PROBABILISTIC OPTIMIZATION VIA APPROXIMATE *P*-EFFICIENT POINTS AND BUNDLE METHODS

W. VAN ACKOOIJ*, V. BERGE‡, W. DE OLIVEIRA§‖, AND C. SAGASTIZÁBAL§

Abstract. For problems when decisions are taken prior to observing the realization of underlying random events, probabilistic constraints are an important modelling tool if reliability is a concern. A key concept to numerically dealing with probabilistic constraints is that of *p*-efficient points. By adopting a dual point of view, we develop a solution framework that includes and extends various existing formulations. The unifying approach is built on the basis of a recent generation of bundle methods called with on-demand accuracy, characterized by its versatility and flexibility. Numerical results for several difficult probabilistically constrained problems confirm the interest of the approach.

Key words. Probabilistic constraints, Stochastic programming, Duality, Bundle methods, *p*-efficient points

AMS subject classifications. 49M29, 49M37, 65K05, 90C15

1. Introduction. Probabilistic constraints arise in many real-life problems, for example electricity network expansion, mineral blending, chemical engineering, [2, 30, 35, 36]. Typically, these constraints are used when in an ordinary inequality system certain random parameters are identified as critical for the decision making process. We are interested in the so-called *separable* case, in which the random quantities appear only on one side of the constraint. This amounts to dealing with a feasible set of the form

$$C := \left\{ x \in \mathbb{R}^n \ : \mathbb{P}[g(x) \geq \xi] \geq p \right\}, \tag{1.1}$$

where $g : \mathbb{R}^n \to \mathbb{R}^m$ is a constraint mapping, $x \in \mathbb{R}^n$ the given a decision vector, $\xi \in \mathbb{R}^m$ a random vector with associated probability measure $\mathbb{P}$, and $p \in (0,1]$ a pre-specified probability level. Since the mapping $x \mapsto \psi(x) := \mathbb{P}[g(x) \geq \xi]$ is nonlinear, writing the constraint in the form $\psi(x) \geq p$ makes (1.1) appear as the feasible set of a conventional nonlinear programming problem. However this writing neglects a hidden difficulty: in most situations explicit values are *not available*. Furthermore, often calculation are inexact, as computing the probability $\psi(x)$ for a given point x typically involves some sort of numerical integration and/or (quasi) Monte Carlo methods.

Another issue is that the feasible set (1.1) sometimes fails to be convex; we refer to [1, 19] for conditions ensuring convexity for sufficiently large probability levels. In order to deal with convex feasible sets regardless of the probability value, throughout we suppose that $g(\cdot)$ is concave and the ξ-density has generalized concavity properties (like the multivariate Gaussian and Student densities) ; see [9].

An overview of the theory and numerical treatment of probabilistic constraints can be found in [33,34]. Regarding solution methods, the first approaches, based on cutting planes, can be found in [35,43]. More recently, the specific case of a linear constraint mapping was addressed by sample average approximations in [25, 26] and by scenario approximations in [6, 7].

In this work we follow the lead of [9, 12] and consider a third class of solution methods, based on the notion of *p-efficient points*, a quantile generalization. The set of *p*-efficient points is defined as follows:

$$\mathcal{V} := \left\{ v \in \mathcal{Z} : \text{ no } w \text{ exists in } \mathcal{Z} \text{ such that } w < v \right\} \text{ where } \mathcal{Z} := \{ v \in \mathbb{R}^m \ : \mathbb{P}[\xi \leq v] \geq p \}$$

is the level set of the probability distribution. Even when the set $\mathcal{V}$ contains a finite number of elements its full identification can prove difficult to the extent that in many situations it is only affordable to compute *one* *p*-efficient point per iteration. Methods based on *p*-efficient points approximate iteratively the feasible set in (1.1) by generating points x satisfying the relation

$$g(x) \geq v \text{ for some } p\text{-efficient point } v \in \mathcal{V}.$$

From an optimization perspective, knowing the whole set $\mathcal{V}$ does not really matter: only *p*-efficient points bounding the constraint near a solution are of interest. For this reason, it is sound to find such points iteratively, in a manner similar to the column-generation technique in Linear Programming.

In our understanding, the combination of those ideas with regularization techniques is behind the excellent results reported in [11, 12], confirmed in our numerical experiments in Section 6 below.

*EDF R&D. OSIRIS, 1, avenue du Général de Gaulle, F-92141 Clamart Cedex France.
‡Student at ENSTA ParisTech, 828 Boulevard des Maréchaux, 91120 Palaiseau, France.
§IMPA, Estrada Dona Castorina 110, 22460-320, Rio de Janeiro, Brazil.
‖BCAM, Alameda de Mazarredo, 14, 48009 Bilbao, Basque Country, Spain.

Our study, which was motivated by the outstanding performance of those approaches, reveals several interesting relations with bundle methods, [21]. Thanks to this connection, we develop a general framework that includes and extends various existing methods. The unifying view uses the proximal bundle theory [32], suitable for methods refered to as dealing with *on-demand accuracy*. This recent bundle variant was designed to solve nonsmooth problems for which the *oracle* providing information can perform its calculations with varying precision, following the directives of the bundle solver. In our setting, this amounts to solving dual formulations of the probabilistically constrained problem of interest by computing p-efficient points with variable accuracy. The interest of this technique is in the fact that it is possible to solve the problem *exactly* by starting the algorithmic procedure with coarse estimations, making the oracle calculations increasingly more exact as the iterations progress. On-demand algorithms keep the convergence properties of classical bundle methods and, as shown by our numerical experiments, can provide very significant gains in CPU time without losing accuracy in the solution.

This paper is organized as follows. Section 2 revises briefly concepts and methods in probabilistic optimization and sets the background and notation necessary to bundle methods. Section 3 presents several variants of these methods, when the oracle information is inexact at iterations yielding certain null steps (see Step 4 in Algorithm 1 below). In addition, this section gives primal and dual convergence results and explains the relation with the regularized dual decomposition [11] and the Progressive Augmented Lagrangian method [12]. Section 4 gives a general algorithm, suitable for inexact oracle calculations (at all iterations, including certain serious steps), which converges to primal and dual solutions, up to the oracle error. Section 5 discusses oracles that compute, in an on-demand mode, approximate p-efficient points. Both discrete and continuous distributions are considered in this section. Section 6 studies the performance of ten different solvers that fit our general framework. The comparison is done on several instances of cash-matching, cascaded reservoir management, and probabilistic transportation problems. A thorough analysis, reporting CPU times and quality of the solution both in terms of optimality and feasibility, gives a clear panorama of the merits of the different methods in the benchmark.

Our notation follows [21]. The inner product and induced norm are $\langle\cdot,\cdot\rangle$ and $|\cdot|$. For a convex function f, a point $u\in\mathbb{R}^m$ and $\eta\geq 0$, the exact and approximate convex analysis subdifferentials are denoted by $\partial f(u)$ and $\partial_\eta f(u)$. For a concave function φ we consider the corresponding objects of its negative, i.e., $\partial(-\varphi)(u)$ and $\partial_\eta(-\varphi)(u)$. The normal cone of the nonnegative orthant in $\mathbb{R}^m$ at u is $N_{\mathbb{R}^m_+}(u)=\{p\in\mathbb{R}^m : p\leq 0 \text{ and } \langle p,u\rangle=0\}$.

2. The probabilistic optimization problem and preliminary concepts. We recall background material relative to probabilistic constraints and to bundle algorithms, respectively from [9, 10, 12] and [32].

2.1. Blanket conditions. We suppose the multivariate random variable $\xi\in\mathbb{R}^m$ has an α-concave distribution, so that the following relations, from [9, Thms. 4.42, 4.60, 4.63, and Lems.4.57 and 4.59], hold:

$$\mathcal{Z}=\bigcup\left\{v+\mathbb{R}^m_+ : v\in\mathcal{V}\right\} \text{ is convex, nonempty and closed, } \operatorname{conv}\mathcal{V}\subset\mathcal{Z}, \text{ and } \mathcal{V} \text{ is bounded from below.}$$

Given a convex function $c:\mathbb{R}^n\to\mathbb{R}$, a nonempty and simple convex compact set $X\subset\mathbb{R}^n$ is (for example a polyhedron), and a concave mapping $g:\mathbb{R}^n\to\mathbb{R}^m$, we suppose that our problem of interest

$$\left\{\begin{array}{ll}\min\limits_{x\in X} & c(x)\\ \text{s.t.} & \mathbb{P}[g(x)\geq\xi]\geq p\,,\end{array}\right. \tag{2.1}$$

has a nonempty solution set and, hence, a finite optimal value, c^*.

In view of the relations above for $\mathcal{Z}$, the problem below, obtained by variable splitting, is convex:

$$\left\{\begin{array}{ll}\min\limits_{(x,v)\in\mathbb{R}^{n\times m}} & c(x)\\ \text{s.t.} & g(x)\geq v\\ & x\in X \text{ and } v\in\mathcal{Z}\,.\end{array}\right. \tag{2.2}$$

For this problem to be well-defined, nonemptiness of the feasible set is usually ensured by a Slater condition. If boundedness of the multipliers in (2.2) is a concern, the stronger (Mangasarian-Fromowitz-like) constraint qualification can be used, [21, Thm.VII.2.3.2]:

$$\exists(x^s,v^s)\in X\times\mathcal{Z} : g(x^s)>v^s \text{ and, in } X\times\mathcal{Z},\ \left\{\begin{array}{l}\text{affine equality constraints are linearly independent}\\ \text{inequality constraints are satisfied strictly by } (x^s,v^s)\,.\end{array}\right. \tag{2.3}$$

For the problem on dual variables

$$\max_{u\in\mathbb{R}^m_+}\varphi(u)\ \text{ with }\ \varphi(u):=h(u)+d(u),\ \text{where}\ \left\{\begin{array}{l} h(u):=\min\limits_{x\in X}\{c(x)-\langle u,g(x)\rangle\}\\ d(u):=\min\limits_{v\in\mathcal{V}}\langle u,v\rangle\ ,\end{array}\right. \tag{2.4}$$

defining d as a minimum is possible ($u\geq 0$ and $\mathcal{V}$ has a lower bound). Also, by the disjunctive expression for $\mathcal{Z}$,

$$d(u)=\min_{v\in\mathcal{V}}\langle u,v\rangle=\min_{v\in\operatorname{conv}\mathcal{V}}\langle u,v\rangle=\min_{v\in\mathcal{Z}}\langle u,v\rangle\qquad\text{for all } u\in\mathbb{R}^m_+, \tag{2.5}$$

because the minimand is linear. As a result of these relations, φ in (2.4) coincides with the dual function of problem (2.2). By weak duality using the point (x^s,v^s) from (2.3), the dual function is bounded above ($\varphi(u)\leq c^*$ for all $u\in\mathbb{R}^m_+$), thus justifying the use of a maximum instead of a supremum in (2.4). Furthermore, since there is no duality gap between (2.2) and (2.4), by [12, Cor. 1] both (2.1) and (2.2) have (2.4) as dual problem and the respective optimal values coincide.

The more general setting in [12], allowing for unbounded sets X, requires additional conditions, such as solvability of the h-problems in (2.4), which are automatic in our case, because X is bounded.

2.2. Primal and dual views. In (2.2), the difficult set $\mathcal{Z}$ is not available explicitly. Solution approaches adopting a primal view, for instance the method called primal-dual in [10], employ at the kth iteration a Dantzig-Wolfe like approximation. More precisely, given an index set $B^k=\{1,\dots,k\}$ keeping track of information generated so far, i.e., of $v^1,v^2,\dots,v^k$, consider the unit simplex associated with B^k:

$$\Delta^{|B^k|}:=\left\{\alpha\in\mathbb{R}^{|B^k|}_+:\ \text{with}\ \sum_j\alpha_j=1\right\}.$$

A new p-efficient point v^{k+1} is generated by using the optimal multiplier of the inequality constraint in

$$\left\{\begin{array}{ll}\min & c(x)\\ \text{s.t.} & g(x)\geq v\\ & x\in X\text{ and }v\in V_k:=\left\{\sum_{j\in B^k}\alpha_j v^j:\alpha\in\Delta^{|B^k|}\right\}.\end{array}\right. \tag{2.6}$$

Since our focus is on dual variants, we do not enter into further details, refering instead to [10, 13] .

Dual methods such as [12] relax the constraint $g(x)\geq v$ in (2.2) and maximize the dual function $h+d$ from (2.4). Computing the h-term amounts to solving a convex problem (X is a simple set), the difficulty lies in the calculation of $d(u)$ (involving the set $\mathcal{V}$ or its convex hull, by (2.5)). Even for distributions with finite support, a challenging mixed-integer linear programming (MILP) problem needs to be solved; see [23, 27]. This confirms the relevance of designing algorithms that can deal with approximate p-efficient points, as in Sections 4 and 5 below.

Suppose for the moment we can solve exactly both the h- and d-problems in (2.4), (2.5). At a given point $u=u^j$, by definition of these two concave nonsmooth functions (see for instance [12, Sec. 3]),

$$s^j_h:=-g(x^j)\ \text{ satisfies }\ -s^j_h\in\partial(-h)(u^j),\ \text{ for } x^j\in X \text{ minimizing } c(x)-\left\langle u^j,g(x)\right\rangle \tag{2.7}$$

$$s^j_d:=v^j\qquad\text{satisfies }\ -s^j_d\in\partial(-d)(u^j),\ \text{ for } v^j\in\operatorname{conv}\mathcal{V}\text{ minimizing }\left\langle u^j,v\right\rangle, \tag{2.8}$$

$$s^j:=s^j_h+s^j_d\ \text{ satisfies }\ -s^j\in\partial(-\varphi)(u^j).$$

Recall that, without loss of generality, the minimizer v^j in (2.8) can be taken in $\mathcal{V}$, $\operatorname{conv}\mathcal{V}$, or $\mathcal{Z}$, by (2.5).

2.3. A bundle method detour. The dual problem (2.4) has a concave objective function which is nonsmooth at those points u^j having more than one x^j or v^j solving (2.7) or (2.8), respectively. If φ was easy to compute (in (2.5) the set $\mathcal{V}$ complicates this calculation), a *proximal algorithm* [29,37] could be employed. At the kth-iteration and having a proximal stepsize $t_k>0$, this method defines iterates as follows:

$$u^{k+1}:=\arg\max_{u\in\mathbb{R}^m_+}\left\{\varphi(u)-\frac{1}{2t_k}\left|u-u^k\right|^2\right\}.$$

In our case, one iterate is as hard to compute as solving problem (2.4). *Bundle methods* [5, Part II] replace the difficult function φ by a simpler *model* M^k, to be improved along iterations. With the exact function, the next iterate u^{k+1} always provides ascent, but now (depending on the model quality), there is no guarantee that $\varphi(u^{k+1})$ will be larger than $\varphi(u^k)$. Bundle methods separate iterates providing sufficient ascent into a special *center subsequence* $\{\hat{u}^k\}$. The limit points of this subsequence, also called sequence of *serious steps*, solve (2.4).

A bundle algorithm of the proximal type defines iterates as below:

$$u^{k+1} = \arg\max_{u\in\mathbb{R}^m_+} \left\{ M^k(u) - \frac{1}{2t_k}\left|u-\hat{u}^k\right|^2 \right\}. \tag{2.9}$$

A rule for deciding when there is a serious step and the iterate becomes the new center $\hat{u}^{k+1}$, are given in Section 3. We just recall here the optimality conditions characterizing u^{k+1}, the unique solution to the concave maximization problem (2.9). These are classical relations in bundle methods; we refer to [21, LemXV.3.1.1] for a proof:

$$u^{k+1} = \hat{u}^k + t_k\hat{s}^k \qquad \text{with } \hat{s}^k := p_1^k - p_2^k \text{ for } \begin{cases} -p_1^k \in \partial(-M^k)(u^{k+1}) \\ p_2^k \in N_{\mathbb{R}^m_+}(u^{k+1}). \end{cases} \tag{2.10}$$

The computational work involved in solving (2.9) depends on the specific model M^k. The general theory for bundle methods in [32] is very flexible in this sense, many different models can be used, below we provide three different choices that are acceptable for the dual function φ in (2.4).

Example 2.1 (Aggregate cutting-plane model: $M^k = \check{\varphi}^k$). Perhaps the most natural choice is to replace the concave function φ by an outer approximation defined by linearizations, or cutting planes. This is the function

$$\check{\varphi}^k(u) := \min_{j\in B^k} L^j(u) \quad \text{where } L^j(u) := c(x^j) + \left\langle u^j, v^j - g(x^j)\right\rangle \quad \begin{array}{l}\text{for } x^j \in X \text{ as in (2.7)} \\ \text{and } v^j \in \operatorname{conv}\mathcal{V} \text{ as in (2.8).}\end{array} \tag{2.11}$$

The index-set B^k gathers past linearization indices until iteration k; for instance all of them: $B^k = \{1,\dots,k\}$ (in Section 3.2 more economical sets B^k, collecting less linearizations, are considered).

Taking the model $M^k = \check{\varphi}^k$ gives in (2.9) a convex quadratic programming problem (QP), easy to solve. Being a piecewise affine convex function, the model subgradients at u^{k+1} are simplicial combinations of gradients of active pieces. In particular, the subgradient in (2.10) is given by

$$p_1^k = \sum_{j\in B^k} \alpha_j^{k+1}(v^j - g(x^j)) \text{ with } \alpha^{k+1} \in \Delta^{|B^k|} \text{ such that } M^k(u^{k+1}) = \sum_{j\in B^k} \alpha_j^{k+1} c(x^j) + \left\langle p_1^k, u^{k+1}\right\rangle. \tag{2.12}$$

In the bundle set B^k, *strongly active* indices correspond to active linearizations with positive weight:

$$j \in B^k \text{ such that } \alpha_j^{k+1} > 0 \text{ and } M^k(u^{k+1}) = L^j(u^{k+1}). \tag{2.13}$$

For asymptotic analysis reasons, we assume that the number of strongly active indices is uniformly bounded in k. This natural property is ensured for instance by most active-set QP solvers, whose linearly independent bases involve at most $m+1$ Carathéodory-like positive simplicial multipliers, regardless of the cardinality of B^k.

The aggregate cutting-plane model is used in the *Regularized Dual Method* described in [12, Sec. 4], taking a constant stepsize $t_k = t > 0$ and a full bundle of information $B^k = \{1,2,\dots,k\}$. □

The following model takes advantage of the sum-structure of the function φ.

Example 2.2 (Disaggregate cutting-plane model: $M^k = \check{h}^k + \check{d}^k$). Each term has its own linearization

$$L_h^j(u) := c(x^j) + \left\langle -g(x^j), u\right\rangle \quad \text{and} \quad L_d^j(u) := \left\langle v^j, u\right\rangle.$$

The disaggregate model is the sum of the individual cutting-plane models, using separate index sets, B^k and $\tilde{B}^k$:

$$\check{h}^k(u) + \check{d}^k(u) \;:=\; \min_{j\in B^k} L_h^j(u) \;+\; \min_{j\in \tilde{B}^k} L_d^j(u).$$

Taking the disaggregate model gives again a convex QP subproblem (2.9), of larger size than the aggregate one, to account separately for the two bundles of information. In particular, the subgradient of this model at u^{k+1} is

$$p_1^k = \sum_{j\in B^k} \alpha_j^{k+1} v^j - \sum_{j\in \tilde{B}^k} \tilde{\alpha}_j^{k+1} g(x^j) \text{ with } \begin{array}{l}\alpha^{k+1}\in\Delta^{|B^k|}\\ \tilde{\alpha}^{k+1}\in\Delta^{|\tilde{B}^k|}\end{array} \text{ such that } M^k(u^{k+1}) = \sum_{j\in B^k}\alpha_j^{k+1} c(x^j) + \left\langle p_1^k, u^{k+1}\right\rangle , \tag{2.14}$$

and, hence, in this model there are strongly active indices (2.13) for each separate bundle. □

The third model takes the exact function h.

Example 2.3 (Partially exact model: $M^k = h + \check{d}^k$). Only the difficult function d is modeled by cutting planes:

$$h(u) + \check{d}^k(u) := h(u) + \min_{j\in\tilde{B}^k} L_d^j(u) = \min_{x\in X}\Big\{ c(x) - \langle g(x), u\rangle \Big\} + \min_{j\in\tilde{B}^k} \left\langle v^j, u\right\rangle .$$

The subgradient for this model is

$$p_1^k = \sum_{j\in\tilde{B}^k} \tilde{\alpha}_j^{k+1} v^j - g(x^{k+1}) \text{ with } \begin{array}{l} h(u^{k+1}) = c(x^{k+1}) + \left\langle u^{k+1}, g(x^{k+1})\right\rangle \\ \tilde{\alpha}^{k+1}\in\Delta^{|\tilde{B}^k|}\end{array} \tag{2.15}$$

and such that $M^k(u^{k+1}) = c(x^{k+1}) + \left\langle p_1^k, u^{k+1}\right\rangle$. Naturally, if neither f nor g are linear or quadratic functions, the model $M^k = h + \check{d}^k$ no longer yields a QP and (2.9) becomes a general convex optimization problem. □

Regarding the quality of the various models above, from their definition it is straightforward that for all $u\in\mathbb{R}^m_+$

$$\check{\varphi}^k(u) \geq \check{h}^k(u) + \check{d}^k(u) \geq h(u) + \check{d}^k(u) \geq \varphi(u)\,, \tag{2.16}$$

i.e., the partially exact model is better than the disaggregate model, in turn better than the aggregate one. On the other hand, subproblem (2.9) becomes easier for the choice $M^k = \check{\varphi}^k$, and more difficult for $M^k = h + \check{d}^k$. For this latter model, it will be shown in Section 3 that (2.9) with $M^k = h + \check{d}_k$ corresponds to the Progressive Augmented Lagrangian method [12], which relaxes the constraint $g(x)\geq v$ in (2.6); see Example 3.4 for details.

3. Link with bundle methods. To progress along iterations, most of the dual approaches identify a new p-efficient point by solving the d oracle in (2.5), at the current dual iterate. Since this calculation involves knowing the set $\mathcal{V}$, this is a computationally heavy task. It is then important to determine to which extent evaluating d approximately impacts on the convergence properties of the methods. For the analysis, we pursue our interpretation through a bundle-algoritm perspective, and rely on the general theory in [32].

In our development, evaluation errors are possible in the following sense:

$$\varphi_{u^j} \in [\varphi(u^j), \varphi(u^j) + \eta_{u^j}] \text{ approximates the exact value, for some error } \eta_{u^j}\geq 0\,. \tag{3.1}$$

As uniform boundedness of the inaccuracy is a condition for convergence, we suppose that

$$\text{for all } u^j\in\mathbb{R}^m_+ \text{ the inaccuracy } \eta_{u^j} \text{ in (4.1) satisfies } \eta_{u^j}\leq\eta \text{ for some } \eta\geq 0. \tag{3.2}$$

To introduce gradually the different features of an inexact algorithm, we start by considering only two situations:

- the oracle always delivers *exact* information ($\eta_{u^j}\equiv 0$), as in Examples 2.1 and 2.3; or
- the oracle delivers exact information only when there is a serious step, at points that become a center ($\eta_{u^j}>0$, except for $\eta_{\hat{u}^k}\equiv 0$), as in Example 3.3 below, derived from Example 2.2.

For all the oracles and models in this work, both in this section and in Section 4, the models satisfy the condition

$$\varphi(u)\leq M^k(u) \quad \text{for all } u\in\mathbb{R}^m_+\,. \tag{3.3}$$

This property will be referred to as having an *upper model* (in the parlance of [31], minimizing the convex function $-\varphi$, the model is of lower type). Similarly, we shall say L^j is an *upper* linearization when

$$L^j(\cdot) \text{ is an affine function such that } \varphi(u)\leq L^j(u) \quad \text{for all } u\in\mathbb{R}^m_+\,. \tag{3.4}$$

3.1. A proximal bundle method for concave maximization. The solution of (2.9) with a model M^k satisfying (3.3) gives u^{k+1}. Given a parameter $m \in (0,1)$, *a serious step* is declared and $\hat{u}^{k+1} := u^{k+1}$, when

$$\varphi_{u^{k+1}} \geq \varphi_{\hat{u}^k} + m\,\delta^k \;\text{ for } \delta^k := M^k(u^{k+1}) - \varphi_{\hat{u}^k}, \text{ where } \varphi_{\hat{u}^k} = \varphi(\hat{u}^k) \text{ because } \eta_{\hat{u}^k} = 0. \tag{3.5}$$

Otherwise, the iteration is declared *null*, keeping $\hat{u}^{k+1} := \hat{u}^k$.

The subgradient inequality for p_1^k in (2.10), combined with the definitions of $\hat{s}^k$ and p_2^k, ensures that $M^k(u^{k+1}) \geq M^k(\hat{u}^k) + t_k|\hat{s}^k|^2$. Then, (3.4) implies that (3.5) is an ascent test, checking increase in the function values:

$$\delta^k \geq M^k(\hat{u}^k) + t_k|\hat{s}^k|^2 - \varphi_{\hat{u}^k} \geq t_k|\hat{s}^k|^2 - \eta_{\hat{u}^k} = t_k|\hat{s}^k|^2 \geq 0 \quad \text{if } \eta_{\hat{u}^k} = \varphi_{\hat{u}^k} - \varphi(\hat{u}^k) = 0. \tag{3.6}$$

For future reference, note that the relations in (3.5) and (3.6) hold because the evaluation error is null at $\hat{u}^k$.

The *aggregate linearization*

$$A^k(u) := M^k(u^{k+1}) + \left\langle p_1^k, u - u^{k+1} \right\rangle \tag{3.7}$$

available after solving (2.9), is of the upper type (due to the definitions in (2.10)):

$$\varphi(u) \leq A^k(u) \;\text{ for all }\; u \in \mathbb{R}^m_+ . \tag{3.8}$$

As shown in [21, LemXV.3.1.2], the aggregate linearization is the highest outer approximation of φ that can be used without losing information (replacing M^k by A^k in (2.9) maintains the solution u^{k+1}). The aggregate linearization A^k condenses all the past information generated by the method and is the key behind the mechanism called *bundle compression* described in Example 3.2 below; see also [5, Ch. 10.3.2].

By combining the rightmost identities in (2.12)-(2.15) with (3.7) written for $u = 0$, we see that

$$A^k(0) = \begin{cases} \displaystyle\sum_{j \in B^k} \alpha_j^{k+1} c(x^j) & \text{with the aggregate and disaggregate models (Examples 2.1 and 2.2)} \\ c(x^{k+1}) & \text{with the partially exact model (Example 2.3).} \end{cases} \tag{3.9}$$

These relations are fundamental to show primal convergence. For dual convergence, two important objects, computable after solving (2.9), are the *aggregate gap* and the *Fenchel measure*, defined respectively by

$$\hat{e}_k := \delta^k - t_k|\hat{s}^k|^2 \;\text{ and }\; \phi_k := \hat{e}_k - \left\langle \hat{s}^k, \hat{u}^k \right\rangle . \tag{3.10}$$

As in (3.6), having upper models with exact evaluations at serious steps implies that the gap is always nonnegative:

$$\hat{e}_k \geq -\eta_{\hat{u}^k} = 0, \quad \text{if } \eta_{\hat{u}^k} = 0. \tag{3.11}$$

We now state some results highlighting the role of these objects regarding convergence.

LEMMA 3.1 (Primal and dual optimality certificates). *Suppose the oracle evaluations and the model satisfy, respectively,* (3.1) *and* (3.3)*. Associated with the model in* (2.9) *consider the primal pair*

$$(\hat{x}^{k+1}, \hat{v}^{k+1}) := \begin{cases} \sum_{j \in B^k} \alpha_j^{k+1} \left(x^j, v^j\right) & \textit{generated with the aggregate model (Ex. 2.1)} \\ \left\langle \sum_{j \in B^k} \alpha_j^{k+1} x^j, \sum_{j \in \tilde{B}^k} \tilde{\alpha}_j^{k+1} v^j \right\rangle & \textit{generated with the disaggregate model (Ex. 2.2)} \\ \left(x^{k+1}, \sum_{j \in \tilde{B}^k} \tilde{\alpha}_j^{k+1} v^j\right) & \textit{generated with the partially exact model (Ex. 2.3).} \end{cases} \tag{3.12}$$

The following holds:

(i) $\varphi(u) \leq \varphi(\hat{u}^k) + \eta_{\hat{u}^k} + \phi^k + \left\langle \hat{s}^k, u \right\rangle$ *for all* $u \in \mathbb{R}^m_+$.

(ii) The primal pair satisfies $(\hat{x}^{k+1}, \hat{v}^{k+1}) \in X \times \operatorname{conv} \mathcal{V} \subset X \times \mathcal{Z}$*, with*

$$\hat{v}^{k+1} \leq g(\hat{x}^{k+1}) + \hat{s}^k \quad \textit{and} \quad c(\hat{x}^{k+1}) \leq \varphi(\hat{u}^k) + \eta_{\hat{u}^k} + \phi^k .$$

(iii) If $\hat{s}^k = 0$ *and* $\phi^k \leq 0$ *then* $\hat{u}^k$ *is an* $\eta_{\hat{u}^k}$*-solution to* (2.4) *and* $(\hat{x}^{k+1}, \hat{v}^{k+1})$ *is an* $\eta_{\hat{u}^k}$*-solution to* (2.2)*.*

Proof. To show (i), we first prove that

$$\varphi(u) \le A^k(0) + \left\langle \hat{s}^k, u \right\rangle . \tag{3.13}$$

Write the subgradient inequality for p_1^k in (2.10) and use (3.7) to see that

$$-M^k(u) \ge -M^k(u^{k+1}) - \left\langle p_1^k, u - u^{k+1} \right\rangle = -A^k(u) .$$

Since the aggregate linearization is affine, $A^k(u) = A^k(0) + \langle p_1^k, u \rangle$, and the definition for $\hat{s}^k$ in (2.10) gives

$$M^k(u) \le A^k(0) + \left\langle \hat{s}^k, u \right\rangle + \left\langle p_2^k, u \right\rangle .$$

The rightmost term is nonpositive because $p_2^k \in N_{\mathbb{R}^m_+}(u^{k+1})$ and, by the normal cone definition, for all $u \in \mathbb{R}^m_+$,

$$\left\langle p_2^k, u \right\rangle \le \left\langle p_2^k, u^{k+1} \right\rangle = 0 . \tag{3.14}$$

By (3.3), the expression in (3.13) must hold. Writing (3.1) with $u^j = \hat{u}^k$ gives that $\varphi_{\hat{u}^k} \le \varphi(\hat{u}^k) + \eta_{\hat{u}^k}$, so if

$$A^k(0) = \varphi_{\hat{u}^k} + \phi^k , \tag{3.15}$$

item (i) will be proven. To show this identity, start with the right hand side, combining (3.10) and (3.5):

$$\varphi_{\hat{u}^k} + \phi^k = \varphi_{\hat{u}^k} + \delta^k - t_k|\hat{s}^k|^2 - \left\langle \hat{s}^k, \hat{u}^k \right\rangle = M^k(u^{k+1}) - \left\langle \hat{s}^k, u^{k+1} \right\rangle , \text{ by (2.10).}$$

Then (3.15) follows from evaluating (3.7) at $u = 0$, because $\langle \hat{s}^k, u^{k+1} \rangle = \langle p_1^k, u^{k+1} \rangle$ by (3.14).
To show item (ii), recall from Section 2.1 that $\operatorname{conv} \mathcal{V} \subset \mathcal{Z}$, so by convexity of X, the primal pair is in $X \times \mathcal{Z}$, as stated. As the normal element in (2.10) satisfies $p_2^k = p_1^k - \hat{s}^k$, writing (3.14) with $u = \hat{u}^k$ yields $\langle p_1^k - \hat{s}^k, \hat{u}^k \rangle \le 0$. Using the concavity of g and the expressions for p_1^k in (2.12)-(2.15), gives $\langle \hat{v}^{k+1} - g(\hat{x}^{k+1}) - \hat{s}^k, \hat{u}^k \rangle \le 0$, and the left relation in item (ii) follows, because $\hat{u}^k \ge 0$. To prove the right statement in item (ii), use (3.9) and the convexity of c in (3.15) to write $\phi^k \ge c(\hat{x}^{k+1}) - \varphi_{\hat{u}^k}$, which together with (3.1) written with $u^j = \hat{u}^k$, gives the desired inequality. Finally, dual approximate optimality is straightforward using that $\hat{s}^k = 0$ and $\phi^k \le 0$ in item (i). Regarding the primal pair, item (ii) with $\hat{s}^k = 0$ ensures feasibility in (2.2). The same item (ii) with $\phi^k \le 0$ gives that $c(\hat{x}^{k+1}) \le \varphi(\hat{u}^k) + \eta_{\hat{u}^k}$. By the dual function definition in (2.4), $\varphi(\hat{u}^k) \le c(x) + \langle \hat{u}^k, v - g(x) \rangle \le c(x)$ for any (x, v) satisfying $g(x) \ge v$, and the result follows. □

The relations in item (ii) reveal that the aggregate gradient and the Fenchel measure respectively estimate primal feasibility and the duality gap; the algorithm stops when those values are sufficiently small. Theorem 3.2 shows that (3.16), an asymptotic version of the conditions in item (iii), guarantees eventual solution of (2.4) and (2.2).

Algorithm 1 Concave Proximal Bundle Method for Upper Models and Exact Serious Evaluations (PBM)

Step 0: initialization. Select $m \in (0,1)$ and $t_1 \ge t_{\texttt{low}} > 0$.
Choose $u^1 \in \mathbb{R}^m_+$ and call the oracles to compute $h(u^1)$, $d(u^1)$ and respective subgradients (2.7) and (2.8).
Choose a model $M^1 \ge \varphi$, stopping tolerances, tol_ϕ, $\text{tol}_{\hat{s}} \ge 0$ and set $\hat{u}^1 = u^1$, $k = 1$.

Step 1: Next iterate. Obtain u^{k+1} by solving (2.9).
Compute δ^k as in (3.5), $\hat{s}^k = \frac{u^{k+1} - \hat{u}^k}{t_k}$ from (2.10), and ϕ_k from (3.10).

Step 2: Stopping test. If $\phi_k \le \text{tol}_\phi$ and $|\hat{s}^k| \le \text{tol}_{\hat{s}}$, stop and return $\hat{u}^k$ and $(\hat{x}^{k+1}, \hat{v}^{k+1})$ from (3.12) as the solution.

Step 3: Oracle call. Compute an upper linearization L^{k+1} as in (3.4), for example using the exact values $h(u^{k+1})$, $d(u^{k+1})$ and respective subgradients (2.7) and (2.8).

Step 4: Ascent test. If (3.5) is satisfied (serious step), make new calculations if necessary to ensure $\eta_{u^{k+1}} = 0$.
Set $\hat{u}^{k+1} = u^{k+1}$ and choose $t_k \ge t_{\texttt{low}}$.
If (3.5) does not hold (null step), set $\hat{u}^{k+1} = \hat{u}^k$ and choose $t_{k+1} \in [t_{\texttt{low}}, t_k]$.

Step 5: Model. Choose a model satisfying $\varphi \le M^{k+1} \le \min\{L^{k+1}, A^k\}$.

Step 6: Loop. Set $k = k + 1$ and go back to Step 1.

Algorithm 1 describes the Proximal Bundle Method (PBM), when the model is of upper type and evaluation errors are null at serious steps: both (3.3) and (3.1) with $\eta_{\hat{u}^k} \equiv 0$ hold.

The need of new calculations mentioned in Step 4 typically arises when the solution procedure in the oracle is organized so that first a coarse estimation is delivered, to check ascent. If the test declares a null step, the algorithm proceeds. Otherwise, the algorithm returns to the oracle requesting an exact calculation; see Steps 3' and 4' in Example 3.3, and also the "coarse" and "fine" phases implemented for the on-demand oracles in Section 6.

The order of steps in Algorithm 1 is the usual one in bundle methods: first the new dual iterate is found in Step 1 and only in Step 3 the oracle finds corresponding primal points (when solving the h and d problems in (2.4), (2.5)). Example 3.4 below illustrates a variant that switches this dual-primal order, defining first primal iterates.

When PBM loops forever, there are two cases: either there is an infinite tail of null steps, or the algorithm generates infinitely many serious steps. These are the two mutually exclusive cases considered for the set K^∞ in Theorem 3.2, noting that always at least one of them has infinite cardinality when $k \to \infty$. Convergence for PBM is shown below by we applying the general framework in [32].

THEOREM 3.2 (Primal and dual convergence for PBM). *Consider a primal problem* (2.2) *and its dual problem* (2.4) *such that the assumptions in Section 2.1 hold. Suppose that in Algorithm 1 the model satisfies* (3.3) *and the stopping tolerances are taken null (*$\mathrm{tol}_\phi = \mathrm{tol}_{\hat{s}} = 0$*). If the oracle satisfies* (3.1) *and* (3.2) *and at serious steps there is no error evaluation (*$\eta_{\hat{u}^k} = 0$*) then*

$$\limsup_{k\in K^\infty} \phi_k \le 0 \ \textit{and} \ \lim_{k\in K^\infty} \hat{s}^k = 0, \tag{3.16}$$

for an infinite iteration-set defined by

$$\begin{array}{lrcl} \textit{either} & K^\infty & := & \{k \ge \hat{k}\} \textit{ if after a last serious step at iteration } \hat{k} \text{ (3.5) } \textit{always fails} \\ \textit{or} & K^\infty & := & \{k : u^{k+1} \textit{ is declared serious in Step 4 }\}, \textit{ otherwise}. \end{array}$$

It follows that the primal subsequence $\{(\hat{x}^{k+1}, \hat{v}^{k+1})\}$ *always has limit points, and any of them solves* (2.2). *When the center subsequence* $\{\hat{u}^k\}$ *has limit points, any of them solves* (2.4).

Proof. The assumption that $\eta_{\hat{u}^k}$ is null in (3.1) implies that $\varphi_{\hat{u}^k} = \varphi(\hat{u}^k)$. If the algorithm stops at some iteration k, then $\phi_k \le \left|\hat{s}^k\right| = 0$ and the corresponding primal and dual iterates are optimal, by Lemma 3.1(iii). If the algorithm loops forever, consider the infinite subsequences associated with K^∞. Existence of limit points for $\{\hat{x}^{k+1}\}$ results from boundeness of X and the assumption that when solving (2.9) the number of active multipliers from (2.13) is kept bounded (Carathéodory theorem). Therefore, when (3.16) holds and $\hat{s}^k \to 0$, the left inequality in Lemma 3.1(ii) implies that the sequence $\{\hat{v}^{k+1}\}$ is bounded above. Since the set $\mathcal{V}$ is bounded below (see Section 2.1), the sequence $\{\hat{v}^{k+1}\} \subset \mathrm{conv}\,\mathcal{V}$ is contained in a compact set and, hence, has limits points too.

To show (3.16) we apply Propositions 6.1 and 6.7 in [32]. Table 3.1 relates our notation with the one in that work, developed for problems minimizing a convex function $f = -\varphi$:

TABLE 3.1
Relation with notation in [32]

throughout in [32]	$f = -\varphi$, $f_k^M = -M^k$, $f_{-k}^L = -A^k$, $\alpha_k = 0$;
in [32, Eq. (4.5)]	$\delta_k^M = \delta^k$ and $\hat{g}^k = -\hat{s}^k$;
in [32, Eq. (4.4)]	$\ell_k = -\varphi_{\hat{u}^k}$ and $f_{-k}^L(\hat{u}^k) = -A(\hat{u}^k)$, so $\hat{e}_k$ in [32] coincides with $\hat{e}_k$ in (3.10);
in [32, Eq. (4.8)]	ϕ_k in [32] coincides with ϕ_k in (3.10);
in [32, Eq. (4.11)]	$\eta^M \equiv 0$;
in [32, Eq. (6.11)]	$\delta_k^E = \delta^k$; and $\ell_k - f_{u_{k+1}} - \delta_k^E = -\varphi_{\hat{u}^k} + \varphi_{u^{k+1}} - \delta^k \le (m-1)\delta^k \le 0$ at null steps.

By Proposition 6.1 in [32], (3.16) holds if $t_k \ge t_{\mathtt{low}}$ and $\delta^k \to 0$ as $K^\infty \ni k \to \infty$. The first condition is satisfied by the stepsize updates in Step 4. For the second condition, consider first the case when K^∞ corresponds to the tail of null steps. Proposition 6.7 in [32] states that $\delta^k \to 0$ if condition (6.11) therein holds. This condition is trivially satisfied, because of the relations in the last line in Table 3.1. In the second case for K^∞, we have that

$$0 \le \sum_{k\in K^\infty} \delta^k = \sum_{k\in K^\infty} \Big(\varphi(\hat{u}^{k+1}) - \varphi(\hat{u}^k)\Big) \le c^* - \varphi(u^1) < +\infty.$$

So (3.16) holds and, as $\eta_{\hat{u}^k} \equiv 0$, passing to the limit in Lemma 3.1(ii) as $K^\infty \ni k \to \infty$ concludes the proof. □

Remark 3.1 (Partially inexact variants). In the proof of Theorem 3.2, rather than exactness of the oracle at each serious step ($\eta_{\hat{u}^k} \equiv 0$), what matters is to have *vanishing* evaluation errors at those points: $\eta_{\hat{u}^k} \to 0$ as $K^\infty \ni k \to \infty$. This means in particular that, if the oracle delivers exact information at serious steps only eventually, convergence for the case when K^∞ comes from an infinite sequence of serious steps is *maintained* (as long as the important property (3.3) is preserved for the model). Bundle methods working with models combining exact information at serious steps with inexact information at null steps are called *partially inexact*; [16]. For an illustration of such a variant, see the Example 3.3 below, and its implementation BM4 in Section 6, with the numerical experience.□

3.2. Relating PBM with some dual methods. We now explain how different choices for the model fit the update in Step 5 of Algorithm 1. We consider the three choices in Examples 2.1, 2.2 and 2.3 and make the link with some dual methods based on p-efficient points.

Example 3.2 (PBM with aggregate cutting-plane model and exact evaluations). When M^k is the aggregate model in Example 2.1, and the oracle information is exact, Algorithm 1 is a standard proximal bundle method, with linearizations of the form

$$L^j(u) = \varphi(u^j) + \langle s^j, u - u^j \rangle = c(x^j) + \langle v^j - g(x^j), u \rangle \qquad \begin{array}{l} \text{for } x^j \in X \text{ as in (2.7)} \\ \text{and } v^j \in \operatorname{conv} \mathcal{V} \text{ as in (2.8).} \end{array} .$$

The regularized dual algorithm of [12] is a particular case of this variant, which maintains t_k fixed along iterations and sets in (2.11) the full index set, $B^k = \{1, \dots, k\}$. The corresponding model update is

$$M^{k+1} = \min\{L^{k+1}, M^k\},$$

which by (3.7) satisfies the conditions in Step 5. A difficulty with this update is that the size of the QPs (2.9) increases at each iteration. To keep the QP size controlled, the bundle can be reduced by introducing either a *selection* or a *compression* mechanism. The latter amounts to taking

$$M^{k+1} = \min\{L^{k+1}, A^k\},$$

(similarly to the "generalized Frank-Wolfe rule" in [38, Eq. (3.31)]). This very economic model satisfies (3.3) and results in a QP with just two constraints, so each bundle iteration is fast, but many iterations may be needed to converge. By contrast, the selection mechanism keeps in the new model only *active* linearizations, as in (2.12):

$$M^{k+1} = \min\left\{L^{k+1}, \min\{L^j : j \in B^k \text{ such that } L^j(u^{k+1}) = M^k(u^{k+1})\}\right\}.$$

When compared with the compression technique, selection yields a less economic QP, but in general the additional time spent in solving (2.9) is compensated by a smaller number of iterations.

In view of Theorem 3.2, the regularized dual algorithm of [12] maintains its convergence properties for varying stepsizes satisfying $t_{\texttt{low}} \le t_k$ and with smaller QP subproblems, two enhancements likely to significantly improve the convergence speed of the method. □

The aggregate model above uses exact information: in (3.1) the error η_{u^j} is always null. If the model M^k is chosen to be the disaggregate cutting-plane model, it is possible to avoid computing the expensive d-information at null steps. We now explain how to implement this saving *without* impairing the convergence results in Theorem 3.2.

Example 3.3 (Disaggregate partially inexact model). In Step 3, to provide an approximate value $\varphi_{u^{k+1}}$ satisfying (3.1), the oracle takes the exact value for $h(u^{k+1})$ and uses the cutting-plane value $\check{d}^k(u^{k+1})$ to replace the d-value. The methods called BM2 and BM3 in the numerical Section 6 implement this variant.

Calculations are organized by modifying Steps 3 and 4 in Algorithm 1 as follows.

Step 3': First oracle call. Compute $h(u^{k+1})$ and its subgradient (2.7).
For the linearization L_d^{k+1} take the approximation $(d_{u^{k+1}}, s_d^{k+1}) := (\check{d}^k(u^{k+1}), \hat{v}^{k+1})$, available from (2.14).

Step 4': Ascent test and possible second oracle call. If the inequality

$$h(u^{k+1}) + d_{u^{k+1}} \ge \varphi(\hat{u}^k) + m\,\delta^k \tag{3.17}$$

is satisfied (serious step), recalculate L_d^{k+1} by computing the exact d-information from (2.8). Set $\hat{u}^{k+1} = u^{k+1}$ and choose $t_k \ge t_{\texttt{low}}$.
Otherwise (null step), set $\hat{u}^{k+1} = \hat{u}^k$ and choose $t_{k+1} \in [t_{\texttt{low}}, t_k]$.

The inequality $\check{d}^k(u) \geq d(u)$ is always satisfied ($\check{d}^k$ is an upper model); so when (3.17) fails the ascent test (3.5) cannot hold and the iteration is declared null. The replacements Step 3' and Step 4' prevent the algorithm from computing the expensive d-information at null points. When this happens, the model update in Step 5 can simply take $M^{k+1} = \check{h}^{k+1} + \check{d}^k$. Otherwise, if the iterate was declared serious, $\eta_{u^{k+1}} = 0$ and the new exact linearization L_d^{k+1} enters the d-model: $M^{k+1} = \check{h}^{k+1} + \check{d}^{k+1}$. In both cases, the model satisfies (3.3) and, since (3.1) holds with $\eta_{\hat{u}^k} \equiv 0$, Theorem 3.2 applies.

For the disaggregate model it is also possible to put in place a selection/compression mechanism, proceeding separately for each term. The optimality conditions (2.10) for this specific model give an aggregate linearization that can be split into two functions, say A_h^k and A_d^k. Then Step 5 can take any cutting-plane model satisfying

$$M^{k+1} = \check{h}^{k+1} + \check{d}^{k+1} \text{ with } h \leq \check{h}^{k+1} \leq \min\left\{L_h^{k+1}, A_h^k\right\} \text{ and } d \leq \check{d}^{k+1} \leq \min\left\{L_d^{k+1}, A_d^k\right\}. \square$$

The next primal-dual form of Algorithm 1 is the *Progressive Augmented Lagrangian*. introduced in [12].

Example 3.4 (PBM with partially exact model). We now focus on the particular case in which the model M^k is

$$M^k(u) = h(u) + \check{d}^k(u) = \min_{x \in X}\left\{c(x) - \langle g(x), u\rangle\right\} + \min_{j \in \tilde{B}^k}\left\langle v^j, u\right\rangle,$$

given in Example 2.3. Following the development in [12] we now make the relation between PBM and the augmented Lagrangian method for (2.2); see also [24].

We start by rewriting the cutting-plane model $\check{d}^k$ as an optimization problem:

$$\check{d}^k(u) = \min_{j \in \tilde{B}^k}\left\langle v^j, u\right\rangle = \left\{\begin{array}{ll} \min_v & \langle v, u\rangle \\ & v \in V_k \text{ from (2.6)} \end{array}\right. = \left\{\begin{array}{ll} \min & \left\langle \sum_{j\in\tilde{B}^k} \alpha_j v^j, u\right\rangle \\ \text{s.t.} & \alpha \in \Delta^{|\tilde{B}^k|}. \end{array}\right.$$

This notation is useful to rewrite the partially exact model M^k in the form

$$M^k(u) = \min_{x\in X, \alpha\in\Delta^{|\tilde{B}^k|}}\left\{c(x) - \langle g(x), u\rangle + \left\langle \sum_{j\in\tilde{B}^k}\alpha_j v^j, u\right\rangle\right\},$$

showing that the model is in fact the dual function associated with problem (2.6). More precisely, letting u denote the multiplier associated with the constraint $g(x) \geq v$, we see that

$$M^k(u) = \min_{x\in X, \alpha\in\Delta^{|\tilde{B}^k|}} L(x,\alpha;u) \quad \text{for } L(x,\alpha;u) := c(x) + \left\langle u, \sum_{j\in\tilde{B}^k}\alpha_j v^j - g(x)\right\rangle.$$

Therefore, the (negative of the concave) model has subgradients of the form $g(x) - \sum_j \alpha_j v^j$ for any pair (x,α) solving the minimization above. In particular,

$$M^k(u^{k+1}) = L(x^{k+1}, \alpha^{k+1}; u^{k+1}) = \min_{x\in X, \alpha\in\Delta^{|\tilde{B}^k|}} L(x,\alpha;u^{k+1}) \text{ for } (x^{k+1},\alpha^{k+1}) \text{ from (2.15).}$$

Regarding Algorithm 1, this model gives for (2.9) in Step 1 the concave subproblem

$$u^{k+1} \text{ solves } \max_{u\geq 0}\left\{M^k(u) - \frac{t_k}{2}|u-\hat{u}^k|^2\right\} \equiv \max_{u\geq 0}\min_{x\in X,\alpha\in\Delta^{|\tilde{B}^k|}}\left\{L(x,\alpha;u) - \frac{t_k}{2}|u-\hat{u}^k|^2\right\}.$$

The argument in the right hand side formulation is the *regularized Lagrangian* from [12, Eq. (25)]. By strict concavity with respect to u and compactness of X, the triplet $(x^{k+1},\alpha^{k+1},u^{k+1})$ is a saddle point for the regularized Lagrangian and, in particular,

$$\max_{u\geq 0}\min_{x\in X,\alpha\in\Delta^{|\tilde{B}^k|}}\left\{L(x,\alpha;u) - \frac{t_k}{2}|u-\hat{u}^k|^2\right\} = \min_{x\in X,\alpha\in\Delta^{|\tilde{B}^k|}} L(x,\alpha;u^{k+1}) - \frac{t_k}{2}|u^{k+1}-\hat{u}^k|^2.$$

The expression of p_1^k from (2.15) together with the normal element definition for p_2^k gives in (2.10) that

$$u^{k+1} := \max\left(0,\, \hat{u}^k + t_k\Big(\sum_{j\in\tilde{B}^k} \alpha_j^{k+1} v^j - g(x^{k+1})\Big)\right) \quad \text{for } (x^{k+1},\alpha^{k+1}) \text{ from (2.15)}, \tag{3.18}$$

which highlights the primal-dual feature of this variant: to make the dual update, the primal points x^{k+1} and $v^{k+1} = \sum_{j\in\tilde{B}^k} \alpha_j^{k+1} v^j$ need to be available.

The Augmented Lagrangian perspective from [12] reveals that the primal points can be computed by solving, either the dual problem (2.9) written with the partially exact model, or a problem on primal variables, involving the Augmented Lagrangian associated with (2.6). More precisely, consider

$$L_k(x,\alpha;u) := c(x) + \langle u, G(x,\alpha;u)\rangle + \frac{t_k}{2}\,|G(x,\alpha;u)|^2$$

where we defined

$$G_i(x,\alpha;u) := \max\left(-\frac{u_i}{t_k},\, \sum_{j\in\tilde{B}^k} \alpha_j v_i^j - g_i(x)\right) \quad \text{for } i = 1,\dots,m.$$

Taking the derivatives and using the definition above, it is easy to see that

$$\frac{\partial L_k(x,\alpha;u)}{\partial(x,\alpha)} = \frac{\partial L(x,\alpha;u+t_kG(x,\alpha;u))}{\partial(x,\alpha)}\,.$$

Since in addition

$$u + t_kG(x,\alpha;u) = \max\left(0,\, u + t_k\Big(\sum_{j\in\tilde{B}^k} \alpha_j v^j - g(x)\Big)\right),$$

together with (3.18) we see that

$$\frac{\partial L_k(x^{k+1},\alpha^{k+1};\hat{u}^k)}{\partial(x,\alpha)} = \frac{\partial L(x^{k+1},\alpha^{k+1};u^{k+1})}{\partial(x,\alpha)}\,.$$

This means that

$$\text{the pair } (x^{k+1},\alpha^{k+1}) \text{ solves } \min_{x\in X,\alpha\in\Delta^{|\tilde{B}^k|}} L_k(x,\alpha;\hat{u}^k) = \min_{x\in X,\alpha\in\Delta^{|\tilde{B}^k|}} L(x,\alpha;u^{k+1})\,,$$

and, hence, solving the left hand side problem above gives the desired primal points, to be used in (3.18) to make the dual update. Accordingly, Algorithm 1 with the partially exact model can be enhanced as follows:

Step 1': Next primal and dual iterates. Obtain (x^{k+1},α^{k+1}) by solving $\min_{x\in X,\alpha\in\Delta^{|\tilde{B}^k|}} L_k(x,\alpha;\hat{u}^k)$,
and compute u^{k+1} as in (3.18).
Compute δ^k as in (3.5), $\hat{s}^k = \frac{u^{k+1}-\hat{u}^k}{t_k}$ from (2.10), and ϕ_k as in (3.10).

Step 3': Oracle call of d. Compute the linearization L_d^{k+1} using the exact information from (2.8).

In Step 3', the oracle only delivers information on d because the model $M^k = h + \check{d}^k$ has no need of linearizations for h. In view of (2.15), once the primal point x^{k+1} is available in Step 1', both the exact function value and a subgradient for h are straightforward to compute. Since only exact information is used in this variant (either via the oracle or directly from h), convergence follows from Theorem 3.2.

Algorithm 1 with the modified Steps 1' and 3' corresponds to the Progressive Augmented Lagrangian algorithm in [12], with the additional flexibility of allowing for varying stepsizes and bundle selection or compression. Specifically, to manage the bundle size, as only a bundle for d is defined, instead of (3.7), the aggregate linearization is the affine function $\check{d}^k(u^{k+1}) + \langle v^{k+1}, u - u^{k+1}\rangle = \langle v^{k+1}, u\rangle$. □

4. Handling inexactness at all iterations. Except for Example 3.3, the models considered so far are built with exact oracle information. Since exact evaluations involve knowing the difficult set of p-efficient points, pursuing further the track of the partially inexact variant from Example 3.3 is appealing. For instance, requiring that evaluations at serious steps become exact only asymptotically. Or simply accepting inexact values, as long as the "noise" produced by the evaluation error does not interfere much with the bundle optimization process.

We now put in place an *on-demand accuracy* bundle method, able to handle oracles that perform approximate evaluations: in (3.1) one may have $\eta_{u^j} > 0$ even when u^j becomes a serious step. If $\eta_{\hat{u}^k}$ is not null, the predicted increase in (3.6) (and the aggregate gap in (3.11)) may become negative. When this situation arises, the test (3.5) is meaningless because it no longer checks ascent. To handle this situation, Step 2.2 of Algorithm 2 declares noise as being "excessive" when the aggregate gap $\hat{e}_k$ becomes "too" negative.

4.1. Oracles with on-demand accuracy. In order to reduce the time spent in the oracle calculations, we now consider that the information is delivered with an inaccuracy η_{u^j}, as follows:

$$\left\{\begin{array}{lcll} h_u^j & = & c(x^j)-\langle u^j, g(x^j)\rangle & \text{and} \quad s_h^j = -g(x^j) \text{ for } x^j \in X \\ d_u^j & = & \langle u^j, v^j\rangle & \text{and} \quad s_d^j = v^j \text{ for } v^j \in \operatorname{conv}\mathcal{V} \\ \varphi_{u^j} & = & h_u^j + d_u^j & \text{and} \quad s^j = s_h^j + s_d^j \text{ are such that} \\ \varphi_{u^j} & & \text{satisfies (3.1)} & \text{and} \quad -s^j \in \partial_{\eta_{u^j}}(-\varphi)(u^j), \ \text{ with } \ \eta_{u^j} \geq 0\,. \end{array}\right. \tag{4.1}$$

The list below describes several possibilities for the oracle inaccuracy, the acronyms between parentheses refer to the corresponding methods benchmarked in Section 6:

- Having $\eta_{u^j} \equiv 0$ corresponds to the exact oracles in (2.7) and (2.8) (BM1, PAL).
- The case in which $\eta_{u^j} = 0$ if u^j yields a serious step gives the partially inexact Example 3.3 (BM2, BM3).
- An asymptotically exact method drives the inaccuracy to zero for all points (BM4).
- A partially asymptotically exact algorithm drives $\eta_{\hat{u}^k} \to 0$ (BM5).

An inexact oracle designed to work in an *on-demand accuracy* mode returns functional values with error smaller than η_{u^j}, sent as an input by the optimization procedure. For the functions h and d we now explain how such a mechanism can be put in place while still ensuring (3.4), a property crucial to have upper models and show convergence.

The departing idea is that, as both h and d in (2.4), (2.5) involve minimizing an objective function that is linear on the dual variable u, any feasible point realizes the η-subgradient inequality in (4.1). For the function h, for instance, let $u^j \in \mathbb{R}^m$ and a bound $\eta_h^j \geq 0$ be given. Suppose without loss of generality that both c and g are linear functions, so that a primal-dual Linear Programming solver is called. If the value η_h^j is set as stopping tolerance of the solver, the output will be a point $x^j \in X$ satisfying

$$\left[c(x^j) - \langle u^j, g(x^j)\rangle\right] - h(u^j) \leq \eta_h^j\,. \tag{4.2}$$

Taking $h_{u^j} := c(x^j) - \langle u^j, g(x^j)\rangle$ satisfies $h_{u^j} \in [h(u^j), h(u^j)+\eta_{u^j}]$, i.e., condition (3.1) for the function h. Similarly for (4.1), taking as subgradient $s_h^j := -g(x^j)$:

$$\begin{aligned} h(u) &= \min_{x\in X}\Big\{c(x) - \langle u, g(x)\rangle\Big\} = \min_{x\in X}\left\{c(x) - \left\langle u^j, g(x)\right\rangle + \left\langle -g(x), u-u^j\right\rangle\right\} \\ &\leq c(x^j) - \left\langle u^j, g(x^j)\right\rangle + \left\langle -g(x^j), u-u^j\right\rangle \qquad (4.3) \\ &= h_u^j + \left\langle s_h^j, u-u^j\right\rangle =: L_h^j(u) \\ &\leq h(u^j) + \eta_h^j + \left\langle s_h^j, u-u^j\right\rangle, \qquad (4.4) \end{aligned}$$

where we used (4.2). Furthermore, and in spite of inexactness, the linearization is still of the upper type. This follows from combining the bottom conditions in (4.1):

$$\varphi(u) \leq \varphi(u^j) + \left\langle s^j, u-u^j\right\rangle \leq \varphi_{u^j} + \left\langle s^j, u-u^j\right\rangle =: L^j(u)\,,$$

ensuring (3.4) also with the inexact oracle. This validates the use of upper models M^k in Algorithm 2.

An on-demand oracle for d can be devised similarly, stopping prematurely the solver for the d-problem in (2.5). Since significant speed-up can be obtained for this oracle, Section 5 considers this alternative, named Heuristic h2, as well as three additional heuristics for computing d inexactly.

The boundedness property (3.2) is straightforward for (4.4), because the inaccuracy is controlled by the bundle solver. We refer to [22, 31, 32, 44–46] for different inaccurate oracles and ways to deal with inexactness.

4.2. Proximal bundle method for inaccurate oracles. We now describe the few modifications that need to be brought into PBM when the oracle call delivers inexact information.

The ascent test as well as the aggregate gap and Fenchel measure remain as in (3.5) and (3.10), respectively. The difference is that now the predicted increase and the gap can be negative. In particular, by (3.10),

$$\delta^k \geq 0 \iff \hat{e}_k \geq -t_k|\hat{s}^k|^2,$$

so to make (3.5) a meaningful ascent test ($\delta^k \geq 0$) the *noise detection* step below checks if $\hat{e}_k < -\beta t_k \left|\hat{s}^k\right|^2$ for a parameter $\beta \in (0,1)$, as in [3, 32]. To *attenuate excessive noise*, the strategy proposed in [20, 22] can be adopted. Namely, increase the stepsize t_k and solve problem (2.9) changing neither the model nor the center. If no noise is detected, the predicted increase is nonnegative and the algorithm proceeds as Algorithm 1.

Algorithm 2 Concave Proximal Bundle Method for Upper Models with Inexact Oracles (PBM$_{\eta_{u^j}>0}$)

Step 0: initialization. As in Step 0 of Algorithm 1, but with inexact values satisfying (4.1), and noise parameters $\mathtt{na} = 0$ and $\beta \in (0,1)$.

Step 1.1: Next iterate. As in Step 1 of Algorithm 1, computing also $\hat{e}_k$ from (3.10).

Step 1.2: Noise detection. If $\hat{e}_k \geq -\beta t_k \left|\hat{s}^k\right|^2$ go to Step 2 (noise is not too cumbersome).

Step 1.3: Noise attenuation. Set $t_{k+1} = 10t_k$, $\mathtt{na} = 1$, $M^{k+1} = M^k$, $\hat{u}^{k+1} = \hat{u}^k$, $k = k+1$, go back to Step 1.1.

Step 2: Stopping test. As in Step 2 of Algorithm 1

Step 3: Oracle call. Compute an upper linearization L^{k+1} as in (3.4), using oracle information satisfying (4.1), written with u^j replaced by u^{k+1}.

Step 4: Ascent test. If (3.5) is satisfied (serious step), set $\hat{u}^{k+1} = u^{k+1}$, $\mathtt{na} = 0$ and choose $t_k \geq t_{\mathtt{low}}$.
Otherwise (null step), set $\hat{u}^{k+1} = \hat{u}^k$ and choose $t_{k+1} \in [(1-\mathtt{na})t_{\mathtt{low}} + \mathtt{na}t_k, t_k]$.

Steps 5 and 6. As in Algorithm 1.

In Algorithm 2 the parameter na is used to block a decrease of the stepsize t_k if the iterate is declared null and, after generating the current stability center, noise attenuation steps had been done in Step 1.3 (otherwise, since Step 1.3 increases t_k, stepsize zigzagging could hinder the convergence process).

Regarding convergence, a difference with PBM is that, in addition to the usual serious and null steps dichotomy, now the algorithm can loop forever inside of Step 1, trying to attenuate noise. Assumption (3.2) ensures that the gap is bounded below, by (3.11). In this situation, having infinitely often $\hat{e}_k < -\beta t_k|\hat{s}^k|^2$ in Step 1.1 while driving $t_k \to \infty$ makes $\hat{s}^k$ eventually null and once more Lemma 3.1 applies. These assertions are now formalized, making use of [32, Cor. 5.3 and Thm. 6.11].

THEOREM 4.1 (Primal and dual convergence for PBM$_{\eta_{u^j}>0}$). *Consider a primal problem* (2.2) *and its dual problem* (2.4) *such that the assumptions in Section 2.1 hold. Suppose that in Algorithm 2 the model satisfies* (3.3) *and the stopping tolerances are taken null (*$\mathrm{tol}_\phi = \mathrm{tol}_{\hat{s}} = 0$*). If the oracle satisfies* (4.1) *and* (3.2) *then the primal subsequence* $\{(\hat{x}^{k+1}, \hat{v}^{k+1})\}$ *always has limit points and any of them solves* (2.2)*, up to the asymptotic inaccuracy at serious points,* $\eta^\infty := \liminf \eta_{\hat{u}^k}$*:*

$$(\hat{x}^\infty, \hat{v}^\infty) \text{ is feasible in } (2.2) \text{ and } c(\hat{x}^\infty) \leq c(x) + \eta^\infty \text{ for all } (x,v) \text{ feasible in } (2.2).$$

As for the center subsequence $\{\hat{u}^k\}$*, any of its limit points* $\hat{u}^\infty$ *(when they exist) solves approximately* (2.4)*:*

$$\varphi(u) \leq \varphi(\hat{u}^\infty) + \eta^\infty \text{ for all } u \in \mathbb{R}^m_+.$$

Proof. Once more, the statements will follow from showing (3.16) and applying Lemma 3.1. When the algorithm stops at some iteration k the result is straighforward. If the algorithm loops forever, first recall the relations in

Table 3.1, which remain all valid. When $k \to \infty$, in addition to the two sets defining K^∞ in Theorem 3.2, a third possibility arises:

either $K^\infty := \{k \geq \hat{k} : \hat{e}_k < -\beta t_k |\hat{s}^k|^2\}$ if after a last serious iteration $\hat{k}$ Step 1.3 occurs infinitely many times,
or K^∞ is one of the two sets in Theorem 3.2.

When K^∞ corresponds to the new case, Corollary 5.3 in [32] shows that the last serious step before an infinite loop of noise attenuation is an $\eta_{\hat{u}^{\hat{k}}} + \eta^M$-solution to (2.4). Since $\eta^M = 0$ in our setting (see the $5th$ line in Table 3.1)), the assertion follows. For the other two cases (infinitely many serious steps or a tail of null steps), Theorem 6.11 in [32] ensures (3.16), provided that two conditions, called (6.15) and (6.16), are satisfied. To check satisfaction of both conditions, we start with (6.15), which in turn gathers three conditions, all satisfied by $\text{PBM}_{\eta_{u^j}>0}$:

1. since the model is of upper type, by (3.3) the relation in [32, Eq. (4.11)] holds with $\eta^M = 0$;
2. the level $\ell_k = -\varphi_{\hat{u}^k}$ satisfies [32, Eq. (3.9)]; and
3. [32, Eq. (6.14)] is the update rule for t_k in Algorithm 2.

As for (6.16), it also gathers three relations, namely [32, Eqs. (6.11) and (6.12)], and a third condition that holds with our choice $\alpha_k = 0$ from Table 3.1 for any $\beta_k \in (0,1)$. Both (6.11) and (6.12) are asymptotic relations involving a quantity $\delta_E^k = \delta^k$ in our case (last line in Table 3.1). Like in Theorem 3.2, condition (6.11) is trivially satisfied, because the expression in the last line in Table 3.1 is nonpositive. Finally, (6.12) requires convergence of the δ_E^k-series, which follows from the inequalities below

$$0 \leq \sum_{k\in\hat{K}} \delta_k^E = \sum_{k\in\hat{K}} \delta^k = \sum_{k\in\hat{K}} \left(\varphi_{\hat{u}^{k+1}} - \varphi_{\hat{u}^k}\right) \leq c^* + \eta - \varphi(u^1) < +\infty,$$

for c^* is the exact optimal (primal and dual) value and where we used the inequalities $\varphi_{\hat{u}^{k+1}} \leq \varphi(\hat{u}^{k+1}) + \eta_{\hat{u}^{k+1}} \leq c^* + \eta$, derived from (4.1) and (3.2), together with $\varphi_{\hat{u}^k} \geq \varphi(\hat{u}^k)$ from (3.1).

Since (3.16) holds with all the possibilities for the set K^∞, the desired results follow from Lemma 3.1. □

Assumption (3.2) does not require an explicit knowledge of the oracle error bound η, its mere existence suffices (the bound is not used by the algorithm). For the on-demand accuracy setting, finding a uniform bound η is an easy task (the inaccuracy corresponds to the stopping tolerance for solving the h and d oracles). The situation is more delicate with the partially inexact model in Example 3.4 if the dual sequence becomes unbounded. When this situation is detected, Step 4' in the example needs to be modified to request exact oracle evaluations until a serious iteration can be declared.

5. Approximating p-efficient points. For the inexact d-evaluations we generate elements in the level set $\mathcal{Z}$, (larger than $\mathcal{V}$, but convex, see Section 2.1). We refer to these elements as *approximate p-efficient points*.

5.1. Discrete distributions. We present and compare four heuristics fitting the requirements (4.1), when the random vector $\xi \in \mathbb{R}^m$ has finitely many realizations $\xi^1, \xi^2, \ldots, \xi^N$ with probabilities $\pi_1, \pi_2, \ldots, \pi_N$, all positive. Let $b \in \mathbb{R}^m$ be defined as $b_j = \min_{1\leq i\leq N} \xi_j^i$, $j = 1, \ldots, m$, and let $u \in \mathbb{R}^m_+$ be given. Following [27]; see also [12, § 6], we write the d-problem in (2.5) as the MILP problem

$$d(u) = \left\{ \begin{array}{lll} \min\limits_{(v,z)\in\mathbb{R}^{m\times N}} & \langle u, v\rangle & \\ \text{s.t.} & \xi^i(1-z_i) & \leq v - bz_i,\ i = 1,\ldots,N, \\ & \sum_{i=1}^N \pi_i z_i & \leq 1-p, \\ & z_i \in \{0,1\}, & i = 1,\ldots,N. \end{array} \right. \tag{5.1}$$

For any feasible $(\bar{v}, \bar{z})$, $\bar{v}$ is an approximate p-efficient point, solving (5.1) gives the "best approximation". For large N, as the computational effort for solving problem (5.1) can become prohibitive, [27] and [23] study "cheaper" combinatorial and mathematical programming formulations. We propose an alternative to (5.1) that, instead of mixed 0-1 variables, uses only binary variables. If $(v,z) \in \mathbb{R}^m \times \{0,1\}^N$ is a feasible point for problem (5.1), then

$$\sum_{i\in I_0(z)}^N \pi_i \geq p \ \text{ and } \ \xi^i \leq v \text{ for all } i \in I_0(z) := \{1 \leq l \leq N : z_l = 0\}.$$

Solving the combinatorial problem below is equivalent to solving problem (5.1)

$$d(u) = \min_{z\in\{0,1\}^N} d_u(z) \text{ s.t. } \sum_{i=1}^{N} \pi_i z_i \le 1-p\,, \tag{5.2}$$

where we defined problem (5.1)

$$d_u(z) := \min_{v\in\mathbb{R}^m} \langle u, v\rangle \text{ s.t. } \xi^i \le v \text{ for all } i \in I_0(z)\,.$$

Even though d_u is neither convex nor continuous on $z \in \{0,1\}^N$, it has the quality of being easy to evaluate. Namely, its minimum is attained at the point

$$\tilde{v} \in \mathbb{R}^m \text{ such that } \tilde{v}_j := \max_{i\in I_0(z)} \xi^i_j \text{ for all } j = 1,\dots,m\,, \tag{5.3}$$

with optimal value

$$d_u(z) = \sum_{j=1}^{m} u_j \left[\max_{i\in I_0(z)} \xi^i_j\right]\,. \tag{5.4}$$

Based on these observations, we defined four different heuristics to solve approximately (5.2), named Heuristics h1, h2, h3, and h4.

Heuristic h1 (Incremental Selection. Input: $u \in \mathbb{R}^m_+$, $p > 0$, as well as ξ^i and π_i for all $i = 1,\dots,N$)

Step 1. Take $z = 0 \in \{0,1\}^N$ (or any point feasible for (5.2)).

Step 2. For all $i = 1,\dots,N$ such that $z_i = 0$, define new trial points $\tilde{z}^i = z$ with $z^i_i = 1$ ($\tilde{z}^i$ differs from z only by its i^{th} component)

Step 3. Evaluate function d_u (given in (5.4)) at all new trial points $\tilde{z}^i$ which are feasible for (5.2)

Step 4. If all trial points $\tilde{z}^i$ are infeasible, stop and return $(d_u(z), \tilde{v})$, where $\tilde{v}$ defined in (5.3) is an approximate p-efficient point.
Otherwise, set z as being the best trial point $\tilde{z}^i$, and go back to Step 1.

As each cycle between Steps 1 and 4 switches a single component of z ($z_i = 0$ becomes $z_i = 1$), Heuristic h1 terminates after finitely many cycles. Instead of working with individual components, a full block could be changed at once, to reduce the number of cycles between Steps 1 and 4, and shorten the CPU time (keeping in mind that the approximation would be less accurate).

An important feature of Heuristic h1 is that, for a sufficiently large probability, the heuristic can be exact, because the bigger is $p \in (0,1)$, the better is the approximate p-efficient point. For large enough p, the feasible points z of (5.2) have only one nonzero component, and the Heuristic h1 will check all the possible combinations, thus providing an optimal solution to problem (5.2).

The next heuristic is similar to the on-demand accuracy oracle presented for h in Section 4.1.

Heuristic h2 (On-Demand Accuracy, input as in Heuristic h1)

Procedure: Stop the MILP solver for (5.1) as soon as it finds a feasible point.

The final heuristics exploit the fact that in (5.4) computing $d_u(z)$ is easy.

Heuristic h3 (Derivative-free, input as in Heuristic h1)

Procedure: Solve problem (5.4)-(5.2) using the derivative-free algorithm given in [14] and available on `http://www.i2c2.aut.ac.nz/Wiki/OPTI/index.php`, see also [8].

Heuristic h4 (Matlab, input as in Heuristic h1)

Procedure: Solve problem (5.4)-(5.2) with Matlab's genetic algorithm `ga`.

For the numerical benchmark, we randomly generated 32 instances of problem (5.1). Each component ξ_i^j of the sample $\{\xi_1,\ldots,\xi_N\}$ follows a normal distribution with mean 10 and standard deviation 5. Random dual vectors $u\in\mathbb{R}^m$ were generated using the command `u=sprand(m,1,.6)` in `Matlab`. The problem sizes are

$$m\in\{50,\,100\}\ \text{ and }\ N\in\{20,\,50,\,80,\,100,\,300,\,500,\,1000,\,2000\},$$

and the probability level $p\in\{0.90,\,0.95\}$.

The runs were performed on a computer with one 2.40GHz processor and 4Gb RAM. To have a reference for comparison, we first run Gurobi 5.6, either until optimality, or the computational time reached its limit of 600 seconds (in the many instances for which time limit was attained, the output point was nearly optimal). The first table in Appendix 6 reports in its fifth column the final functional values, considered as the exact value of $d(u)$ in the benchmark. The table also reports, for each instance and all the heuristics, both the relative errors and CPU time reduction, computed with respect to the exact values. The time reduction is negative when, rather than reducing the CPU time, it took longer for the variant to provide an output. Figure 5.1 displays these results graphically.

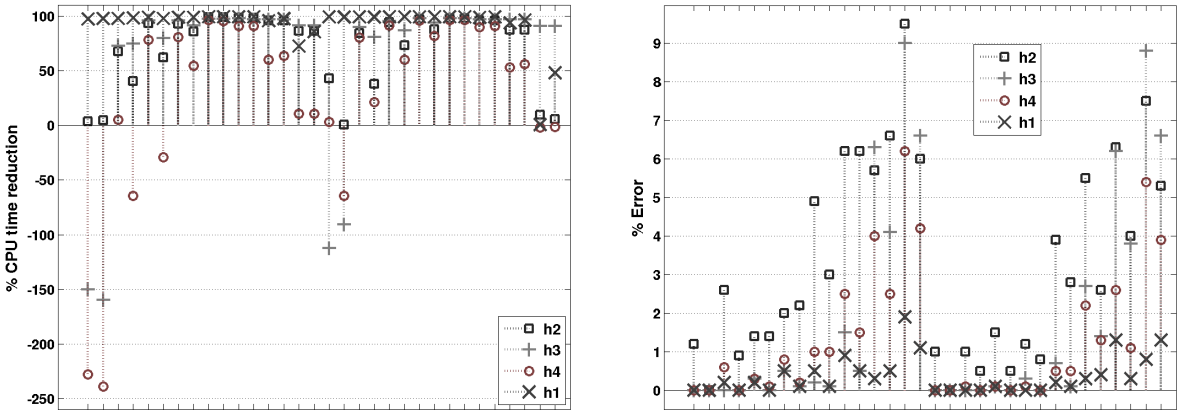

FIG. 5.1. *Time reduction (left) and evaluation error (right) for Heuristics h1-h4.*

We note that Heuristic h1 performs best, both in terms of accuracy and speed: errors are within a range of 1.9%, and the average time reduction is above 93%. Heuristic h2 is competitive in terms of speed, but presents a larger variation in the error. An interesting feature of this variant is that it can exploit warm starts. Indeed, when a feasible point is found, the variant can return to the bundle solver, to check satisfaction of (3.5). If deemed necessary (the point has potential to become a serious step), Heuristic h2 can continue the optimization process from the point where it was frozen. The numerical experience on the bundle methods in Section 6 shows the positive impact of this strategy to reduce computational time (for instance in Table 6.2),

5.2. Continuous distributions. When the random variable has an infinite support, we now explain how to obtain approximate p-efficient points by combining sampling and restoration. We also provide new theoretical insights on the link between the sample size N and feasibility for the continuous distribution.

Sampling. Consider (5.1) defined for a given sample with N realizations $\xi^1,\ldots,\xi^N$, and let $\tilde{v}$ be a feasible point, computed for instance by means of Heuristic h1.

In general, $\tilde{v}$ will not be feasible for the continuous distribution, i.e., $\mathbb{P}[\xi\leq\tilde{v}]<p$, so $\tilde{v}$ is not an approximate p-efficient point. To ensure that $\tilde{v}\in\mathcal{Z}$, the restoration step explained below can be used.

Restoration. Suppose that a Slater point for (5.1), satisfying $\mathbb{P}[\xi\leq v^s]>p$ with the continuous distribution is available (such points are relatively easy to compute because $\lim_{v\to\infty}\mathbb{P}[\xi\leq v]=1$ and, hence, taking sufficiently large components ensures the inequality is strict). Restoration is achieved by computing (the smallest) $\lambda\in(0,1)$ such that $\mathbb{P}[\xi\leq v(\lambda)]\geq p$, where $v(\lambda)=\lambda\tilde{v}+(1-\lambda)v^s$. In general this procedure requires a few interpolation steps only, depending on the required accuracy for feasiblity.

Linking the sample size to approximate p-efficiency. We start with a technical result relating feasibility for the continuous distribution with feasibility for problem (5.1). Differently from [26], we base our proof on the simultaneous use of Bernstein's and Hoeffding's bounds.

LEMMA 5.1. *Let $G : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$ be a mapping, $\xi \in \mathbb{R}^m$ a multivariate random variable and consider the probabilistic constraint $\psi(x) := \mathbb{P}[G(x,\xi) \leq 0] \geq p$, with feasible set $C := \{x \in \mathbb{R}^n : \psi(x) \geq p\}$. For $N \geq 1$, let $\xi^1, ..., \xi^N$ be an i.i.d sample of ξ with approximate feasible set*

$$M_q^N := \left\{ x \in \mathbb{R}^n : \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{G(x,\xi_i) \leq 0} \geq q \right\},$$

where $1 > q > p$. For any $x \notin C$, the following estimate holds true:

$$\mathbb{P}[x \in M_q^N] \leq \exp\left(-N(q-p)^2 \max\left\{ 2, \frac{1}{2\psi(x)(1-\psi(x)) + \frac{2}{3}(q-p)} \right\} \right).$$

Proof. Fix arbitrary $x \notin C$, so that $\psi(x) < p$ by definition. For any $i = 1, \ldots, N$ define the Bernoulli random variable $Y_i = \mathbb{1}_{G(x,\xi_i) \leq 0}$, noting that $Y_i = 1$ with probability $\psi(x)$, $Y_i = 0$ otherwise, and $\mathbb{E}(Y_i) = \psi(x)$. Now

$$\begin{aligned} \mathbb{P}[x \in M_q^N] &= \mathbb{P}[\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{G(x,\xi_i) \leq 0}] = \mathbb{P}[\sum_{i=1}^{N} Y_i \geq qN] = \mathbb{P}[\sum_{i=1}^{N} (Y_i - \mathbb{E}(Y_i)) \geq qN - \psi(x)N] \\ &\leq \mathbb{P}[\sum_{i=1}^{N} (Y_i - \mathbb{E}(Y_i)) \geq (q-p)N] \leq \exp\left(-2N(q-p)^2\right), \end{aligned}$$

where we used Hoeffding's inequality and $\psi(x) < p$. Using instead Bernstein's inequality yields

$$\mathbb{P}[x \in M_q^N] \leq \exp\left(-\frac{1}{2} \frac{N^2(q-p)^2}{\psi(x)(1-\psi(x))N + \frac{1}{3}N(q-p)} \right),$$

since $\mathrm{Var}(Y_i) = \psi(x)(1-\psi(x))$. Since both inequalities hold simultaneously, the probability is also bounded by their minimum, which leads to the given expression. □

The previous lemma is an important technical tool to derive a statement concerning "$M_q^N \subseteq C$", i.e., with what confidence level a point that is feasible for problem (5.1) is feasible for the continuous probabilistic constraint.

Before proceeding, a covering argument is needed, so we define a grid which roughly covers the zone of interest. Here the advantage of being able to switch between Hoeffding's and Bernstein's bounds shows its full potential (the weaker form, using only Hoeffding's bound, is [26, Thm. 5]). Since many points in the grid will have a relatively low probability level, often the feasible region C has a very gully-shaped look, with "many" x having small $\psi(x)$ and the mapping $\psi(x)$ rapidly increases over a small range; [28]. The advantage of using Bernstein's bound over Hoeffding's bound arises when $\psi(x) \leq \frac{1}{2} - \sqrt{\frac{1}{4} + (\frac{1}{3}(q-p) - \frac{1}{4})}$ or $\psi(x) \geq \frac{1}{2} - \sqrt{\frac{1}{4} + (\frac{1}{3}(q-p) - \frac{1}{4})}$. Therefore if many grid-points have a low probability level, a significant improvement could be obtained; the numerical results below confirm this expectation; see Figure 5.2.

Accordingly, in addition to the assumptions of Lemma 5.1, suppose the set $X \subseteq \mathbb{R}^n$ is compact. For $\eta > 0$, a collection of points $\mathcal{G} := \{x_i\}_{i=1}^K \subseteq X$ is called a *g-dominating η-lattice* if and only if for any $x \in X$ there exists $x_i \in \mathcal{G}$ with $|x - x_i| \leq \eta$ and $G(x,\xi) \leq 0$ implies $G(x_i,\xi) \leq 0$ almost surely.

The notion of a g-dominating η-lattice is very similar to the requirements of precedence in [39]. For the separable case ($G(x,\xi) = g(x) - \xi$), the construction in [26, Theorem 9] shows how such a g-dominating η-lattice can be set up. To this end, let l be the lower bound for $\mathcal{V}$ in Section 2.1. By compactness of X there exists $U \in \mathbb{R}^k$ such that $g(x) \leq U$ for all $x \in X$. We may therefore restrict our attention to the set $\{x \in X : l \leq g(x) \leq U\}$ without loss of generality. Now define the set of points $Y_j = \left\{ l_j + i\frac{(U_j - l_j)}{P}, i = 1, ..., P \right\}$ for each $j = 1, ..., k$ and $\mathcal{G} = \prod_{j=1}^k Y_j$. Then for any $y \in [l, U]$ we can find $y' \in \mathcal{G}$ such that $y \leq y'$ and $|y - y'| \leq \eta$. Indeed define the jth component of y' as $y'_j = \min_{w \in Y_j : w \geq y_j}$. By construction $y' \geq y$ and $\left| y'_j - y_j \right| = y'_j - y_j \leq \frac{U_j - l_j}{P}$, which entails

$|y'-y|_\infty \leq \max_{1\leq j\leq k} \frac{U_j-l_j}{P}$. By equivalence of norms in $\mathbb{R}^k$, it is immediate that one can select P in such a way as to make the right-hand size smaller than any desired $\eta > 0$.

We are now in a position to link the sample size N in problem (5.1) with feasibility of the resulting solutions for the probabilistic constraint with continuous distribution:

THEOREM 5.2. *With the assumptions and notation in Lemma 5.1, suppose that the set $X \subseteq \mathbb{R}^n$ is compact and ψ is Lipschitz continuous with constant L (w.r.t. the norm $|\cdot|$). Let $\mathcal{G} := \{x_i\}_{i=1}^K$ be a g-dominating η-lattice, for $\eta > 0$ such that $L\eta \in (0, q-p)$. For the approximate feasible set M_q^N given in Lemma 5.1, we have that*

$$\mathbb{P}[M_q^N \subseteq C] \geq 1 - \sum_{j=1}^{K} \exp\left(-N(q-p-L\eta)^2 \max\left\{2, \frac{1}{2\psi(x_j)(1-\psi(x_j)) + \frac{2}{3}(q-p-L\eta)}\right\}\right) \mathbb{1}_{x_j \notin M(p+L\eta)}.$$

Proof. We begin by showing that $\mathbb{P}[M_q^N \subseteq C] \geq \mathbb{P}[M_q^N \cap \mathcal{G} \subseteq M(p+L\eta) \cap \mathcal{G}]$. To this end assume that $M_q^N \cap \mathcal{G} \subseteq M(p+L\eta) \cap \mathcal{G}$ holds for a given random realization, the dependency on which we will not explicit. Let $x \in M_q^N$ be arbitrary, then there exists $x_i \in \mathcal{G}$ such that $|x - x_i| \leq \eta$. Consequently, by g-domination, $G(x, \xi^j) \leq 0$ implies $G(x_i, \xi^j) \leq 0$ and hence $\frac{1}{N}\sum_{j=1}^N \mathbb{1}_{G(x,\xi^j)\leq 0} \geq q$ implies $\frac{1}{N}\sum_{j=1}^N \mathbb{1}_{G(x_i,\xi^j)\leq 0} \geq q$, i.e., $x_i \in M_q^N \cap G$. This implies by assumption that $x_i \in M(p+L\eta) \cap G$. By g-domination of $\mathcal{G}$, we have $\psi(x) \geq \psi(x_i)$, which when combined with Lipschitz continuity of ψ, gives $\psi(x) \geq \psi(x_i) - L|x - x_i| = \psi(x_i) - L\eta \geq p + L\eta - L\eta = p$. We have thus shown that $x \in C$ and hence $M_q^N \subseteq C$. This means that we have derived that "$M_q^N \cap \mathcal{G} \subseteq M(p+L\eta) \cap \mathcal{G} \Rightarrow M_q^N \subseteq C$".

We will now bound $\mathbb{P}[M_q^N \cap \mathcal{G} \subseteq M(p+L\eta) \cap \mathcal{G}]$. By using Lemma 5.1 we obtain:

$$\begin{aligned}\mathbb{P}[M_q^N \cap \mathcal{G} \nsubseteq M(p+L\eta) \cap \mathcal{G}] &= \mathbb{P}[\,\exists x_j \in M_q^N \cap \mathcal{G} \,:\, x_j \notin M(p+L\eta)] \leq \textstyle\sum_{j=1}^K \mathbb{P}[x_j \in M_q^N]\, \mathbb{1}_{x_j \notin M(p+L\eta)} \\ &\leq \textstyle\sum_{j=1}^K \exp\left(-N(q-p-L\eta)^2 \max\left\{2, \frac{1}{2\psi(x_j)(1-\psi(x_j)) + \frac{2}{3}(q-p-L\eta)}\right\}\right) \mathbb{1}_{x_j \notin M(p+L\eta)}.\end{aligned}$$

As the left hand side is larger than $1 - \mathbb{P}[M_q^N \cap \mathcal{G} \nsubseteq M(p+L\eta) \cap \mathcal{G}]$, the stated result follows. □ For the graphical illustration in Figure 5.2, we use the bound below, obtained as a particularization of Theorem 5.2:

$$\begin{gathered}\text{Suppose that the set } \{x_j \in \mathcal{G} : \psi(x_j) \leq 0.1\} \text{ contains } K_1 \leq K \text{ points.} \\ \text{Then } \mathbb{P}[M_q^N \subseteq C] \geq 1 - K_1 \exp\left(-\frac{N(q-p-L\eta)^2}{\frac{18}{100} + \frac{2}{3}(q-p-L\eta)}\right) - (K-K_1)\exp\left(-2N(q-p-L\eta)^2\right),\end{gathered}$$

noting that when taking in the set $\psi(x_j) \leq 0.01$ the constant $\frac{18}{100}$ above is to be replaced by 0.0198.

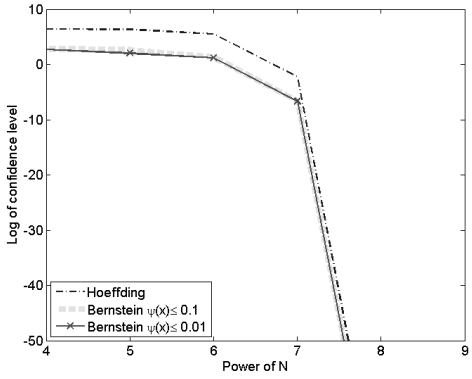

FIG. 5.2. *The size of N versus precision for the cash matching problem, with a logarithmic scale.*

For the cash-matching problem [18], we let $K_1 \approx 0.98K$ and, for a given confidence level $1-\delta$ and using concrete data $p = 0.8$, $q = 0.802$, $\eta = 0.001$, $L = 1$, we plotted the dependency of δ on N in Figure 5.2. The advantage of using Bernstein's over Hoeffding's bound can be noticed in the figure, which shows that the change roughly brings a gain of one order of magnitude on N. Further improvements can be achieved when for $0.98K$ points in the grid the probability is $\psi(x) < 0.01$, as shown by comparing the two Bernstein's plots in Figure 5.2.

6. Numerical Comparison of Solvers. We now compare 10 different methods (all implemented in C++), obtained by combining several models (Examples 2.1-2.3) and oracles (approximating p-efficient points).

Instances. We consider seven problems listed in Table 6.1, they are linear programs with bilateral probabilistic constraints fitting (2.1), with $\xi \in \mathbb{R}^m$ a centered multivariate Gaussian random variable,

$$X = \{x \in [\underline{x}, \overline{x}] \subset \mathbb{R}^n : Ax \leq b\}, \quad \text{and} \quad \mathbb{P}[a^r + A^r x \leq \xi \leq B^r x + b^r] \geq p\,.$$

TABLE 6.1
Size of problems in the benchmark. Here # A stands for the number of rows in matrix A .

Problem	n	# A	m	p	description
CM	3	1	15	0.9	cash matching [18]
Ain48	672	1296	48	0.8	cascaded reservoir management [2]
Isr48	566	268	48	0.8	cascaded reservoir management [2]
Isr96	566	172	96	0.8	cascaded reservoir management [2]
Isr168	566	28	168	0.8	cascaded reservoir management [2]
PTP1	2000	40	50	0.9	probabilistic transportation problem [26]
PTP2	2000	40	50	0.9	probabilistic transportation problem [26]

For each problem, we considered 6 different instances, corresponding to $N \in \{50, 100, 250, 500, 1000, 2000\}$.

Oracles. The calculations for h in (2.4) involve solving a simple LP problem, so the h-oracle is exact. For the d-oracle (2.5), the oracle error bound in (3.2) is $\eta^\infty = 1e^{-4}$. We defined several on-demand accuracy versions: at iteration k the oracle receives u^{k+1}, a target $\texttt{tar}^k$ (typically some value greater than $\varphi_{\hat{u}^k}$, for instance the right hand side term in the ascent test (3.5)), and a bound $\texttt{gap}^k$ for the inaccuracy $\eta_{u^{k+1}}$.

To compute the oracle output, calculations are split into a *coarse* and a *fine* phase:

1. The coarse phase uses Heuristic h1 or h2 from Section 5 to define an estimate d_u^{k+1} for $d(u^{k+1})$. This phase is meant to be fast (only a few minutes). Then we check if the target is reached
$$h(u^{k+1}) + d_u^{k+1} < \texttt{tar}^k\,.$$
If such is the case, the oracle returns the information to the bundle method and the algorithm proceeds in Step 4 to test for ascent. Otherwise, if $h(u^{k+1}) + d_u^{k+1} \geq \texttt{tar}^k$, the oracle passes to the next phase.
2. The fine phase computes better estimates by solving problem (5.1) until reaching either a relative gap inferior to $\texttt{gap}^k$ or the one hour CPU time limit.

Solution methods. The following bundle method (BM) variants were considered for benchmarking:

- BM1: $\texttt{tar}^k = -\infty$ and $\texttt{gap}^k = 10^{-4}$. In view of the target choice, this method always passes to the fine phase. This variant roughly corresponds to an exact PBM, except when the time limit of 1 hour is reached and the oracle provides inexact information.
- BM2: $\texttt{tar}^k = \varphi(\hat{u}^k)$ and $\texttt{gap}^k = 10^{-4}$. With this choice, the method requires exact (or at least more accurate) d-oracle information at some iterates which provide some ascent with respect to the threshold $\varphi(\hat{u}^k)$. This is PBM with $\eta_{\hat{u}^k} = 0$.
- BM3: $\texttt{tar}^k = \varphi(\hat{u}^k) + m\,\delta^k$ and $\texttt{gap}^k = 10^{-4}$. This method requires exact (or at least more accurate) d-oracle information at serious iterates. This is another version of PBM with $\eta_{\hat{u}^k} = 0$.
- BM4: $\texttt{tar}^k = -\infty$ and $\texttt{gap}^k = \min\left\{0.5, \frac{0.01\delta^k}{k}\right\}$. Since the oracle error vanishes because $\texttt{gap}^k$ goes to zero, the d-oracle information is asymptotically exact (at all iterations).
- BM5: $\texttt{tar}^k = \varphi(\hat{u}^k) + m\,\delta^k$ and $\texttt{gap}^k = \min\left\{0.5, \frac{0.01\delta^k}{k}\right\}$. The d-oracle information is asymptotically exact at serious steps, as in the partially inexact method in Remark 3.1.
- PAL: $\texttt{tar}^k = -\infty$ and $\texttt{gap}^k = 10^{-4}$, and the partially model from Example 2.3. As explained in Example 3.4, this is the *Progressive Augmented Lagrangian - PAL* method.

All the method have a total CPU time limit of 48 hours. Unless stated otherwise, for each method we used the aggregate cutting-plane model in Example 2.1 and the disaggregate one given in Example 2.2. The bundle methods use $m = 0.1$ to test for descent in (3.5), $\beta = -1.0$ to test for noise in Step 1.2 of $\text{PBM}_{\eta_{u^j}>0}$, and no

bundle compression/selection. The rule to update the stepsize t_k is the *poorman* formula in [5, Sec. 10.3.3]. To solve (2.9) we use the dual simplex QP method of CPLEX 12.4. The computations were carried out on Intel Xeon X5670 westmere computer cluster nodes with 8 Gb of reserved memory.

To refer to the many possible combinations, we use the following mnenomics: we append to BM1 (or BM2, BM3, etc) first the heuristic name and then a letter, a or d, to identify the aggregate or disaggregate cutting-plane model employed. For instance, when BM2 uses Heuristic h1 and an aggregate model, it is referred to as BM2h1a. Its counterpart using Heuristic h2 and disaggregate model is BM2h2d.

Results. To assess the quality of the obtained solution $\hat{x}^{k_{stop}+1}$ from (3.12) (iteration k_{stop} triggered the stopping test), we check feasibility by computing the probability $\mathbb{P}[a^r + A^r x^{k_{stop}+1} \leq \xi \leq B^r x^{k_{stop}+1} + b^r] \geq p$ using the code [17]. We also compute the relative optimality gap, and CPU time. Table 6.2 reports the output for problem Isr48 for the 10 methods and 6 scenario instances. We see that only the variants performing the coarse phase could solve all the instances. Differently from Section 5, Heuristic h2 performed better than Heuristic h1, likely thanks to its warm-starting capabilities. If no more than 1000 scenarios are employed, PAL is comparable to the fastest BM variants (with better feasibility and gap); otherwise, it fails.

TABLE 6.2
Problem Isr48

	N	BM1a	BM1d	BM2h1a	BM2h2a	BM2h2d	BM3h2d	BM4d	BM5h1d	BM5h2d	PAL
Gap (%)	50	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
	100	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.00
	250	0.01	0.00	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.00
	500	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.00	0.00
	1000	-	-	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.00
	2000	-	-	0.01	0.01	0.01	0.01	-	0.01	0.01	-
Infeas	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	100	0.01	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
	250	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	500	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
	1000	-	-	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
	2000	-	-	0.00	0.00	0.00	0.00	-	0.00	0.00	-
CPU (min)	50	0.03	0.03	0.01	0.01	0.03	0.03	0.01	0.01	0.01	0.03
	100	0.14	0.06	0.06	0.03	0.03	0.03	0.08	0.06	0.03	0.04
	250	2.84	1.39	1.21	0.18	0.14	0.09	0.78	0.93	0.11	0.23
	500	69.71	14.08	16.19	1.46	0.23	0.63	7.69	8.31	0.39	4.61
	1000	-	-	85.43	3.79	10.79	3.88	107.43	39.68	1.91	38.66
	2000	-	-	1250.06	146.93	102.28	107.03	-	599.34	9.48	-

TABLE 6.3
Problem PTP2

	N	BM1a	BM1d	BM2h1a	BM2h2a	BM2h2d	BM3h2d	BM4d	BM5h1d	BM5h2d	PAL
Gap (%)	50	0.01	0.01	0.01	0.01	0.03	0.04	-3.13	0.01	-3.13	0.00
	100	0.01	0.00	0.00	-0.01	0.00	0.00	-3.47	-0.01	-3.47	0.00
	250	-	-	-	-	-	0.00	-3.47	0.05	-3.47	0.00
	500	-	-	-	-	-	-	-2.34	-	-2.26	-
	1000	-	-	-	-	-	-	-4.50	0.01	-5.27	-
	2000	-	-	-	-	-	-0.50	-	0.02	-2.73	-0.06
Infeas	50	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60
	100	0.40	0.40	0.40	0.40	0.40	0.40	0.38	0.40	0.38	0.40
	250	-	-	-	-	-	0.23	0.20	0.23	0.20	0.23
	500	-	-	-	-	-	-	0.04	-	0.04	-
	1000	-	-	-	-	-	-	0.07	0.08	0.08	-
	2000	-	-	-	-	-	0.06	-	0.06	0.03	0.06
CPU (min)	50	5.73	5.26	2.33	2.56	1.99	1.36	0.19	0.04	0.08	0.11
	100	95.06	119.29	52.88	33.38	30.44	16.26	1.29	0.23	0.49	1.21
	250	-	-	-	-	-	1860.53	7.38	2.68	0.38	420.06
	500	-	-	-	-	-	-	38.98	-	0.98	-
	1000	-	-	-	-	-	-	374.73	145.71	5.83	-
	2000	-	-	-	-	-	1381.13	-	1971.34	10.03	420.16

Table 6.3 reports similar output, obtained for problem PTP2, a hard problem for which only the "asymptotic" methods BM4 and BM5 were able to solve more instances within the 48 hours time limit. We also observe a deterioration in PAL's performance. The ouput of the remaining problems in Table 6.1 can be found in Appendix 6.

Figure 6.1 reports the gap and infeasibility values for problems Ain48 and PTP1, both in absolute values and relative to the corresponding CPU time. For problem Ain48 good quality solutions can be obtained with small-

sized samples: already $N = 50$ realizations provided small optimality gap and infeasibility. The situation is very different with problem PTP1, as (almost) feasibility could only be reached $N \geq 1000$ (negative gaps for PTP1 in the figure are explained to lack of feasibility in the obtained solutions).

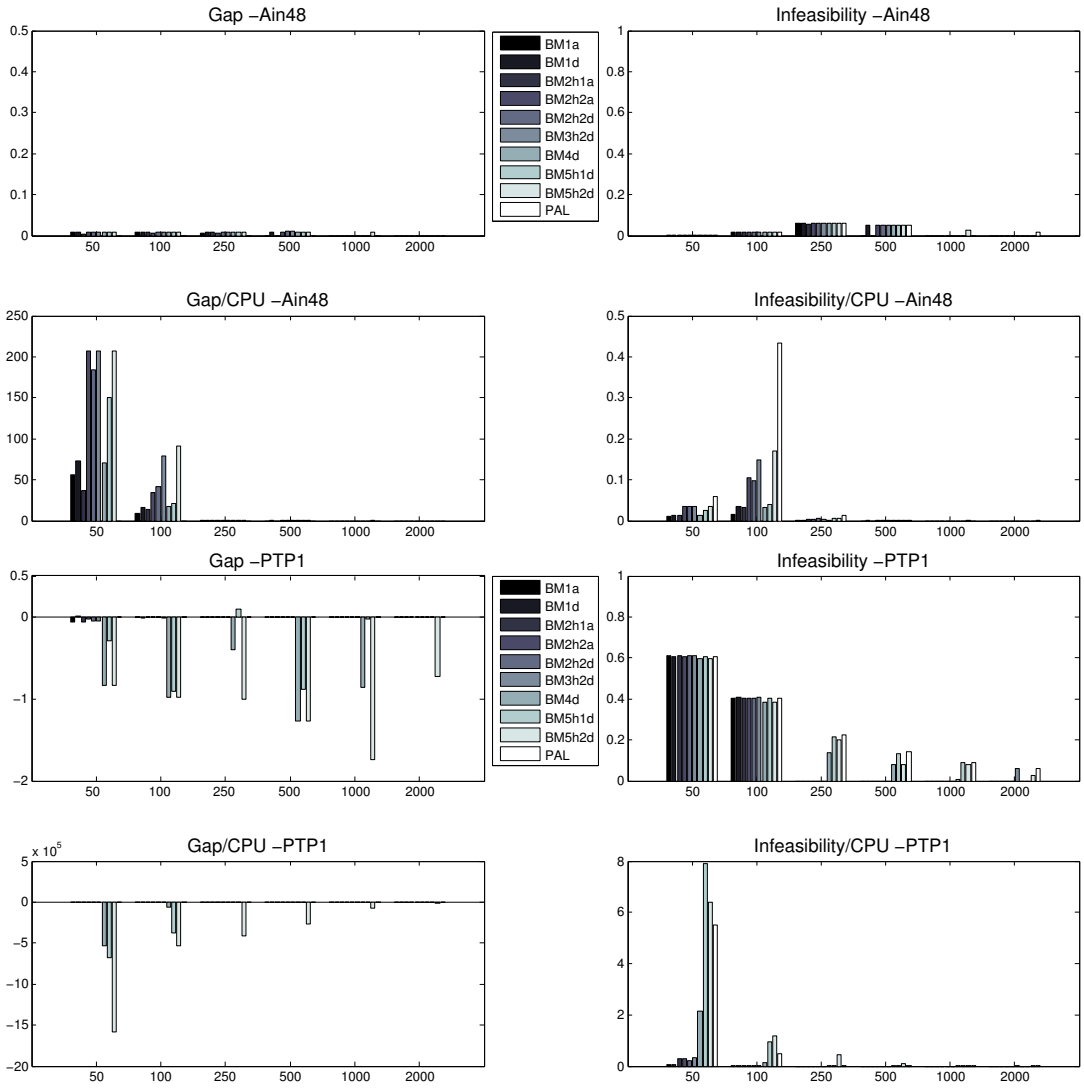

FIG. 6.1. *Problems Ain48 and PTP1: Optimality gap and infeasibility*

Figure 6.2 shows the performance profile on CPU times for the ten methods, over the 6 instances and 7 problems. For each method, the graph plots the proportion of problems that is solved within a factor of the time required by the best algorithm (denoted in the figure by $\varphi(\gamma)$ in the ordinate and γ in the abscissa); see [15]. When it comes to speed, the leftmost ordinate value gives the probability of each method to be the fastest in the benchmark. We observe that BM5h2d is the fastest almost 60% of the times, followed by PAL, which was fastest about 25% of the times. When it comes to robustness, the rightmost ordinate value gives the proportion of problems solved by each method. No method could solve all the instances (no line reaches the value 1), and the clear advantage of BM5h2d is once more confirmed (with almost 85% of the instances solved), followed by PAL (slightly more than 60% of problems solved).

Concluding Remarks. A unified framework for dual approaches based on p-efficient points was presented. Thanks to the adopted bundle perspective, it is possible to compute approximate p-efficient points without losing precision in the solution. The numerical experiments highlight the interest of such inexact oracles, especially when N is large. The need of having a sufficiently large number of realizations to ensure feasibility was also

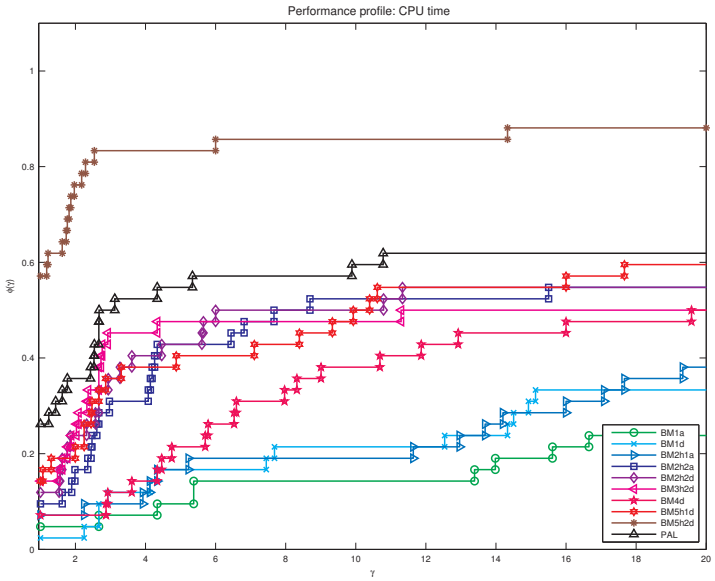

FIG. 6.2. *Performance profiles: CPU time*

demonstrated. As line of future work, possibly escaping the convenient setting (3.3), we mention the development of solvers dealing with oracles that progressively increase N along iterations, all the while using bundle management in order to eliminate previously generated points that are not p-efficient.

Acknowledgments. The first author would like to thank James Luedtke for having provided the PTP instances and discussions regarding probabilistically constrained programming. Part of this work was carried out during the internship of the second author at IMPA, from May 13 to August 2, 2013. The third author gratefully acknowledges financial support provided by Severo Ochoa Program SEV-2013-0323 and Basque Government BERC Program 2014-2017. The fourth author is partially supported by Grants CNPq 303840/2011-0, AFOSR FA9950-11-1-0139, as well as by PRONEX-Optimization and FAPERJ.

REFERENCES

[1] W. VAN ACKOOIJ, *Eventual convexity of chance constrained feasible sets*, Optimization, (2013), pp. 1–22.
[2] W. VAN ACKOOIJ, R. HENRION, A. MÖLLER, AND R. ZORGATI, *Joint chance constrained programming for hydro reservoir management*, Optimization and Engineering, 15 (2014), pp. 509–531.
[3] W. VAN ACKOOIJ AND C. SAGASTIZÁBAL, *Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems*, Siam Journal on Optimization, 24 (2014), pp. 733–765.
[4] M. BERTOCCHI, G. CONSIGLI, AND M.A.H. DEMPSTER (EDS), *Stochastic Optimization Methods in Finance and Energy: New Financial Products and Energy Market Strategies*, International Series in Operations Research and Management Science, Springer, 2012.
[5] J.F. BONNANS, J.C. GILBERT, C. LEMARÉCHAL, AND C. SAGASTIZÁBAL, *Numerical Optimization: Theoretical and Practical Aspects*, Springer-Verlag, 2nd ed., 2006.
[6] G. C. CALAFIORE AND M. C. CAMPI, *Uncertain convex programs: Randomized solutions and confidence levels*, Mathematical Programming, 102 (2005), pp. 25–46.
[7] M. C. CAMPI AND S. GARATTI, *A sampling-and-discarding approach to chance-constrained optimization: Feasibility and optimality*, Journal of Optimization Theory and Applications, 148 (2011), pp. 257–280.
[8] J. CURRIE AND D. I. WILSON, *OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User*, in Foundations of Computer-Aided Process Operations, N. Sahinidis and J. Pinto, eds., Savannah, Georgia, USA, 8–11 January 2012.
[9] D. DENTCHEVA, *Optimisation Models with Probabilistic Constraints. Chapter 4 in [41]*, MPS-SIAM series on optimization, SIAM and MPS, Philadelphia, 2009.
[10] D. DENTCHEVA, B. LAI, AND A. RUSZCZYŃSKI, *Dual methods for probabilistic optimization problems*, Mathematical Methods of Operations Research, 60 (2004), pp. 331–346.
[11] D. DENTCHEVA AND G. MARTINEZ, *Two-stage stochastic optimization problems with stochastic ordering constraints on the recourse*, European Journal of Operational Research, 219 (2012), pp. 1–8.
[12] ———, *Regularization methods for optimization problems with probabilistic constraints*, Math. Programming (series A), 138 (2013), pp. 223–251.

[13] D. DENTCHEVA, A. PRÉKOPA, AND A. RUSZCZYŃSKI, *Concavity and efficient points for discrete distributions in stochastic programming*, Mathematical Programming, 89 (2000), pp. 55–77.

[14] S. LE DIGABEL, *Algorithm 909: Nomad: Nonlinear optimization with the mads algorithm*, ACM Transactions on Mathematical Software, 37 (2011), pp. 1 –15.

[15] E. D. DOLAN AND J. J. MORÉ, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213.

[16] M. GAUDIOSO, G. GIALLOMBARDO, AND G. MIGLIONICO, *An incremental method for solving convex finite min-max problems*, Math. of Oper. Res., 31 (2006).

[17] A. GENZ AND F. BRETZ, *Computation of multivariate normal and t probabilities.*, no. 195 in Lecture Notes in Statistics, Springer, Dordrecht, 2009.

[18] R. HENRION, *Introduction to chance constraint programming*, Tutorial paper for the Stochastic Programming Community HomePage, http://www.wias-berlin.de/people/henrion/publikat.html, (2004).

[19] R. HENRION AND C. STRUGAREK, *Convexity of Chance Constraints with Dependent Random Variables: the use of Copulae. (Chapter 17 in [4])*, Springer New York, 2011.

[20] M. HINTERMÜLLER, *A proximal bundle method based on approximate subgradients*, Computational Optimization and Applications, 20 (2001), pp. 245–266. 10.1023/A:1011259017643.

[21] J.B. HIRIART-URRUTY AND C. LEMARÉCHAL, *Convex Analysis and Minimization Algorithms II*, no. 306 in Grundlehren der mathematischen Wissenschaften, Springer-Verlag, 2nd ed., 1996.

[22] K.C. KIWIEL, *A proximal bundle method with approximate subgradient linearizations*, SIAM Journal on Optimization, 16 (2006), pp. 1007–1023.

[23] M. LEJEUNE AND N. NOYAN, *Mathematical programming approaches for generating p-efficient points*, European Journal of Operational Research, (2010), pp. 590–600.

[24] C. LEMARÉCHAL, *Lagrangian decomposition and nonsmooth optimization: bundle algorithm, prox iteration, augmented lagrangian*, in Nonsmooth optimization methods and applications, F. Giannessi, ed., Gordon & Breach, 1992, pp. 201–216.

[25] J. LUEDTKE, *A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support*, Mathematical Programming, To Appear (2013), pp. 1–26.

[26] J. LUEDTKE AND S. AHMED, *A sample approximation approach for optimization with probabilistic constraints*, SIAM Journal on Optimization, 19 (2008), pp. 674–699.

[27] J. LUEDTKE, S. AHMED, AND G.L. NEMHAUSER, *An integer programming approach for linear programs with probabilistic constraints*, Mathematical Programming, 122 (2010), pp. 247–272.

[28] J. MAYER, *On the Numerical solution of jointly chance constrained problems. Chapter 12 in [42]*, Springer, 1st ed., 2000.

[29] J.J. MOREAU, *Proximité et dualité dans un espace Hilbertien*, Bulletin de la Société Mathématique de France, 93 (1965), pp. 273–299.

[30] D.R. MORGAN, J.W. EHEART, AND A.J. VALOCCHI, *Aquifer remediation design under uncertainty using a new chance constraint programming technique*, Water Resources Research, 29 (1993), pp. 551–561.

[31] W. DE OLIVEIRA AND C. SAGASTIZÁBAL, *Bundle methods in the xxi century: A birds'-eye view*, Pesquisa Operacional, 34 (2014), pp. 647 – 670.

[32] W. DE OLIVEIRA, C. SAGASTIZÁBAL, AND C. LEMARÉCHAL, *Convex proximal bundle methods in depth: a unified analysis for inexact oracles*, Mathematical Programming, 148 (2014), pp. 241–277.

[33] A. PRÉKOPA, *Stochastic Programming*, Kluwer, Dordrecht, 1995.

[34] A. PRÉKOPA, *Probabilistic programming. In [40] (Chapter 5)*, Elsevier, Amsterdam, 2003.

[35] A. PRÉKOPA AND T. SZÁNTAI, *Flood control reservoir system design using stochastic programming*, Math. Programming Study, 9 (1978), pp. 138–151.

[36] ———, *On optimal regulation of a storage level with application to the water level regulation of a lake*, European Journal of Operations Research, 3 (1979), pp. 175–189.

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

[38] R. T. ROCKAFELLAR AND R.J.-B. WETS, *A Lagrangian finite generation technique for solving linear-quadratic problems in stochastic programming*, Mathematical Programming Study, 28 (1986), pp. 63–93.

[39] A. RUSZCZYŃSKI, *Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra.*, Mathematical Programming, 93 (2002), pp. 195–215.

[40] A. RUSZCZYŃSKI AND A. SHAPIRO, *Stochastic Programming*, vol. 10 of Handbooks in Operations Research and Management Science, Elsevier, Amsterdam, 2003.

[41] A. SHAPIRO, D. DENTCHEVA, AND A. RUSZCZYŃSKI, *Lectures on Stochastic Programming. Modeling and Theory*, vol. 9 of MPS-SIAM series on optimization, SIAM and MPS, Philadelphia, 2009.

[42] S. URYAS'EV (ED), *Probabilistic Constrained Optimization: Methodology and Applications*, Kluwer Academic Publishers, 2000.

[43] A.F. VEINOTT, *The supporting hyperplane method for unimodal programming*, Operations Research, 15 (1967), pp. 147–152.

[44] C. WOLF, C. I. FÁBIÁN, A. KOBERSTEIN, AND L. STUHL, *Applying oracles of on-demand accuracy in two-stage stochastic programming a computational study*, European Journal of Operational Research, 239 (2014), pp. 437–448.

[45] S. ZAOURAR AND J. MALICK, *Prices stabilization for inexact unit-commitment problems*, Mathematical Methods of Operations Research, to Appear, 78 (2013), pp. 341–359.

[46] V. ZVEROVICH, C. I. FÁBIÁN, E. F. D. ELLISON, AND G. MITRA, *A computational study of a solver system for processing two-stage stochastic lps with enhanced benders decomposition*, Mathematical Programming Computation, 4 (2012), pp. 211–238.

Appendix: Tables with Results

Benchmark of four different techniques for finding approximate p-efficient points.

			MILP solver		Error (%)				CPU time reduction (%)			
m	*N*	*p*	CPU (s)	d(u)	h2	h3	h4	h1	h2	h3	h4	h1
50	20	0.90	0.2993	141.9615	1.2	0.0	0.0	0.0	3.6	-150.2	-228.0	97.6
50	20	0.95	0.162	143.2774	0.0	0.0	0.0	0.0	4.7	-159.7	-238.8	98.1
50	50	0.90	1.4174	152.6517	2.6	0.0	0.6	0.2	67.7	72.7	5.2	98.0
50	50	0.95	0.7214	155.2201	0.9	0.0	0.0	0.0	40.5	74.8	-64.2	98.5
50	80	0.90	11.5323	154.9331	1.4	0.3	0.3	0.2	93.8	95.9	78.2	99.2
50	80	0.95	1.7683	156.4743	1.4	0.0	0.1	0.0	62.3	80.0	-29.2	98.0
50	100	0.90	17.2153	155.9279	2.0	0.5	0.8	0.5	93.3	95.9	81.1	99.3
50	100	0.95	7.1399	158.0871	2.2	0.1	0.2	0.1	86.0	91.4	54.5	99.1
50	300	0.90	603.3811	157.8947	4.9	0.2	1.0	0.5	99.3	97.7	96.4	99.8
50	300	0.95	445.2641	160.097	3.0	0.1	1.0	0.1	99.1	97.7	95.4	99.9
50	500	0.90	607.1419	158.2272	6.2	1.5	2.5	0.9	98.7	98.1	91.2	99.4
50	500	0.95	607.1014	161.375	6.2	0.5	1.5	0.5	98.7	97.2	91.1	99.6
50	1000	0.90	622.8591	158.6351	5.7	6.3	4.0	0.3	95.9	96.9	60.5	96.0
50	1000	0.95	622.3926	161.9426	6.6	4.1	2.5	0.5	95.9	96.9	63.9	97.9
50	2000	0.90	679.3827	158.463	9.5	9.0	6.2	1.9	86.6	91.5	10.8	72.8
50	2000	0.95	679.3481	162.0196	6.0	6.6	4.2	1.1	86.7	91.7	10.6	86.0
100	20	0.90	0.5159	268.65	1.0	0.0	0.0	0.0	42.9	-112.2	3.2	99.6
100	20	0.95	0.2906	270.1507	0.0	0.0	0.0	0.0	0.8	-90.5	-64.5	99.4
100	50	0.90	5.3918	281.8679	1.0	0.0	0.1	0.0	84.4	90.2	80.3	99.6
100	50	0.95	1.3536	285.2036	0.5	0.0	0.0	0.0	38.0	81.2	21.2	99.4
100	80	0.90	27.0308	287.0593	1.5	0.1	0.1	0.1	94.4	95.7	91.6	99.8
100	80	0.95	5.608	290.0021	0.5	0.0	0.0	0.0	73.4	86.8	60.2	99.5
100	100	0.90	84.1459	290.2488	1.2	0.3	0.1	0.0	97.5	98.1	96.3	99.9
100	100	0.95	17.0205	293.2395	0.8	0.0	0.0	0.0	87.8	94.2	81.8	99.7
100	300	0.90	609.9811	297.0122	3.9	0.7	0.5	0.2	98.3	98.2	96.6	99.7
100	300	0.95	609.7378	301.4066	2.8	0.1	0.5	0.1	98.3	98.2	96.6	99.9
100	500	0.90	623.2317	299.7205	5.5	2.7	2.2	0.3	96.0	97.3	90.2	99.1
100	500	0.95	622.8708	306.2347	2.6	1.4	1.3	0.4	96.1	97.2	91.3	99.5
100	1000	0.90	682.8977	302.7364	6.3	6.2	2.6	1.3	87.5	96.9	53.3	94.1
100	1000	0.95	689.2382	309.6351	4.0	3.8	1.1	0.3	87.7	96.9	56.4	96.8
100	2000	0.90	631.6813	305.7674	7.5	8.8	5.4	0.8	9.7	91.0	-1.7	0.8
100	2000	0.90	620.3487	312.2142	5.3	6.6	3.9	1.3	5.7	91.2	-1.5	48.3

Problem Ain48.

	N	BM1a	BM1d	BM2h1a	BM2h2a	BM2h2d	BM3h2d	BM4d	BM5h1d	BM5h2d	PAL
Gap (%)	50	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.00
	100	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
	250	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
	500	-	0.01	-	0.01	0.01	0.01	0.01	0.01	0.01	0.00
	1000	-	-	-	-	-	-	-	-	0.01	-
	2000	-	-	-	-	-	-	-	-	-	0.00
Infeas	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	100	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
	250	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
	500	-	0.05	-	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	1000	-	-	-	-	-	-	-	-	0.03	-
	2000	-	-	-	-	-	-	-	-	-	0.02
CPU (min)	50	0.14	0.11	0.11	0.04	0.04	0.04	0.13	0.06	0.04	0.03
	100	1.14	0.54	0.59	0.18	0.19	0.13	0.56	0.46	0.11	0.04
	250	77.89	34.83	18.36	19.59	12.03	12.76	30.84	11.56	9.21	4.68
	500	-	2102.14	-	676.79	635.04	514.24	1787.21	669.79	497.91	273.69
	1000	-	-	-	-	-	-	-	-	490.08	-
	2000	-	-	-	-	-	-	-	-	-	300.21

Problem CM.

	N	BM1a	BM1d	BM2h1a	BM2h2a	BM2h2d	BM3h2d	BM4d	BM5h1d	BM5h2d	PAL
Gap (%)	50	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-1.81	-0.00	-0.00
	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-1.96	-1.99	0.00
	250	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.00	0.00	0.01
	500	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.47	0.00
	1000	0.00	0.00	0.00	0.00	0.00	0.00	-1.24	0.00	0.00	-0.08
	2000	0.01	0.01	0.00	-0.01	0.01	0.01	-0.06	0.01	-0.12	0.01
Infeas	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	250	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	500	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1000	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2000	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CPU (min)	50	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	100	0.01	0.03	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	250	0.04	0.04	0.04	0.03	0.03	0.01	0.04	0.03	0.01	0.03
	500	0.68	0.13	0.23	0.38	0.08	0.08	0.13	0.14	0.04	0.08
	1000	6.18	3.93	2.13	0.33	0.36	0.26	0.88	1.14	0.11	0.34
	2000	321.49	124.26	221.86	27.66	46.01	94.94	23.49	8.11	4.06	43.78

Problem Isr168.

	N	BM1a	BM1d	BM2h1a	BM2h2a	BM2h2d	BM3h2d	BM4d	BM5h1d	BM5h2d	PAL
Gap (%)	50	0.01	-	0.01	0.01	0.00	0.00	0.01	0.01	0.01	0.00
	100	0.01	-	0.01	0.01	-	-	-	-	-	0.00
	250	-	-	-	0.01	0.01	0.01	-	0.01	-	0.00
	500	-	-	-	-	-	-	-	-	-0.00	0.00
	1000	-	-	-	-	-	-	-	-	-0.00	-
	2000	-	-	-	-	-	-	-	-	-	-
Infeas	50	0.27	-	0.28	0.28	0.80	0.80	0.28	0.27	0.27	0.27
	100	0.07	-	0.07	0.06	-	-	-	-	-	0.07
	250	-	-	-	0.05	0.05	0.05	-	0.05	-	0.05
	500	-	-	-	-	-	-	-	-	0.03	0.03
	1000	-	-	-	-	-	-	-	-	0.02	-
	2000	-	-	-	-	-	-	-	-	-	-
CPU (min)	50	0.39	-	0.16	0.08	0.01	0.03	0.36	0.18	0.14	0.04
	100	10.86	-	9.04	1.83	-	-	-	-	-	0.78
	250	-	-	-	1418.68	906.84	579.66	-	619.68	-	717.96
	500	-	-	-	-	-	-	-	-	71.98	1742.29
	1000	-	-	-	-	-	-	-	-	92.53	-
	2000	-	-	-	-	-	-	-	-	-	-

Problem Isr96.

	N	BM1a	BM1d	BM2h1a	BM2h2a	BM2h2d	BM3h2d	BM4d	BM5h1d	BM5h2d	PAL
Gap (%)	50	0.01	0.01	0.01	0.00	0.00	0.00	0.01	0.01	0.01	0.00
	100	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.00
	250	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.00
	500	-	-	-	0.01	0.01	0.01	-	0.01	0.01	0.00
	1000	-	-	-	-	0.01	-	-	-	0.01	-
	2000	-	-	-	-	-	-	-	-	-	-
Infeas	50	0.13	0.13	0.13	0.13	0.14	0.14	0.14	0.13	0.14	0.13
	100	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
	250	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
	500	-	-	-	0.00	0.00	0.00	-	0.00	0.00	0.00
	1000	-	-	-	-	0.01	-	-	-	0.01	-
	2000	-	-	-	-	-	-	-	-	-	-
CPU (min)	50	0.23	0.14	0.18	0.04	0.06	0.04	0.16	0.09	0.06	0.01
	100	1.03	1.16	1.31	0.33	0.28	0.16	0.91	0.64	0.14	0.08
	250	147.54	71.71	70.19	20.46	14.53	11.49	52.84	24.11	11.31	4.94
	500	-	-	-	764.84	542.23	436.03	-	838.48	292.91	783.68
	1000	-	-	-	-	814.09	-	-	-	145.16	-
	2000	-	-	-	-	-	-	-	-	-	-

Problem PTP1.

	N	BM1a	BM1d	BM2h1a	BM2h2a	BM2h2d	BM3h2d	BM4d	BM5h1d	BM5h2d	PAL
Gap (%)	50	-0.06	0.01	-0.06	-0.03	-0.05	-0.05	-0.84	-0.30	-0.84	0.00
	100	0.00	-0.01	0.00	0.00	-0.01	-0.01	-0.98	-0.91	-0.98	0.00
	250	-	-	-	-	-	-	-0.40	0.10	-1.00	0.00
	500	-	-	-	-	-	-	-1.27	-0.89	-1.27	0.00
	1000	-	-	-	-	-	-	-0.86	-0.02	-1.74	0.00
	2000	-	-	-	-	-	-0.00	-	-	-0.72	0.00
Infeas	50	0.61	0.60	0.61	0.61	0.61	0.61	0.60	0.60	0.60	0.60
	100	0.40	0.41	0.40	0.40	0.41	0.41	0.39	0.40	0.39	0.40
	250	-	-	-	-	-	-	0.14	0.21	0.20	0.23
	500	-	-	-	-	-	-	0.08	0.13	0.08	0.14
	1000	-	-	-	-	-	-	0.01	0.09	0.08	0.09
	2000	-	-	-	-	-	0.06	-	-	0.03	0.06
CPU (min)	50	10.58	11.13	2.26	2.13	2.84	1.83	0.28	0.08	0.09	0.11
	100	168.01	189.84	97.89	39.24	30.34	18.01	2.94	0.43	0.33	0.83
	250	-	-	-	-	-	-	10.73	7.09	0.44	180.04
	500	-	-	-	-	-	-	114.04	127.26	0.88	180.04
	1000	-	-	-	-	-	-	827.98	213.49	4.41	180.06
	2000	-	-	-	-	-	1741.03	-	-	18.21	180.09