimality condition (1.4) (accuracy on the objective function).

M1FC1 uses the following rule for the value of δ appearing in the optimality condition (1.4):

$$\delta = 10^{-4}(\text{EPS})^2\|\pi^1\|^2/(\text{DF1})^2,$$

where π^1 is the subgradient at the starting point y^1.

The program can terminate in any one of several ways. The most desirable one is where the optimality criterion (1.4) is met for preassigned values of EPS, DX and DF1; we shall call this *Normal End.* The other termination criteria are:

(a) when the maximum number of iterations is reached (*MaxItr*),

(b) when the maximum number of calls to SIMUL (to compute dual function and subgradient) is reached (*MaxSim*),

(c) when the accuracy DX (i.e., input on required accuracy on successive y's) is reached without making any more improvement on g (*DxEnd*).

We have added another termination criterion after observing outputs from sample runs. Moving from one dual iterate to another requires on average two or three calls to SIMUL, and only rarely more than nine. Close to convergence, however, SIMUL can be called repeatedly because of numerical error; see [14] for details. We have thus imposed a limit of fifteen calls to SIMUL during the line search; this termination test is called *Max15.*

We have compiled this code on a DEC VAX 11/780 and on a DEC VAXstation II workstation, both running under the Berkeley UNIX (4.3 bsd) operating system, using the f77 compiler invoking −O option (optimizer). We have also compiled our code on a DEC VAX 11/780 running the VMS operating system using the VMS Fortran compiler "FOR". The version on UNIX-based machines used the single precision ZX0LP due to non-availability of the double precision version on the machine; however, the rest of the BUNDECOMP code is in double precision. On the other hand, the version of BUNDECOMP that runs on VMS is entirely in double precision.

Our implementation is mainly designed to study the performance of the algorithm and to compare it with other existing methods. It is an experimental code and does not have all the sophistication of a commercial code (like storage conservation, numerical stability and accuracy etc.).

4. Computational experience

To test the BBD algorithm, we have randomly generated test problems to present computational results (more results can be found in [17]). We have also done comparisons with MINOS version 5.0 [20] and DECOMP [7,11]. MINOS solves linear programming problems using a reliable implementations of the simplex maintaining a sparse LU factorization. It was developed at the Systems Optimizations Laboratory, Stanford University, by Murtagh and Saunders [20]. DECOMP is an implementation of the Dantzig–Wolfe decomposition method to solve block angular linear programming problems, developed by Ho and Loute. For documentation on DECOMP, see [7,11]. Computational experience with

DECOMP can be found in Ho and Loute [8–10]. Due to limitations of the version of DECOMP available to us, different sets of test problems were used in the comparisons with MINOS and DECOMP. First, we report computational experience with BUNDECOMP and its comparison with MINOS.

4.1. Comparison with MINOS

In Table 4.1, we give the specifications of the randomly generated problems tested with BUNDECOMP and MINOS. In this table, the sizes $m_i \times n_i$ of the subproblem (3.1) are rough estimates. For instance, consider example a2. The full size of this test problem is 350×500. Out of 350 constraints, ten are coupling constraints. Thus, the remaining 340 constraints and 500 variables are used for the block-angular structure. For the size, $m_i \times n_i$, of each subproblem, 340×500 is almost equally divided into forty subproblems which means the size of each subproblem is approximately 9×13. This approximate size is reflected in the column "approx. size of subproblems".

We present the solution time obtained using BUNDECOMP in CPU minutes as well as other pertinent information in Table 4.2. We report the number of dual iterations and the number of dual objective function (and subgradient) evaluations to reach the final solution, the final primal and dual objective value, the absolute and the relative accuracy (defined later in the paragraph) of the final dual subgradient, and the termination criterion. For each entry in the table, three different numbers are shown in the column "Time in Min.": the first entry is the total CPU time for solving the whole problem; the second entry is the time for solving all the subproblems for the first time (from scratch), i.e., first call to SIMUL; and the last entry is the average computing time for calls to SIMUL at subsequent steps. This shows that it is considerably faster to use parametric programming at subsequent steps. The absolute gradient accuracy is the infinity norm, i.e., $\|\pi\|_\infty = \max_{1 \leqslant i \leqslant m}\{ |\pi_i| \}$, where m is the dimension of the vector π. The relative accuracy is the measure $ra = \max_{1 \leqslant i \leqslant m}\{ |\pi_i / a_i| \}$, where a is the right hand side of the coupling constraints. The starting dual point is randomly chosen. From Table 4.2, observe that whenever the algorithm terminates with

Table 4.1
Problem specifications (A)

Problem name	Size of whole problem	Number of coupling constraints	Number of subproblems	Approx. size of subproblems	Density (%)
a1	350×500	10	100	3×5	3.83
a2	350×500	10	40	9×13	5.29
a3	350×500	30	100	3×5	9.49
a4	850×1500	10	100	8×15	2.16
a5	850×1500	10	40	21×38	3.65
a6	850×1500	30	100	8×15	4.49
a7	1250×2800	10	100	13×28	1.79
a8	1250×2800	10	40	31×70	3.28
a9	1250×2800	30	100	13×28	3.38
a10	4000×10000	10	100	40×100	3.25

Table 4.2
Output information from BUNDECOMP

Problem name	EPS	Objective value: primal/dual	ITER/NSIM	Gradient accuracy, absolute/relative	Time (Min)	Stopping rule
a1	0.05	-1066.0138	35	0.519×10^{-14}	1.37	Normal
		-1066.0502	92	0.103×10^{-15}	0.07	End
					0.01	
a2	1.0	-1260.5870	29	0.104×10^{-13}	1.81	Normal
		-1261.5869	73	0.113×10^{-15}	0.19	End
					0.02	
a3	1.0	-1083.2228	102	0.217×10^{1}	6.14	Max15
		-1084.1033	221	0.799×10^{-1}	0.06	
					0.02	
a4	0.5	-6362.6378	30	0.212×10^{1}	6.82	Max15
		-6363.1406	91	0.101	0.70	
					0.07	
a5 *	1.0	-7067.6398	20	0.151×10^{-13}	18.75	Normal
		-7068.6396	51	0.557×10^{-16}	3.23	End
					0.31	
a6	1.0	-6453.8673	60	0.324×10^{1}	13.85	Max15
		-6454.3823	147	0.552	0.60	
					0.09	
a7	1.0	-17194.2746	14	0.239×10^{-13}	11.55	Normal
		-17195.2734	42	0.177×10^{-15}	2.05	End
					0.23	
a8	1.0	-16687.6813	22	0.379×10^{1}	70.05	Max15
		-16688.6406	63	0.206×10^{-1}	13.13	
					0.92	
a9	5.0	-16926.6433	42	0.437×10^{1}	26.47	Max15
		-16931.6289	105	0.675×10^{-1}	2.12	
					0.23	
a10	0.5	-68400.6086	22	0.105	401.12	Max15
		-68401.1094	66	0.801×10^{-4}	80.57	
					4.93	

* DX is set at 10^{-6}; rest are set at 10^{-7}.
DF1 is set at 10^{-4}.

"Normal End", the accuracy of the "zero" element in the bundle, and hence the accuracy of the primal optimal solution, is very good. Recall the result on the duality gap from Theorem 2.6. If $\bar{d}$ is very close to zero, then the gap is on the order of ϵ. We observe from computational experience that the difference between the optimal dual and primal objective functions is on the order of the preassigned value of EPS. In other words, the pre-assigned value of EPS provides a good approximation to the duality gap in practice. In our work, we have chosen values of EPS from 0.05 to 5.0 as reported in Table 4.2 to primarily show the relation of EPS to the duality gap. A user can decide on a value of EPS that suits his/her requirement on the acceptable level of the duality gap. Note that $\bar{d}$ is the "zero" belonging to $\partial_\epsilon g(\bar{y})$. If $\bar{d}$ is not close enough to "zero", the method will try to take an ascent step

[14] to find a new iterate for the dual variable y, and then the QP (2.4) will be solved to obtain $\bar{d}$ closer to zero. Lemaréchal et al. remark in [14] that failing to solve the QP is rather exceptional and it happens only near the optimal solution to the problem. We have observed a similar behavior.

In Table 4.3, we present timing comparisons between BUNDECOMP and MINOS. The column "iterations" for MINOS denotes the total number of simplex iterations required to reach the optimal solution (the number in parenthesis is the number of iterations to do in phase I of the simplex method). The column "factor" is determined by dividing the MINOS time by the BUNDECOMP time. It should be noted that BUNDECOMP is a special purpose code to exploit the structure of problem (1.3) whereas MINOS is a general purpose LP code. The comparison is done with MINOS due to its wide availability and to obtain some insight on suitability of a special purpose method for solving (1.3). It may be noted that using some factorization scheme which exploits the special structure of the problem (1.3), a general purpose simplex code may possibly obtain better timing than obtained using MINOS. As suggested by Saunders [28] (reflected in results with MINOS in Table 4.3), we varied two parameters for MINOS software, PARTIAL PRICE and FACTORIZATION FREQUENCY, to reduce computing time (all the other parameters are set to the default values). For example, for test problem a4, changing these two parameters values (from default values) reduced the number of iterations by about 14% and run time to about a half of its original run time by using PARTIAL PRICE = 10 and FACTORIZATION FREQUENCY = 100.

Finally, we comment on the accuracy of the solution. BUNDECOMP terminates for user-prescribed tolerances (see Table 4.2). The tolerances for MINOS are set to the default values which are more stringent than the tolerances used in BUNDECOMP. Thus, in models where obtaining a low-accuracy solution is sufficient, BUNDECOMP appears to be a promising choice, especially due to the result on the duality gap.

Table 4.3
Comparison between BUNDECOMP and MINOS (time in minutes)

Problem name	BUNDECOMP time	MINOS		Factor
		iterations	time	
a1	1.37	630 (425)	8.82	6.44
a2	1.81	926 (625)	16.20	8.95
a3	6.14	1107 (653)	36.11	5.88
a4	6.82	2896 (1606)	130.87	19.19
a5	18.75	4181 (2332)	330.87	17.65
a6	13.85	4298 (2293)	378.37	27.32
a7	11.55	5799 (2713)	790.78	68.47
a8	70.05	8472 (3812)	1336.53	19.08
a9	26.47	10008 (4493)	1785.03*	67.44*
a10	401.12	na	na	–

* stopped at maximum iterations; na: MINOS could not be run due to the size of the problem.
Both MINOS and BUNDECOMP run on a DEC VAXstation II workstation.

4.2. Comparison with DECOMP

DECOMP is an implementation of the Dantzig–Wolfe decomposition method to solve block-angular linear programs. It was developed by Ho and Loute [7]. In both the Dantzig–Wolfe decompositions (DWD) method and the bundle-based decomposition method, the subproblems have the same form. We note here that the essential difference between these two methods is how the "master" problem is solved. In the DWD method, columns are generated from the solution of the subproblems and then a master linear program is solved. On the other hand, as described in Section 2.1 the approach taken in the BBD method is to solve a nonsmooth concave problem by a modified bundle method.

The problem set we tested both with DECOMP and BUNDECOMP is different than the set tested with MINOS. This is due to some restrictions in the version of DECOMP available to us, e.g., the maximum number of subproblems allowed and the number of nonzero elements in each subproblem. This version of DECOMP allowed at most six subproblems. We thus generated two sets of problems: the first set has six subproblems and the second set varies from twelve to sixty subproblems (see Table 4.4). For the test problems generated with more than six subproblems, we combine two or more into one "bigger" subproblem so that the "modified" total number of subproblems is not more than six. For example, consider test problem x5 in Table 4.4. Originally twelve subproblems are generated for this test problem. We aggregate them into six bigger subproblems by combining two subproblems into one bigger subproblem. This means the first two blocks of the original subproblems, whose feasible regions are

$$S^1 := \{x_1 \in \mathbb{R}^{n_1} \mid B_1 x_1 = b_1,\ x_1 \geqslant 0\}$$

and

$$S^2 := \{x_2 \in \mathbb{R}^{n_2} \mid B_2 x_2 = b_2,\ x_2 \geqslant 0\}\ ,$$

are combined into one feasible region in $\mathbb{R}^{n_1+n_2}$ and the constraints are considered to be

Table 4.4
Problem Specifications (x)

Problem name	Size of whole problem	Number of coupling constraints	Number of subproblems	Approx. size of subproblems	Density (%)
x1	150×300	10	6	23×50	22.22
x2	150×300	20	6	23×50	27.78
x3	200×450	10	6	32×75	20.83
x4	200×450	20	6	32×75	25.00
x5	320×550	10	12	25×46	11.20
x6	320×550	20	12	25×46	14.06
x7	500×900	10	60	8×15	3.63
x8	500×900	20	60	8×15	5.60
x9	600×1200	30	60	10×20	3.31

$$\begin{pmatrix} B_1 & 0 \\ 0 & B_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geqslant 0 .$$

Similarly, we combine three and four together, five and six together and so on. Accordingly, test problem x5 with 12 subproblems, each of approximate size 25×46, is regarded as a problem with six "bigger" subproblems, each of size 50×92 for DECOMP. (See Fig. 1 for a pictorial representation of subproblem blocks where four smaller subproblems are combined to form a bigger subproblem.)

Theoretically, multiple proposals that pass a candidacy test (see [11, pp. 23–29] for details) can be sent from a subproblem to the DWD master problem. The version of DECOMP we used takes a proposal from each subproblem. Also this version of DECOMP does *not* obtain a primal optimal solution whereas BUNDECOMP obtains an approximate primal optimal solution. (In the DWD method, a primal optimal solution to the original problem is usually recovered by solving N LP subproblems *once more* [phase three] – this time each with $(m_i + m)$ constraints (instead of m_i constraints) and n_i variables ([2,7,11]) – after satisfying the optimality conditions.)

All the test problems listed in Table 4.4 were run on a VAX 11/780 running the VMS operating system with both BUNDECOMP and DECOMP. The first four test problems have six subproblems each. The computational results of these four test problems (i.e., x1, x2, x3, x4) with both DECOMP and BUNDECOMP are reported in Table 4.5.

The rest of the test problems (x5, x6, x7, x8, x9) from Table 4.4 have more than six subproblems each. DECOMP solves these test problems assuming each of them to consist of six "bigger" subproblems, as described above. With BUNDECOMP, we ran the problems both ways: (a) with the original smaller subproblems, and (b) also with six "bigger" subproblems. For example, test problem x8 is solved with six subproblems, each of size 80×150, and also with sixty subproblems, each of size 8×15 (see Fig. 1 to understand the difference). As noted before, the subproblems to be solved in both methods are linear programs of similar size and structure. In DECOMP, the subproblems are solved using a sophisticated implementation of the revised simplex method by Tomlin [7] storing only

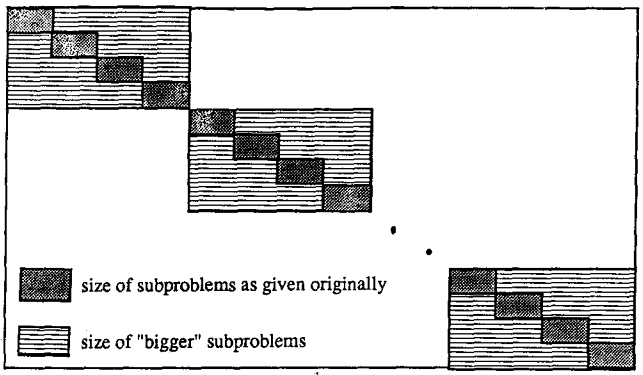
size of subproblems as given originally
size of "bigger" subproblems

Fig. 1. Two different sites of subproblems as considered by BUNDECOMP for solving LP subproblems (refer to Section 4.2 and Table 4.6).

Table 4.5
Comparison of DECOMP and BUNDECOMP (six subproblems to start with)

Problem name	DECOMP		BUNDECOMP		Factor $= T_d/T_b$
	Time T_d (min)	Number of cycles	Time T_b (min)	Number of iterations	
x1	5.75	26	1.90	35	3.023
x2	9.12	32	2.37	52	3.854
x3	14.14	30	4.78	33	2.959
x4	24.55	43	5.22	38	4.703

Table 4.6
Comparison of DECOMP and BUNDECOMP (more than six subproblems to start with)

Problem name	DECOMP		BUNDECOMP			
			As given originally		As six subproblems	
	Time (min)	Number of cycles	Time (min)	Number of iterations	Time (min)	Number of iterations
x5	12.20	27	3.03	18	10.79	19
x6	26.07	47	4.49	50	13.58	52
x7	9.45	28	1.01	23	28.13	19
x8	25.97	55	1.86	41	34.09	39
x9	19.94	37	1.31	20	na	na

na: = not available as ZXOLP failed to solve subproblems of size 95×200.

the nonzero entries of the coefficient matrix. Recall that, in BUNDECOMP, the linear programming subproblems are solved using IMSL routine, ZX0LP, which stores the coefficient matrix in dense form. Thus, in the case when the subproblems are aggregated to form bigger subproblems, ZX0LP cannot take full advantage of the structure.

We have tested problems x5, x6, x7, x8 and x9 from Table 4.4 with DECOMP and two versions of BUNDECOMP described above. We report results on computational time in Table 4.6. The objective function values obtained using BUNDECOMP and DECOMP are of the same order (match to four significant digits). Note that the number of iterations with the two versions of BUNDECOMP differ. This difference is due to the fact that the solutions of the aggregated linear programming subproblems may not be the same as the solutions of the smaller linear programming subproblems put together. Thus, the two version may take different trajectories to the dual optimal solution. Finally, recall that DECOMP does not obtain a primal optimal solution.

5. Discussion

In this paper, we presented a posteriori error estimates on the approximate primal optimal solution and on the duality gap for the bundle-based decomposition method for solving convex optimization problems of type (1.1). We discussed development and implementation of an experimental code BUNDECOMP to solve a special case of this type of problems, namely, block-angular linear programming problems (1.3). The BBD method appears to be promising based on our limited experience, especially when a low-accuracy solution is sufficient, and compares favorably with existing methods. The computational experience with the code shows that the pre-assigned ϵ provides a good approximation to the duality gap in practice. We also discussed the essential difference between this method and the Dantzig–Wolfe decomposition method. Replacing the routine ZX0LP (used as a part of BUNDECOMP) by a more efficient routine which exploits the structure of the subproblems may possibly enhance the performance of the code. Similarly, MINOS or DECOMP may be specifically tailored to solve (1.3) more efficiently. Note also that we have conducted our study on randomly generated test problems. More studies need to be done to see the performance of the method on real-life problems. Finally, we note that this decomposition method leads to developing parallel algorithms ([17,18]) which produce high speedup and efficiency for large problems.

Acknowledgements

Most of the work was done while the author was at the University of Wisconsin-Madison, supported by the National Science Foundation under Grant No. CCR-8502202. The author would like to thank S.M. Robinson for his guidance and encouragement throughout the course of this work and for careful reading of a draft of this paper. Thanks are due to C. Lemaréchal for providing M1FC1 and J.K. Ho for providing DECOMP and R.P. Sundarraj for help with DECOMP. The author would also like to thank two anonymous referees and an anonymous associate editor for many helpful suggestions which improved the content and the presentation of the paper considerably.

References

[1] G.B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations Research* 8 (1960) 101–111.
[2] Y.M.I. Dirickx and L.P. Jennergren, *Systems Analysis by Multilevel Methods: With Applications to Economics and Management* (Wiley, Chichester, England, 1979).
[3] P.E. Gill, W. Murray and M.H. Wright, *Practical Optimization* (Academic Press, New York, 1981).
[4] C.D. Ha, "Decomposition methods for structured convex programming," Ph.D. dissertation, Department of Industrial Engineering, University of Wisconsin-Madison (Madison, WI, 1980).
[5] M. Held, P. Wolfe and H. Crowder, "Validation of subgradient optimization," *Mathematical Programming* 6 (1974) 62–88.
[6] J.-B. Hiriart-Urruty, "ϵ-subdifferential calculus," in: *Convex Analysis and Optimization*, Research note in Mathematics, Series 57 (Pitman, London, 1982) pp. 43–92.
[7] J.K. Ho and É. Loute, "DECOMP User's Guide," unpublished manuscript.
[8] J.K. Ho and É. Loute, "An advanced implementation of the Dantzig–Wolfe decomposition algorithm for linear programming," *Mathematical Programming* 20 (1981) 303–326.

[9] J.K. Ho and É. Loute, "Computational experience with advanced implementation of decomposition algorithms for linear programming," *Mathematical Programming* 27 (1983) 283–290.

[10] J.K. Ho and É. Loute, "Computational aspects of dynamico: a model of trade and development in the world economy," *Revue Française d'Automatique, Informatique et Recherche opérationnelle* 18 (1984) 403–414.

[11] J.K. Ho and R.P. Sundarraj, *DECOMP: An Implementation of Dantzig–Wolfe Decomposition for Linear Programming* (Lecture Notes in Economics and Mathematical Systems, Vol. 338) (Springer-Verlag, New York, 1989).

[12] IMSL User's manual, Edition 9.2, International Mathematical and Statistical Library, Houston, Texas (1984).

[13] C. Lemaréchal, "A view of line-searches", in: Auslender, Oettli and Stoer, eds., *Optimization and Optimal Control* (Lecture Notes in Control and Information Sciences, Vol. 30) (Springer-Verlag, Berlin, 1981) pp. 59–78.

[14] C. Lemaréchal, J.J. Strodiot and A. Bihain, "On a bundle algorithm for nonsmooth optimization," in: O.L. Mangasarian, R.R. Meyer and S.M. Robinson, eds., *Nonlinear Programming 4* (Academic Press, New York, 1981) pp. 245–282.

[15] C. Lemaréchal and M.-C. Bancora Imbert, "Le module M1FC1", preprint: INRIA, B.P. 105, Le Chesnay, France (France, March 1985).

[16] C. Lemaréchal, Private communication (1986).

[17] D. Medhi, "Decomposition of structured large-scale optimization problems and parallel optimizations," Ph.D. dissertation, Technical Report #718, Computer Sciences Dept., University of Wisconsin-Madison (September 1987).

[18] D. Medhi, "Parallel bundle-based decomposition algorithm for large-scale structured mathematical programming problems," *Annals of Operations Research* 22 (1990) 101–127.

[19] R. Mifflin, "A stable method for solving certain constrained least-squares problems," *Mathematical Programming* 16 (1979) 141–158.

[20] B.A. Murtagh and M.A. Saunders, "MINOS 5.0 user's guide", Technical Report #SOL 83-20, Department of Operations Research, Stanford University (Stanford, CA, December 1983).

[21] K.G. Murty, *Linear Programming* (John Wiley and Sons, New York, 1983).

[22] B.T. Polyak, "Subgradient method: a survey of Soviet research," in: C. Lemaréchal and R. Mifflin, eds., *Nonsmooth Optimization* (Pergamon Press, Oxford, 1978) pp. 5–28.

[23] S.M. Robinson, "Bundle-based decomposition: Description and preliminary results," in: A. Prékopa, J. Szelezsán and B. Strazicky, eds., *System Modelling and Optimization* (Lecture Notes in Control and Information Sciences, Vol. 84) (Springer-Verlag, Berlin, 1986) pp. 751–756.

[24] S.M. Robinson, "Bundle-based decomposition: conditions for convergence," *Annales de l'Institute Henri Poincaré: Analyse Non Linéaire* 6 (1989) 435–447.

[25] R.T. Rockafellar, *Convex Analysis* (Princeton University Press, 1970).

[26] R.T. Rockafellar, *Conjugate Duality and Optimization* (SIAM, Philadelphia, 1974).

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

[28] M.A. Saunders, Private communication (1987).