Operations Research Letters 44 (2016) 80–85

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Benders decomposition: Solving binary master problems by enumeration

D. Antony Tarvin [a], R. Kevin Wood [b], Alexandra M. Newman [c,*]

[a] White Sands Missile Range, NM 88002, USA

[b] Operations Research Department, Naval Postgraduate School, Monterey, CA 93943, USA

[c] Mechanical Engineering Department, Colorado School of Mines, Golden, CO 80401, USA

ARTICLE INFO

Article history:
Received 3 February 2014
Received in revised form
24 August 2015
Accepted 18 November 2015
Available online 27 November 2015

Keywords:
Benders decomposition
Explicit enumeration
Facility location

ABSTRACT

We develop a variant of Benders decomposition for mixed-integer programming that solves each master problem by explicit enumeration. By storing the master problem's current objective-function value for each potential solution, computational effort remains essentially constant across iterations. Using both serial and parallel processing, tests against competing methods show computational speedups that exceed two orders of magnitude.

1. Introduction

Many mixed-integer programs (MIPs) solve efficiently only through decomposition (e.g., [27]). Benders decomposition is typical [3], and has been employed in such applications as distribution planning [11], power-flow optimization [1] and preventive-maintenance scheduling [6]. Magnanti and Wong [16] note that a key computational challenge for the Benders decomposition algorithm (**BD**) is solving the (*relaxed*) *mixed-integer master problem* repeatedly. Each iteration of **BD** introduces a new constraint, a *Benders cut*, which increases the master problem's size and, thus, its worst-case solution time [18, p. 125]. A typical **BD** implementation solves the master problems by standard, linear-programming-based branch-and-bound (**B&B**). Our paper investigates how the decomposition algorithm's empirical efficiency may improve by solving those master problems through the explicit enumeration of feasible solutions.

A number of techniques have been developed to improve **BD**'s computational efficiency, for example: (i) strengthening master-problem cuts by exploiting interior-point subproblem solutions [17]; (ii) adding knapsack cover cuts based on Benders cuts [23]; (iii) deleting Benders cuts from early iterations that become "unattractive" [10]; and (iv) reformulating to yield a "multicut master problem", which possesses one group of cuts for each independent subproblem [4]. Rather than modifying master-problem formulations, however, we seek a faster method to solve master problems, a method that applies to standard and multicut formulations, and which should benefit from enhancements such as the inclusion of knapsack cuts or other integer cutting planes. Because of this focus, we compare our method against two important, existing variants of **BD** that also modify how master problems are solved: (i) *master-problem suboptimization*, which relaxes standard optimality requirements for the master problem [11]; and (ii) *Benders branch-and-cut*, which adds Benders cuts within the branch-and-bound enumeration tree of a single master problem [25]. We describe both techniques in more detail later.

Salmeron and Wood [20] first suggest the technique investigated in our paper: converting standard **BD** into "**BD-E**" by solving **BD**'s master problem by explicit enumeration. This technique has clear computational limitations, but those limitations may not be relevant in many settings. For example, models for infrastructure protection and/or interdiction [5] often describe a "target-rich but resource-poor environment", in which resources limit the number of targets (e.g., bridges, electric power substations) that might be attacked or protected. In this case, a feasible master-problem solution corresponds exactly to a resource-feasible attack or protection plan, so enumeration of each such plan and evaluation in the relaxed master problem may be computationally viable [20]. (Evaluating each plan exactly through the Benders "subproblem" would be computationally prohibitive, however.)

* Corresponding author.
E-mail addresses: tony.tarvin@hotmail.com (D.A. Tarvin), kwood@nps.edu (R.K. Wood), anewman@mines.edu (A.M. Newman).

http://dx.doi.org/10.1016/j.orl.2015.11.009
0167-6377/

The implementation of the enumeration procedure in [20] produces a decomposition algorithm that does not consistently outperform standard **BD**. That paper restricts itself, however, to programming the enumeration using a non-compiled, algebraic modeling language, which employs only serial processing. To examine the technique more completely, we implement both serial and parallel algorithms using a compiled programming language and test in the context of an interdiction model, as well as a two-stage stochastic program.

2. Background on Benders decomposition

To facilitate subsequent development, this section describes a standard **BD** to solve a MIP. For simplicity, we limit attention to a MIP whose discrete variables are binary and which is assumed to have a finite optimal solution:

$$MIP \quad z^* = \min_{\mathbf{x}\in\mathcal{X},\ \mathbf{y}\in\mathcal{R}_+^{n_y}} \mathbf{cx} + \mathbf{fy} \tag{1}$$

$$\text{s.t. } B\mathbf{x} + D\mathbf{y} \geq \mathbf{d}, \tag{2}$$

where $\mathcal{X} \equiv \{\mathbf{x} \in \{0,1\}^{n_x} | A\mathbf{x} \leq \mathbf{b}\}$; $\mathcal{R}_+^n$ denotes the n-dimensional, non-negative real space; A, B, and D are dimensioned $m_1 \times n_x$, $m_2 \times n_x$, and $m_2 \times n_y$, respectively; and the vectors **b**, **c**, **d**, **f**, **x**, and **y** conform. In addition, we require an efficient enumeration algorithm for $\mathcal{X}$, which should also have modest cardinality, say $|\mathcal{X}| < 10^9$. (Parallel processing might relax the cardinality limit by several orders of magnitude.) Our two test problems define $\mathcal{X}$ through single knapsack constraints, so efficient enumeration is straightforward. More complex constraint sets defining $\mathcal{X}$ are certainly possible, and we suggest those used in [11,24] as good examples.

Fixing $\mathbf{x} = \hat{\mathbf{x}} \in \mathcal{X}$ in *MIP* yields the following *subproblem*:

$$SUB(\hat{\mathbf{x}}) \quad z^*_{\text{SUB}}(\hat{\mathbf{x}}) = \min_{\mathbf{y}\in\mathcal{R}_+^{n_y}} \mathbf{c}\hat{\mathbf{x}} + \mathbf{fy} \tag{3}$$

$$\text{s.t. } D\mathbf{y} \geq \mathbf{d} - B\hat{\mathbf{x}}. \quad [\boldsymbol{\alpha}(\hat{\mathbf{x}})] \tag{4}$$

where: (i) in brackets and next to its corresponding constraints, $\boldsymbol{\alpha}(\hat{\mathbf{x}})$ represents the optimal dual vector produced by our linear-programming solver; (ii) we will use $\mathcal{A} \equiv \cup_{\hat{\mathbf{x}}\in\mathcal{X}}\{\boldsymbol{\alpha}(\hat{\mathbf{x}})\}$; (iii) $\hat{\boldsymbol{\alpha}} \equiv \boldsymbol{\alpha}(\hat{\mathbf{x}})$, with the argument suppressed for simplicity; and (iv) $\hat{\mathbf{y}}$ denotes the optimal primal solution given $\hat{\mathbf{x}}$ (with no argument for the sake of simplicity).

Assumption 1. $SUB(\hat{\mathbf{x}})$ is feasible $\forall\, \hat{\mathbf{x}} \in \mathcal{X}$. ■

Assumption 1 corresponds to "relatively complete recourse" in the stochastic-programming literature (e.g., [28]). While convenient for exposition, the assumption is not limiting. If it does not hold, then **BD** can (i) introduce "feasibility cuts" to the master problem as necessary [27], or (ii) solve a standard, reformulated model that penalizes constraint violations in the subproblem. Section 6 explains how to implement feasibility cuts in **BD-E**.

Under Assumption 1, **BD** reformulates *MIP* into this *equivalent master problem*:

$$MP(\mathcal{A}) \quad z^* = z^*_{\text{MP}}(\mathcal{A}) = \min_{\mathbf{x}\in\mathcal{X},\,\eta} \eta \tag{5}$$

$$\text{s.t. } \eta \geq \mathbf{cx} + \hat{\boldsymbol{\alpha}}(\mathbf{d} - B\mathbf{x}) \quad \forall\, \hat{\boldsymbol{\alpha}} \in \mathcal{A}. \tag{6}$$

Rather than exhaustively computing the elements of $\mathcal{A}$ and then solving $MP(\mathcal{A})$ directly, a general version of **BD** begins with an arbitrary set $\hat{\mathcal{A}} \subset \mathcal{A}$ and with an initial feasible solution $\hat{\mathbf{x}} \in \mathcal{X}$. Then, iteratively, **BD** (i) solves $SUB(\hat{\mathbf{x}})$ for $\hat{\mathbf{y}}$ and $\hat{\boldsymbol{\alpha}}$, (ii) adds $\hat{\boldsymbol{\alpha}}$ to $\hat{\mathcal{A}}$, and (iii) solves the *relaxed master problem* $MP(\hat{\mathcal{A}})$ for $\hat{\mathbf{x}}$. The algorithm terminates once $MP(\hat{\mathcal{A}})$ approximates $MP(\mathcal{A})$ well enough to prove near-optimality of some previously discovered $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ [3]. "Algorithm 1" summarizes **BD** for later reference; we refer the reader to Benders' original paper [3] for a proof of correctness.

Algorithm 1—Benders decomposition algorithm (**BD**)

Input:	An instance of *MIP* and allowable optimality gap $\varepsilon \geq 0$.
Output:	An ε-optimal solution to *MIP*.
Step 0:	**Initialization**
a.	$K \leftarrow 1$; $\underline{z} \leftarrow -\infty$; $\bar{z} \leftarrow \infty$; $\hat{\mathcal{A}}_0 \leftarrow \varnothing$; $\hat{\mathbf{x}}_1 \leftarrow$ any $\mathbf{x} \in \mathcal{X}$;
b.	Dummy step for initialization;
Step 1:	**Subproblem**
	Solve $SUB(\hat{\mathbf{x}}_K)$ for $\hat{\mathbf{y}}_K$, $\hat{\boldsymbol{\alpha}}_K$ and $z^*_{\text{SUB}}(\hat{\mathbf{x}}_K)$;
	$\hat{\mathcal{A}}_K \leftarrow \hat{\mathcal{A}}_{K-1} \cup \{\hat{\boldsymbol{\alpha}}_K\}$;
	If ($z^*_{\text{SUB}}(\hat{\mathbf{x}}_K) < \bar{z}$) { $\bar{z} \leftarrow z^*_{\text{SUB}}(\hat{\mathbf{x}}_K)$;
	$(\mathbf{x}^*, \mathbf{y}^*) \leftarrow (\hat{\mathbf{x}}_K, \hat{\mathbf{y}}_K)$; }
	If ($\bar{z} - \underline{z} \leq \varepsilon$) go to Step 3;
Step 2:	**Master Problem**
a.	Solve $MP(\hat{\mathcal{A}}_K)$ for $\hat{\mathbf{x}}_{K+1}$ and $z^*_{\text{MP}}(\hat{\mathcal{A}}_K)$;
b.	$\underline{z} \leftarrow z^*_{\text{MP}}(\hat{\mathcal{A}}_K)$;
c.	If ($\bar{z} - \underline{z} \leq \varepsilon$) go to Step 3;
d.	$K \leftarrow K + 1$; Go to Step 1;
Step 3:	**Print Solution**
	Print ("ε-optimal solution to *MIP* is", $(\bar{z}, \mathbf{x}^*, \mathbf{y}^*)$); Halt.

3. Solving $MP(\hat{\mathcal{A}}_K)$ by enumeration

Typically, **BD** solves the Kth relaxed master problem $MP(\hat{\mathcal{A}}_K)$ by **B&B**, but if $|\mathcal{X}|$ is sufficiently small, we will demonstrate that explicit enumeration can solve $MP(\hat{\mathcal{A}}_K)$ more efficiently. The comparative efficiency of enumeration will become apparent as the number of cuts in $MP(\hat{\mathcal{A}}_K)$ increases with increasing K and **B&B** slows dramatically.

Let $\hat{\boldsymbol{\alpha}}_k$ denote the dual vector generated in **BD**'s kth iteration. The master problem in iteration K is then

$$MP(\hat{\mathcal{A}}_K) \quad z^*_{\text{MP}}(\hat{\mathcal{A}}_K) = \min_{\mathbf{x}\in\mathcal{X},\,\eta} \eta \tag{7}$$

$$\text{s.t. } \eta \geq g_0(\hat{\mathbf{x}}_k) + \sum_{j=1}^{n_x} g_j(\hat{\mathbf{x}}_k)x_j \quad \forall\, k \in \{1,\ldots,K\}, \tag{8}$$

where, letting $B_{\cdot j}$ denote B's jth column, $g_j(\hat{\mathbf{x}}_k) \equiv -\hat{\boldsymbol{\alpha}}_k B_{\cdot j} + c_j$ for $j = 1,\ldots,n_x$, and $g_0(\hat{\mathbf{x}}_k) \equiv \hat{\boldsymbol{\alpha}}_k\mathbf{d}$. Equivalently, $MP(\hat{\mathcal{A}}_K)$ may be written as

$$MP(\hat{\mathcal{A}}_K) \quad z^*_{\text{MP}}(\hat{\mathcal{A}}_K) = \min_{\mathbf{x}\in\mathcal{X}} \eta_K(\mathbf{x}), \quad \text{where} \tag{9}$$

$$\eta_K(\mathbf{x}) = \max_{k=1,\ldots,K}\left\{g_0(\hat{\mathbf{x}}_k) + \sum_{j=1}^{n_x} g_j(\hat{\mathbf{x}}_k)x_j\right\}, \tag{10}$$

and where $\hat{\mathbf{x}}_{K+1} = \text{argmin}_{\mathbf{x}\in\mathcal{X}}\eta_K(\mathbf{x})$.

Now, rather than solving $MP(\hat{\mathcal{A}}_K)$ "from scratch" in each iteration as a mathematical program defined by (7) and (8), Eqs. (9) and (10) lead to a simple "update computation" for evaluating $\eta_K(\mathbf{x})$, and thus solving $MP(\hat{\mathcal{A}}_K)$:

$$\eta_K(\mathbf{x}) \leftarrow \max\left\{\eta_{K-1}(\mathbf{x}),\ g_0(\hat{\mathbf{x}}_K) + \sum_{j=1}^{n_x} g_j(\hat{\mathbf{x}}_K)x_j\right\} \quad \forall\, \mathbf{x} \in \mathcal{X}. \tag{11}$$

By storing $\eta_{K-1}(\mathbf{x})$ for each $\mathbf{x} \in \mathcal{X}$, the computational effort to solve $MP(\hat{\mathcal{A}}_K)$ will not increase with K, as it tends to with standard **B&B**. Algorithm 2 presents the modifications of Algorithm 1 that convert standard **BD** into **BD-E**. **BD-E** is simply a particular implementation of **BD**, so Benders' proof of correctness [3] still applies.

To solve $MP(\hat{\mathcal{A}})$ as described above, **BD-E** must generate every $\mathbf{x} \in \mathcal{X}$. Let us assume that $\mathcal{X}$ is defined through a single knapsack constraint

$$\mathcal{X} \equiv \left\{\mathbf{x} \in \{0,1\}^{n_x} \mid \mathbf{ax} \leq b\right\}, \tag{12}$$