DECOMPOSITION OF 2–STAGE STOCHASTIC INTEGER PROGRAMS: EXPERIMENTAL COMPARISON OF DUAL SOLUTION METHODS

ROBIN VUJANIC, PEYMAN MOHAJERIN ESFAHANI

1. Introduction

Two–stage stochastic programs encapsulate decision making processes under uncertainty in which part of the decisions have to be taken before uncertainty is revealed, and another part after. In many practical applications, some of the decisions are discrete in nature (on or off, buy or not buy, etc.), leading to so–called stochastic integer programs [11, 14]. While there is substantial practical interest in these models, one of the main limiting factors are difficulties associated with solving large instances. The size of the corresponding optimization programs grows with the number of scenarios, and when a large number of them is necessary to effectively represent the desired effects of uncertainty, or when the number of stages considered in the decision process increases, models quickly escape the solution ability of state–of–the–art general purpose solvers.

One approach extensively studied in the literature is to apply the Lagrangian relaxation framework [19, 17, 22] to decompose structured optimization programs, such as multistage stochastic integer problems, in smaller subproblems [14]. When the original problem is non–convex, this procedure is generally unable to guarantee that an optimizer is found [7]. Despite this, several solution and approximation algorithms for stochastic integer programs rely on duality. Dual decomposition combined with Branch and Bound is common [14, 8] and is used in some practical contexts; for example, in [13], the authors apply it to stochastic unit commitment in the power sector, while in the process industry, [33] applies it to the scheduling problem of a multi product batch plant with uncertainties. Other algorithms include progressive hedging [30] and augmented Lagrangian methods in the convex case [24]. There are also several different ways to decompose the same problem; scenario decomposition is common [14, 20], while the authors of [31] investigate node decomposition, and [15] compares it with geographical decomposition.

The central step of these algorithms involves solving the so–called dual problem associated to the original optimization program. While the dual problem is convex, independently of whether the primal problem is convex or not, it is non–differentiable by construction [REF]. This impedes the application of several methods that require smoothness to function; in [9, Sec. 2.1.3] the author presents a simple non–differentiable convex program and uses it to construct a counterexample showing how a traditional gradient method with backtracking fails to converge to an optimizer, even though the method itself never encounters a point in which the function is non-differentiable.

An extensive number of methods have been proposed to address the difficulties associated with non–differentiability. The relative performance of many of them, especially the more recent ones, is however largely unknown. In this paper we aim to address this lack of experimental evidence. Specifically, we have implemented and deployed 10 different methods to solve the dual problem associated to 2 stage stochastic integer programs. Our selection ranges from traditional subgradient [34, 12] and cutting planes [22, 7] methods, to state–of–the–art bundle methods [23, 6, 32] and beyond, including quasi monotone [27] and universal [26] subgradient methods. To the best of our knowledge, the use of quasi monotone and universal methods has not been reported yet for the application domain of decision processes under uncertainty. We make all our implementations publicly available at [REF]for the benefit of the community.

We perform extensive experimental performance comparisons on several datasets publicly available at [1]. These reflect a broad range of use cases, since they entail instances of various dimensions and structures. Some of these datasets are synthetic, some other originate from practice.

Additionally, the novel quasi monotone [27] and universal [26] subgradient methods require the solution of certain auxiliary optimization programs. For the application discussed in this paper, however, we show that these optimizations can be solved analytically, leading to a significant simplification of the computations required for each iteration.

Structure of the paper. In the next section we introduce the 2 stage stochastic integer programming model we consider in this paper, and discuss the application of Lagrangian duality for its decomposition. In Section 3 we introduce the methods we have implemented for our experiments, and for the more advanced ones, also provide the corresponding pseudo codes. In Section 4 we first present the setup and procedure used in our experiments, then discuss the results we obtained. In Section 5 we draw the conclusions of the paper.

2. Problem Statement and Dual Decomposition Approach

We consider two-stage stochastic optimization problems in the form

$$\begin{array}{rlll} p^\star \doteq & \min\limits_{x,y_s} & \langle c,x\rangle + \sum\limits_{s\in S} p_s\langle q_s,y_s\rangle & \\ & \text{s.t.} & Ax \le b & \\ & & T_s x + W_s y_s \le h_s & \forall s\in S \\ & & x\in X & \\ & & y_s \in Y_s & \forall s\in S. \end{array} \tag{$\mathcal{P}$}$$

We refer to x and y_s as first and second stage decisions respectively, where $s \in S$ indexes the scenarios, and $p_s \in R$ is the probability of a scenario so that $p_s \ge 0$ and $\sum_{s\in S} p_s = 1$. We assume that the set

$$\overline{X} \doteq X \cap \{x \in R^{n_x} |\ Ax \le b\}, \quad X = Z^{n_x^z} \times R^{n_x^r}, \tag{1}$$

and the feasible set of $\mathcal{P}$ are non-empty and compact sets, and that the sets $Y_s, s \in S$ can be written as $Y_s = \left\{ y_s \in Z^{n_y^z} \times R^{n_y^r} \mid C_s y_s \le d_s \right\}$ for given matrices C_s and vectors d_s of appropriate dimensions.

To allow computations on instances entailing a large number of scenarios, approaches based on dual decomposition have been proposed which separate $\mathcal{P}$ into a set of smaller optimization problems, see, e.g., [14, 15]. In this paper we follow the traditional approach introduced in [14], in which copies of first stage variables are introduced and the resulting copy constraints are relaxed[1]. We first write the following optimization program equivalent to $\mathcal{P}$:

$$\begin{array}{lll} \min\limits_{x,y_s} & \sum\limits_{s\in S} p_s \cdot (\langle c,x_s\rangle + \langle q_s,y_s\rangle) & \\ \text{s.t.} & x_s = x & \forall s\in S \\ & Ax_s \le b & \forall s\in S \\ & T_s x_s + W_s y_s \le h_s & \forall s\in S \\ & x_s \in X, y_s \in Y_s & \forall s\in S, \end{array} \tag{2}$$

where we have introduced $|S|$ copies of x. Note that the initial x is still in the problem, and except for the equality copy–constraints it is now unconstrained and does not enter in the objective function. Problem (2)

[1] Several decomposition strategies exist whereby, for example, scenarios are grouped into clusters, and the subproblems correspond to these clusters. Here we generate one subproblem for each scenario.

can be decomposed by dualizing the copy constraints, leading to the dual function

$$\begin{array}{rcll} d(\lambda) & \doteq & \min\limits_{x,x_s,y_s} & \sum\limits_{s\in S}\left[p_s\cdot(\langle c+\lambda_s,x_s\rangle+\langle q_s,y_s\rangle)\right]-\sum\limits_{s\in S}\langle\lambda_s,x\rangle \\ & & \text{s.t.} & Ax_s\le b & \forall s\in S \\ & & & T_sx+W_sy_s\le h_s & \forall s\in S \\ & & & x_s\in X, y_s\in Y_s & \forall s\in S. \end{array} \tag{3}$$

Since x is unbounded in (3), the only way for $d(\lambda)$ to remain bounded is to ensure that

$$\sum_{s\in S}\langle\lambda_s,x\rangle=\langle x,\sum_{s\in S}\lambda_s\rangle=0 \quad \forall x\in R^{n_x} \quad \Rightarrow \quad \sum_{s\in S}\lambda_s=0. \tag{4}$$

The resulting dual problem is therefore

$$d^\star\doteq\max_{\lambda\in\Lambda}\ d(\lambda), \tag{$\mathcal{D}$}$$

with

$$\Lambda\doteq\left\{\lambda\in R^{n_\lambda}\ \middle|\ \sum_{s\in S}\lambda_s=0^{n_x}\right\}, \tag{5}$$

where $n_\lambda=|S|\cdot n_x$. For a given $\lambda\in R^{n_\lambda}\in\Lambda$, we indicate with the tuple $(x(\lambda),x_s(\lambda),y_s(\lambda))$ an optimizer to the inner problem in (3), while $\Lambda^\star\subseteq\Lambda$ is the set of optimizers to $\mathcal{D}$.

Remark 1 (Well-posedness)**.** *A tuple $(x(\lambda),x_s(\lambda),y_s(\lambda))$ exists for any $\lambda\in\Lambda$, and the set $\Lambda^\star$ is non-empty.*

This follows from the assumption that the feasible set of $\mathcal{P}$ is non-empty and compact.

Note that, for any $\lambda\in\Lambda$, $x(\lambda)$ can be set to any arbitrary point in R^{n_x} since it is unconstrained and does not influence the objective. In particular, we are allowed to pick $x(\lambda)=\frac{1}{|S|}\sum_{s\in S}x_s(\lambda)$. This choice dictates how we compute the residual when we want to determine a subgradient of the dual function.

Remark 2 (Lemma 4.5, [32])**.** *Given $f:R^{n_x}\to R$, let $\nabla f(x):R^{n_x}\to R^{n_x}$ be a subgradient of f at x. Then, for $\lambda\in\Lambda$, a subgradient of $d(\lambda)$ is given by $\nabla d(\lambda)=[x_s(\lambda)-x(\lambda)]_{s\in S}$.*

We can also provide a–priori upper bounds to the norm of dual optimizers and subgradients of $d(\lambda)$.

Proposition 1 (Bound on dual optimizers for equality constraints)**.** *Consider the following optimization problem and its dual counterpart*

$$\widehat{\mathcal{P}}:\begin{cases}\widehat{p}\doteq & \min\limits_{x\in\widehat{X}} & f(x)\\ & \text{s.t.} & Hx=b,\end{cases}\qquad \widehat{\mathcal{D}}:\widehat{d}\doteq\max_{\lambda}\langle b,\lambda\in\Lambda\rangle+\left\{\min_{x\in\widehat{X}}f(x)-\langle Hx,\lambda\rangle\right\}.$$

Suppose that $\widehat{\mathcal{P}}$ is feasible, and there exist constants $\widehat{d}_{\text{lb}}$, $\widehat{p}^{\max}_{\text{ub}}$, and γ such that

$$\begin{cases}\widehat{p}^{\max}_{\text{ub}}\ge\max\limits_{x\in\widehat{X}}f(x)\ge\widehat{d}\ge\widehat{d}_{\text{lb}},\\ \min\limits_{\lambda\in\Lambda}\max\limits_{x\in\widehat{X}}\frac{\langle Hx,\lambda\rangle}{\|\lambda\|}\ge\gamma>\|b\|.\end{cases} \tag{6}$$

Then, any optimizer of the dual program achieving the optimal value $\widehat{d}$ satisfies the bound $\|\lambda^\star\|\le\frac{\widehat{p}^{\max}_{\text{ub}}-\widehat{d}_{\text{lb}}}{\gamma-\|b\|}$. In particular, if $\text{conv}(\widehat{X})$ contains a ball of radius ρ (i.e., $\{x\in\mathbb{R}^n:\|x\|\le\rho\}\subset\text{conv}\,(\widehat{X})$), then

$$\|\lambda^\star\|\le\frac{\widehat{p}^{\max}_{\text{ub}}-\widehat{d}_{\text{lb}}}{\rho\sigma_{\min}(H)-\|b\|},$$

where $\sigma_{\min}(H)$ is the minimum non-zero singular value of the matrix H.

Note that we do not require zero duality gap between $\widehat{\mathcal{P}}$ and $\widehat{\mathcal{D}}$.

Proof. Let $\lambda^\star$ be an optimizer of $\widehat{\mathcal{D}}$. Based on the assumptions, observe that

$$\begin{aligned}\widehat{d}_{\text{lb}} \leq \widehat{d} &= \langle b, \lambda^\star \rangle + \Big\{ \min_{x \in \widehat{X}} f(x) - \langle Hx, \lambda^\star \rangle \Big\} \\ &\leq \|b\| \, \|\lambda^\star\| + \widehat{p}_{\text{ub}}^{\max} - \max_{x \in \widehat{X}} \langle Hx, \lambda^\star \rangle \leq \|b\| \, \|\lambda^\star\| + \widehat{p}_{\text{ub}}^{\max} - \gamma \|\lambda^\star\| = \widehat{p}_{\text{ub}}^{\max} - (\gamma - \|b\|) \|\lambda^\star\|,\end{aligned}$$

where the first and second inequalities are concluded based on the first and second assumptions in (6), respectively. Thus, the first assertion of the proposition follows immediately from the above relation.

Concerning the second part, it suffices to show that the parameter γ in (6) is bounded by $\rho \sigma_{\min}(H)$. To that end, it is straightforward to see that

$$\max_{x \in \widehat{X}} \langle Hx, \lambda^\star \rangle = \max_{x \in \widehat{X}} \langle x, H^\top \lambda^\star \rangle = \max_{x \in \text{conv}(\widehat{X})} \langle x, H^\top \lambda^\star \rangle \geq \max_{\|x\| \leq \rho} \langle x, H^\top \lambda^\star \rangle \geq \rho \|H^\top \lambda^\star\| \geq \rho \sigma_{\min}(H) \|\lambda^\star\|,$$

which concludes the proof. □

A related result has been proposed for problems in which inequalities are dualized, see [25, Lemma 1]. This result requires the existence of a strictly feasible point (i.e., the dualized constraint is satisfied with strict inequality), and therefore cannot be applied to (2), where we dualize equalities. It is, however, worth noting that the suggested bounds in both cases share similar the geometrical feature reflected through a "*deep*" feasible point. In the next step we validate the required conditions in Proposition 1 for the two-stage stochastic program $\mathcal{P}$.

In the next step, we validate the required conditions of Proposition 1 in the case of two-stage stochastic program $\mathcal{P}$ with the respective dual program $\mathcal{D}$. A key object to quantify the proposed bound is to suggest a *relative* interior ball with the radius ρ. Note that the relevant space is the one involved in the dualized constraint is the first stage variable x. Then, by virtue of Proposition 1, it suffices to investigate a strict feasible point in regard to the first stage variable while the second stage decision is incorporated into the objective function.

Lemma 3. *Optimizers of the dual program $\mathcal{D}$ satisfy the following relation:*

$$\|\lambda^\star\| \leq \frac{p_{\text{ub}}^{\max} - d_{\text{lb}}}{\rho} \doteq R_{\mathcal{D}} \tag{7}$$

where

$$\begin{aligned} p_{\text{ub}}^{\max} \geq \max_{x} \min_{y_s \in Y} \quad & \langle c, x \rangle + \sum_{s \in S} p_s \cdot \langle q_s, y_s \rangle & \\ \text{s.t.} \quad & Ax \leq b & \\ & T_s x + W_s y_s \leq h_s & \forall s \in S \\ & x \in X & \\ & y_s \in Y_s & \forall s \in S, \end{aligned} \tag{8}$$

d_{lb} is any lower bound to $d^\star$, e.g. $d(0)$, and ρ is the radius of a ball contained in the relative interior of $\text{conv}(\overline{X})$.

Proof. First note that for the special case of (2), $\|b\| = 0$ and $\sigma_{\min}(H) = 1$. Then, observe that program (2) fits $\widehat{\mathcal{P}}$ with

$$\begin{aligned} f(x) \doteq \min_{y_s, s \in S} \quad & \textstyle\sum_{s \in S} p_s \langle q_s, y_s \rangle \\ \text{s.t.} \quad & T_s x + W_s y_s \leq h_s \\ & y_s \in Y_s, \end{aligned} \tag{9}$$

and $\widehat{X} \doteq \text{conv}(\overline{X})$; thus ρ is the radius of a ball in the relative interior of $\text{conv}(\overline{X})$.

□

Given the

Fact 4. *For any $\lambda \in \Lambda$, we have that $\|\nabla d(\lambda)\| \leq D(\overline{X})$, where $\overline{X}$ is as in (1), and the diameter of $\overline{X}$ is*

$$D(\overline{X}) \doteq \begin{array}{ll} \max & \|x - y\| \\ \text{s.t.} & x, y \in \overline{X}. \end{array} \tag{10}$$

Since $\mathcal{P}$ is not convex, we generally have that $d^\star < p^\star$ (duality gap). This implies that $\mathcal{P}$ and $\mathcal{D}$ are not equivalent and that, in general, we cannot recover an optimal (sometimes even feasible) solution to $\mathcal{P}$ from $(x(\lambda^\star), x_s(\lambda^\star), y_s(\lambda^\star))$, for $\lambda^\star \in \Lambda^\star$ [7]. Solving the dual problem is still of significant practical interest since, as mentioned in the introduction, a number of solution and approximation algorithms rely on it. One challenging aspect is that the dual problem is, by construction, non–differentiable, so that appropriate methods have to be deployed for solving it. In the next section we introduce the algorithms that we have implemented to solve $\mathcal{D}$.

3. Solution Methods Considered

To solve $\mathcal{D}$, we consider iterative methods which make calls to a so–called first–order oracle of the problem [29]. For a given iterate λ_k, a first–order oracle returns the value of $d(\lambda_k)$, by solving the inner problem in (3), and a subgradient $\nabla d(\lambda_k)$ which can be computed according to Remark 2. In our application, the dual space Λ is constrained (5), so the oracle also has to provide an appropriate projection function $\Pi_\Lambda : R^{n_\lambda} \to R^{n_\lambda}$. All methods are initialized at $\lambda_0 = \Pi_\Lambda(0^{n_\lambda})$, where 0^{n_λ} indicates the vector of 0's in R^{n_λ}.

We have selected and implemented 10 different first–order methods to solve $\mathcal{D}$. We describe them below briefly and provide revelevant references to the papers that introduced them and that contain their convergence properties. We also provide all the pseudo codes.

3.1. Subgradient Methods.

We first include two variants of the traditional projected subgradient method [34]:

$$\lambda_{k+1} = \Pi_\Lambda(\lambda_k + s_k \nabla d(\lambda_k)), \tag{11}$$

one with constant stepsize $s_k = s_0$, and the other with decaying stepsize $s_k = s_0/(k+1)$. They are indicated by "SG const" and "SG $1/k$" in the experiments. The main convergence properties of these methods are explained in [7, 32, 12]; additional properties of subgradient optimization when used in a duality context are reported in [4]. For the application of 2–stage stochastic programming, we can derive the following result.

Theorem 5 (Convergence Rate of Subgradient Methods [12])**.** *The sequence $\{\lambda_k\}$ generated by the projected subgradient method (11) applied to problem $\mathcal{D}$ with initialization $\lambda_0 = \Pi_\Lambda(0^{n_\lambda})$ satisfies the following relation:*

$$d(\lambda^\star) - d^{\text{best}}(\lambda_k) \leq \frac{R_{\mathcal{D}} + D(\overline{X})^2 \sum_{t=1}^k s_t^2}{2 \sum_{t=1}^k s_t}. \tag{12}$$

Proof. According to [12, p. 3], we generally have that

$$d(\lambda^\star) - d^{\text{best}}(\lambda_k) \leq \frac{R^2 + G^2 \sum_{t=1}^k s_t^2}{2 \sum_{t=1}^k s_t}, \tag{13}$$

where $\|\lambda_0 - \lambda^\star\| \leq R$, for all $\lambda^\star \in \Lambda^\star$ and $\|\nabla d(\lambda)\| \leq G$. In our specific case, according to Fact 4, $G = D(\overline{X})$. Furthermore, $\Pi_\Lambda(0^{n_\lambda}) = 0^{n_\lambda}$ for Λ as given in (5) and hence, $R = R_{\mathcal{D}}$ as defined in Lemma 3. □

3.2. **Cutting Planes and Bundle Methods.** Cutting Planes (**CP**) methods construct a piecewise approximation of the dual function (3) and find its optimum by solving an appropriate LP [32, Fig. 7.6] (ref appendix instead?). Under our assumptions on $\mathcal{P}$, $d(\lambda)$ is a piecewise affine function with a finite number of segments and CP methods are guaranteed to converge to an optimal solution in a finite number of iterations [32, Thm. 7.8, Example 7.23].

One of the major drawbacks of CP methods is that the size of the LP to be solved increases every iteration as the model of the dual function $d(\lambda)$ is improved; they also produce "unstable" iterates, i.e., points whose distance is large from the previous position even after many iterations. Bundle methods [32, Sec. 7.4] were introduced to alleviate these problems. They keep a restricted "bundle" of planes as a description of $d(\lambda)$, which is updated every iterations, and use a quadratic regularization term to stabilize iterates. This method is also guaranteed to converge to an optimal solution in finitely many steps [32, Lem. 7.18]

We have implemented a version of these methods by adapting from [6, p.21] and [32, p.374]. Pseudo code reflecting our implementation is in (TOT). CP requires an initial bounded set in which to perform the search. We select a bounded box to further constrain Λ to achieve this. It also accepts the required optimality tolerance ϵ as a parameter. The bundle algorithm, on the other hand, entails three main tuning parameters: the optimality tolerance $\epsilon > 0$, the value of $\gamma \in (0, 1)$ which sets the trade–off between stabilizing iterates and making progress by updating the center $\hat{\lambda}_k$, and $\mu > 0$, which parametrizes the quadratic regularization term.

Bundle Method, adapted from [6, p.21] and [32, p.374]

Initialization.

- Choose $\epsilon > 0$ (optimality tolerance), $\gamma \in (0, 1)$, $\mu > 0$.
- Let $\hat{\lambda}_0 \in \Pi_\Lambda(0^{n_\lambda})$ and set $\lambda_0 = \hat{\lambda}_0$.
- Query oracle to determine $d(\hat{\lambda}_0)$ and $\nabla d(\hat{\lambda}_0)$.

Iterations. For $k \geq 0$:

(1) Compute

$$\lambda_{k+1} \in \arg \min_{\lambda \in \Lambda, r \in R} \; r + \tfrac{\mu}{2}\|\lambda - \hat{\lambda}_k\|^2 \quad \text{s.t.} \quad r \geq -d(\lambda_\ell) - \langle \nabla d(\lambda_\ell), \lambda - \lambda_\ell \rangle, \quad \ell = 0, \ldots, k \tag{14}$$

and set $d_k^\star$ the optimal value of (14).

(2) Define

$$\epsilon_k = d_k^\star - d(\hat{\lambda}_k) \geq 0. \tag{15}$$

(3) If $\epsilon_k < \epsilon$, Stop.

(4) Query oracle to determine $d(\lambda_{k+1})$ and $\nabla d(\lambda_{k+1})$.

(5) If $d(\lambda_{k+1}) - d(\hat{\lambda}_k) \geq \gamma \epsilon_k$, Serious Step: $\hat{\lambda}_{k+1} = \lambda_{k+1}$. Else Null Step: $\hat{\lambda}_{k+1} = \hat{\lambda}_k$.

(6) Set $k = k + 1$, and go to Step 1.

3.3. **Quasi–Monotone Subgradient Methods.** Nesterov et al. [27] recently proposed a number of new subgradient methods for solving non-smooth convex optimization problems such as $\mathcal{D}$. They are relatively simple to implement and, as shown in our experiments, they are also very competitive in terms of convergence performance.

We have implemented the Subgradient Method with Double Simple Averaging [27, Eq. 34], which we refer to as **DSA**, and two variants of the Subgradient Method with Triple Averaging [27, Eq. 43], which are referred to as **TA 1** and **TA 2** respectively. Pseudo code is provided in TOT. The difference between TA 1 and TA 2 is in the way the parameters a_t and γ_t are updated. More variants could be generated, as these parameters could also be updated according to TOT SEE STEFAN EMAIL. In the experiments, the main parameter chosen for tuning these methods is the multiplicative factor γ.

Similarly to CP and Bundle methods, these methods generally require the solution of additional convex optimization programs at each iteration; see, e.g., Step 1 in [27, Eq. 34]. Because of the specific structure of $\mathcal{D}$, we can find an analytic solution to these programs and hence reduce these optimizations to a single algebraic passage; see Appendix TOT.

Theorem 6. *Let the sequence $\{\lambda_k\}$ be generated by DSA according to Algorithm TOT. Then,*

$$d^\star - d(\lambda_k) \leq \frac{1}{\sqrt{k+1}} \left(\gamma \frac{1}{2} (R_{\mathcal{D}} + R)^2 + 1/\gamma \cdot D(\overline{X})^2 \right) - D(\overline{X})R. \tag{16}$$

Proof. From [27, Corollary 3.4], we have that

$$d^\star - d(\lambda_k) + \|s_k\|_R^* \leq \frac{1}{\sqrt{k+1}} \left(\gamma G_R + 1/\gamma \cdot G^2 \right) \tag{17}$$

where $\|\nabla d(\lambda)\| \leq G$, $s_k = -\frac{1}{k+1} \sum_{t=0}^{k} \nabla d(\lambda_k)$, and

$$\|s\|_R^* = \max_{\lambda \in \Lambda} \langle s, \lambda^\star - \lambda \rangle, \quad \text{s.t.} \|\lambda - \lambda^\star\| \leq R$$
$$G_R = \max_{\lambda \in \Lambda} \frac{1}{2} \|\lambda\|^2, \quad \text{s.t.} \|\lambda - \lambda^\star\| \leq R,$$

in which R =???. Observe that

$$\max_{\lambda \in \Lambda, \|\lambda - \lambda^\star\| \leq R} \langle s, \lambda - \lambda^\star \rangle \leq \|s\| R \leq \|\nabla d(\lambda)\| R \leq D(\overline{X}) R, \tag{18}$$

and $G_R \leq \frac{1}{2}(R_{\mathcal{D}} + R)^2$. The desired result follows from substituting these quantities. □

Subgradient with Double Simple Averaging, by Nesterov et al. [27, Eq. 34]

Initialization.

- Choose $\gamma > 0$. Ideal setting is $\gamma = \frac{L}{\sqrt{|d(\lambda^\star)|}}$, where $L \geq \|\nabla d(\lambda)\|$, $\forall \lambda \in \Lambda$, see [27, Eq. 6] and discussion below [27, Eq. 13].
- Let $\lambda_0 \in \Pi_\Lambda(0^{n_\lambda})$.

Iterations. For $k \geq 0$:

(1) Compute

$$\lambda_k^+ = \Pi_\Lambda \left(\frac{1}{\gamma \sqrt{k+1}} \cdot \sum_{t=0}^{k} \nabla d(\lambda_k) \right). \tag{19}$$

(2) Next iterate is

$$\lambda_{k+1} = \frac{k+1}{k+2} \lambda_k + \frac{1}{k+2} \lambda_k^+. \tag{20}$$

Subgradient with Triple Averaging, by Nesterov et al. [27, Eq. 43]

Initialization.

- Let $\lambda_0 \in \Pi_\Lambda(0^{n_\lambda})$.
- For **TA 1**, see [27, Eq. 38] and paragraph before [27, Eq. 34].
  - $a_k = 1$
  - $\gamma_k = \gamma\sqrt{k+1},\ k \geq 0$
- For **TA 2**, see end of [27, Sect. 3].
  - $a_k = k$
  - $\gamma_k = \gamma(k+1)^{3/2},\ k \geq 0$
- Choose $\gamma > 0$. Ideal setting is $\gamma = \frac{L}{\sqrt{|d(\lambda^\star)|}}$, where $L \geq \|\nabla d(\lambda)\|,\ \forall \lambda \in \Lambda$, see [27, Eq. 6] and discussion below [27, Eq. 13].

Iterations. For $k \geq 0$:

(1) Compute

$$\lambda_k^+ = \Pi_\Lambda\left(\frac{1}{\gamma_k}\sum_{t=0}^{k} a_t \cdot \nabla d(\lambda_t)\right). \tag{21}$$

(2) Let $A_k = \sum_{t=0}^{k} a_t$, and

$$\hat{\lambda}_k = \frac{\gamma_k}{\gamma_{k+1}}\lambda_k^+ + \left(1 - \frac{\gamma_k}{\gamma_{k+1}}\right)\lambda_0, \tag{22}$$

$$\tau_k = \frac{a_{k+1}}{A_{k+1}}. \tag{23}$$

(3) Next iterate is

$$\lambda_{k+1} = (1-\tau_k)\lambda_k + \tau_k\hat{\lambda}_k. \tag{24}$$

3.4. **Universal Gradient Methods.** Universal Gradient Methods have been recently proposed by Nesterov in [26]. We have implemented all the variants proposed, namely the Universal Primal Gradient Method **UPGM** [26, Eq. 2.17], the Universal Dual Gradient Method **UDGM** [26, Eq. 3.2] and the Universal Fast Gradient Method **UFGM** [26, Eq. 4.1]. The paper is relatively technical, and a substantial amount of work was necessary to translate and adapt these results for our application. The corresponding pseudo codes can be found in TOT.

One of the most attractive features of these methods is that they do not require any tuning at all except for choosing the desired (absolute) optimality gap $\epsilon > 0$. Another interesting feature is that each iteration of these methods can, and usually does, make several oracle calls. The information from these calls is used to adaptively adjust the step length, which is mainly dictated by the value of L_k (see pseudo codes).

Theorem 7. *Let $\left\{\tilde{\lambda}_k\right\}$ be the sequence generated by a universal gradient method, according to Algorithm TOT, TOT or TOT. Then,*

$$d(\lambda^\star) - d(\tilde{\lambda}_k) \leq \frac{\epsilon}{2} + \frac{4 \cdot (D(\overline{X})R_{\mathcal{D}})^2}{\epsilon \cdot (k+1)}. \tag{25}$$

Proof. According to [26, Thm. 1, Thm. 2, Thm. 3], we generally have that

$$d(\lambda^\star) - d(\tilde{\lambda}_k) \leq \frac{\epsilon}{2} + \frac{M_0^2}{\epsilon \cdot (k+1)}\|\lambda_0 - \lambda^\star\|^2, \tag{26}$$

where

$$M_0 = \sup_{\substack{\lambda_1,\lambda_2 \in \Lambda,\\ \lambda_1 \neq \lambda_2}} \|\nabla d(\lambda_1) - \nabla d(\lambda_2)\|, \tag{27}$$

see [26, Eq. 2.2]. For the specific structure of $\mathcal{D}$, we have that $M_0 \leq 2 \cdot D(\overline{X})$, and since $\Pi_\Lambda(0^{n_\lambda}) = 0^{n_\lambda}$, $\|\lambda_0 - \lambda^\star\|^2 \leq R_{\mathcal{D}}^2$. □

Universal Primal Gradient Method (UPGM), by Nesterov [26, Eq. 2.17]

Initialization.

- Set $\lambda_0 = \Pi_\Lambda(0^{n_\lambda})$.
- Choose $\epsilon > 0$ (optimality tolerance) and some initial $L_0 > 0$.

Iterations. For $k \geq 0$:

(1) Find smallest $i_k \geq 0$ such that for

$$\lambda_k^+ = \mathcal{B}_{2^{i_k} L_k}(\lambda_k) \tag{28}$$

we have

$$-d(\lambda_k^+) \leq -d(\lambda_k) + \langle -\nabla d(\lambda_k), \lambda_k^+ - \lambda_k \rangle + 2^{i_k - 1} L_k \|\lambda_k^+ - \lambda_k\|^2 + \frac{\epsilon}{2} \tag{29}$$

where, for a given $M \in R$,

$$\mathcal{B}_M(\lambda) = \Pi_\Lambda(\lambda + \frac{1}{M} \nabla d(\lambda)). \tag{30}$$

(2) Set

$$\begin{array}{l} \lambda_{k+1} = \mathcal{B}_{2^{i_k} L_k}(\lambda_k), \\ L_{k+1} = 2^{i_k - 1} L_k. \end{array} \tag{31}$$

(3) (Optionally off-line) Compute output iterates as:

$$\begin{array}{l} S_k = \sum_{i=0}^k \frac{1}{L_{i+1}} \\ \tilde{\lambda}_k = \frac{1}{S_k} \sum_{i=0}^k \frac{1}{L_{i+1}} \lambda_i \\ \tilde{d}_k = \frac{1}{S_k} \sum_{i=0}^k \frac{1}{L_{i+1}} d(\lambda_i) \end{array} \tag{32}$$

Universal Dual Gradient Method (UDGM), by Nesterov [26, Eq. 3.2]

Initialization.

- Set $\lambda_0 = \Pi_\Lambda(0^{n_\lambda})$, and let $\Phi_0 = \lambda_0$.
- Choose $\epsilon > 0$ (optimality tolerance) and some initial $L_0 > 0$.

Iterations. For $k \geq 0$:

(1) Find smallest $i_k \geq 0$ such that for point

$$\lambda_{k,i_k} = \Pi_\Lambda(\Phi_k + \frac{1}{2^{i_k} L_k} \nabla d(\lambda_k)) \tag{33}$$

we have

$$\begin{array}{rl} -d(\mathcal{B}_{2^{i_k} L_k}(\lambda_{k,i_k})) \leq & -d(\lambda_{k,i_k}) + \left\langle -\nabla d(\lambda_{k,i_k}), \mathcal{B}_{2^{i_k} L_k}(\lambda_{k,i_k}) - \lambda_{k,i_k} \right\rangle \\ & + \frac{2^{i_k} L_k}{2} \|\mathcal{B}_{2^{i_k} L_k}(\lambda_{k,i_k}) - \lambda_{k,i_k}\|^2 + \frac{\epsilon}{2}, \end{array} \tag{34}$$

where, for a given $M \in R$,

$$\mathcal{B}_M(\lambda) = \Pi_\Lambda(\lambda + \frac{1}{M} \nabla d(\lambda)). \tag{35}$$

(2) Update values:

$$\begin{array}{l} \lambda_{k+1} = \lambda_{k,i_k}, \\ L_{k+1} = 2^{i_k - 1} L_k, \\ \Phi_{k+1} = \Phi_k + \frac{1}{2L_{k+1}} \nabla d(\lambda_k). \end{array} \tag{36}$$

Optional. We can compute an averaged process of iterates that is endowed with additional theoretical properties (WHICH ONES?)

$$\begin{array}{l} S_k = \sum_{i=0}^k \frac{1}{L_{k+1}} \\ y_k = \mathcal{B}_{2^{i_k} L_k}(\lambda_{k,i_k}) \\ \tilde{\lambda}_k = \frac{1}{S_k} \sum_{i=0}^k \frac{1}{L_{i+1}} y_i \\ \tilde{d}_k = \frac{1}{S_k} \sum_{i=0}^k \frac{1}{L_{i+1}} d(y_i) \end{array} \tag{37}$$

Universal Fast Gradient Method (UFGM), by Nesterov [26, Eq. 4.1]

Initialization.

- Set $\lambda_0 = \Pi_\Lambda(0^{n_\lambda})$, and define $\phi_0(\lambda) = \frac{1}{2}\|\lambda_0 - \lambda\|^2$, $\tilde{\lambda}_0 = \lambda_0$, $A_0 = 0$, $\Phi_0 = \lambda_0$.
- Choose $\epsilon > 0$ (optimality tolerance) and some initial $L_0 > 0$.

Iterations. For $k \geq 0$:

(1) Set

$$v_k = \Pi_\Lambda(\Phi_k) \tag{38}$$

(2) Find the smallest $i_k \geq 0$ such that for

$$a_{k+1,i_k} = \frac{1 + \sqrt{1 + A_k 2^{i_k+2} L_k}}{2^{i_k+1} L_k} \tag{39}$$

$$A_{k+1,i_k} = A_k + a_{k+1,i_k} \tag{40}$$

$$\tau_{k,i_k} = \frac{a_{k+1,i_k}}{A_{k+1,i_k}} \tag{41}$$

$$\lambda_{k+1,i_k} = \tau_{k,i_k} v_k + (1 - \tau_{k,i_k})\tilde{\lambda}_k \tag{42}$$

$$\hat{\lambda}_{k+1,i_k} = \Pi_\Lambda(v_k + a_{k+1,i_k} \nabla d(\lambda_{k+1,i_k})) \tag{43}$$

$$\tilde{\lambda}_{k+1,i_k} = \tau_{k,i_k} \hat{\lambda}_{k+1,i_k} + (1 - \tau_{k,i_k})\tilde{\lambda}_k \tag{44}$$

we have

$$\begin{aligned} -d(\tilde{\lambda}_{k+1,i_k}) \leq \quad & -d(\lambda_{k+1,i_k}) + \left\langle -\nabla d(\lambda_{k+1,i_k}), \tilde{\lambda}_{k+1,i_k} - \lambda_{k+1,i_k} \right\rangle \\ & + 2^{i_k - 1} L_k \|\tilde{\lambda}_{k+1,i_k} - \lambda_{k+1,i_k}\|^2 + \tfrac{\epsilon}{2} \tau_{k,i_k}. \end{aligned} \tag{45}$$

(3) Set

$$\lambda_{k+1} = \lambda_{k+1,i_k} \tag{46}$$

$$\tilde{\lambda}_{k+1} = \tilde{\lambda}_{k+1,i_k} \tag{47}$$

$$a_{k+1} = a_{k+1,i_k} \tag{48}$$

$$\tau_k = \tau_{k,i_k} \tag{49}$$

$$A_{k+1} = A_k + a_{k+1} \tag{50}$$

$$L_{k+1} = 2^{i_k - 1} L_k \tag{51}$$

$$\Phi_{k+1} = \Phi_k + a_{k+1} \nabla d(\lambda_k). \tag{52}$$

3.5. Appendix: Simplifying the QPs in the Universal and Quasi–Monotone Gradient Methods.

In the algorithms proposed in [27, 26], several steps require the solution of auxiliary convex optimization programs; see the definition of Bregman map [26, Eq. 2.9], Step 1 of [26, Eq. 3.2], Step 1 and 2 of [26, Eq. 4.1], Step 1 of [27, Eq. 34] and Step 1 of [27, Eq. 43]. For the special case of optimizing $\mathcal{D}$, and our choice of prox-function $d(\lambda) = \frac{1}{2}\|\lambda - \hat{\lambda}\|^2$, see discussion after [27, Eq. 50], they all revolve around solving

$$\min_{\lambda \in \Lambda} f(\lambda) \tag{53}$$

with

$$f(\lambda) = a\frac{1}{2}\|\lambda - \hat{\lambda}\|^2 + \langle b, \lambda \rangle + c, \tag{54}$$

for some given $a > 0$, $b, \hat{\lambda} \in R^{n_\lambda}$, $c \in R$ and where Λ is as in (5). Next we show that (53) can be solved by projecting the unconstrained minimizer of (54) on Λ. Given the Lagrangian function associated to (53)

$$\mathcal{L}(\lambda, w) = f(\lambda) + w' \sum_{s \in S} \lambda_s, \quad w \in R^{n_x}, \tag{55}$$

Class	Instances	Scenarios	Application	Structure	Source
SSLP	12	3–2000	Stochastic server location problem.	Pure binary first stage, mixed integer second stage.	[28]
DCAP	6	200–500	Dynamic capacity acquisition and allocation.	Mixed–integer first stage, pure binary second stage.	[2]
SEMI	3	2–4	Planning of semiconductor tool purchases.	Mixed–integer first stage, continuous second stage.	[5]
SMKP	30	20	Stochastic multiple knapsack problems.	Pure integer first and second stage.	[3]
SIZES	1	10	Product substitution applications.	Mixed–integer first and second stage.	[21]

TABLE 1. classes of instances studied; corresponding datasets are publicly available at [1].

the KKT conditions read

$$\begin{aligned} \textbf{stationarity:} \quad & \nabla_\lambda \mathcal{L}(\lambda, w)|_{\lambda^\star, w^\star} = a(\lambda^\star - \hat{\lambda}) + b + [w^\star, \dots, w^\star] = 0 \\ \textbf{primal feasibility:} \quad & \textstyle\sum_{s\in S} \lambda_s^\star = 0 \\ \textbf{dual feasibility:} \quad & w^\star \in R^{n_x} \\ \textbf{complementarity:} \quad & \emptyset. \end{aligned}$$

It is straightforward to verify that

$$\lambda_s^\star = \hat{\lambda}_s - \frac{1}{a}(b_s + w^\star), \quad s \in S$$
$$w^\star = \frac{a}{|S|}\left(\sum_{\tilde{s}\in S} \lambda_{\tilde{s}}^\star - b_{\tilde{s}}\right)$$

satisfies all the KKTs and, since (53) is a convex program satisfying Slater conditions, it is a valid optimizer. It also corresponds to the point obtained by taking the unconstrained optimizer $\tilde{\lambda}^\star = \hat{\lambda} - \frac{1}{a}b$ and projecting it on Λ.

Remark 8. *It is easy to verify that the same principle applies to other structures of Λ, for example when $\Lambda = R_+^{n_\lambda}$, which arises in applications of duality in which the relaxed constraints are inequalities.*

4. EXPERIMENTS

4.1. **Setup.** The datasets used in our experiments are instances of two–stage stochastic mixed–integer programs that are publicly available at [1]. We have specifically used instances from the datasets SSLP, SEMI, SIZES, DCAP and SMKP, which entail optimization programs of varying sizes and structures, and containing both synthetic as well as practical datasets, see Table 1. They amount to 52 instances in total, and are distributed using the standard SMPS file format [16], for which we have developed a parsing utility in Python available at [REF].

The performance of the methods is measured exclusively based on the number of oracle calls n_{oracle} they require. This is important because other measures of performance (e.g., computation time) are heavily dependent on implementation details (language and data structures used, hardware available, etc.). Also, often in practice the most expensive operation is computing the output of the oracle.

Class	SG 1/k	SG const	UPGM	UDGM	UFGM	DSA	TA 1	TA 2	CP	bundle
SSLP	10	0.1	1	1	1	0.1	0.01	0.01	0.1	0.01
DCAP	1	0.01	10	10	0.01	1	1	1	100	1
SEMI	1	1	10	100	100	1	1	100	100	1
SMKP	1	0.1	0.01	100	100	10	1	1	100	10
SIZES	0.1	0.01	100	100	100	10	100	10	100	100

TABLE 2. parameters settings for the various methods investigated.

All methods have been implemented in Python 2.7 and are available at [REF], allowing for replication of the results. Evaluation of the first–order oracle requires a solution to the inner problem in (3), which we obtain using Gurobi [18].

4.2. **Procedure.** The overall experiment procedure entails two stages: parameter tuning and performance measurement.

Parameter Tuning. The first stage involves finding a reasonable setting for the tuning parameters of the methods. The parameters p considered for each method are,

- the initial stepsize $p = s_0$ for subgradient methods,
- the optimality tolerance $p = \epsilon$ for cutting planes, bundle and universal gradient methods,
- and $p = \gamma$ for quasi monotone methods.

To do find a reasonable parameter setting, we select one representative instance for each model class (SSLP, SEMI, SIZES, DCAP and SMKP). For each one of these instances, we run each algorithm six times with a different parameter setting each time, taken from $p \in \{0.01, 0.1, 1.0, 10, 100\}$. We then record, for each method, the best $d(\lambda_k)$ found across all parameters, and determine which parameter setting was earliest, in terms of oracle calls, within 0.1% of that value. This is then the parameter setting used for all other instances within that class in the second stage of our experiment. The resulting parameters choices are in Table 2.

Note that we specifically do not want to find the best parameter tuning for each method and instance. Rather, we find a good parameter setting for one representative instance of the entire class, and use that setting for all other instances within that class. This better reflects the practical circumstance in which once a set of parameters is chosen from initial experimentations, it is kept constant with new instances.

Performance Measurement. Combined, there are 52 optimization program instances and 10 methods, leading to a total of 520 experiments. The number of dual iterations for each instance is adjusted such that each experiment lasts up to 1 hour; however, to ensure that we have a sufficient number of points to make evaluations, we set the minimum number of oracle calls to 100. This lower cap affects some harder instances from SSLP, SMKP and SEMI. We also set a maximum of 1000 oracle calls.

Methods' progress on each instance, as measured by the values of $d(\lambda_k)$, is then stored and post processed to determine a ranking. For each instance, we identify the best (highest) $d(\lambda_k)$ value achieved across all methods and call it $\bar{d}^\star$, the best proxy for the actual $d^\star$ that is available to us. We then assign a rank based on the number of oracle calls required to be within a pre-determined relative gap from $\bar{d}^\star$. For example, we set the gap to be 10% and the first method achieving it (in terms of oracle calls) is given the rank 1. Methods that achieve a gap with the same number of calls are given the same rank; this is frequently the case for the 10% threshold, since many instances satisfy $\frac{d(\lambda_0)-\bar{d}^\star}{\bar{d}^\star} \leq 0.1$ at $\lambda_0 = \Pi_\Lambda(0)$. As a penalty, methods that are unable to achieve a given gap are assigned a score of 20 for that particular experiment. The relative gaps measured are at 10%, 1%, 0.1% and 0.01%.

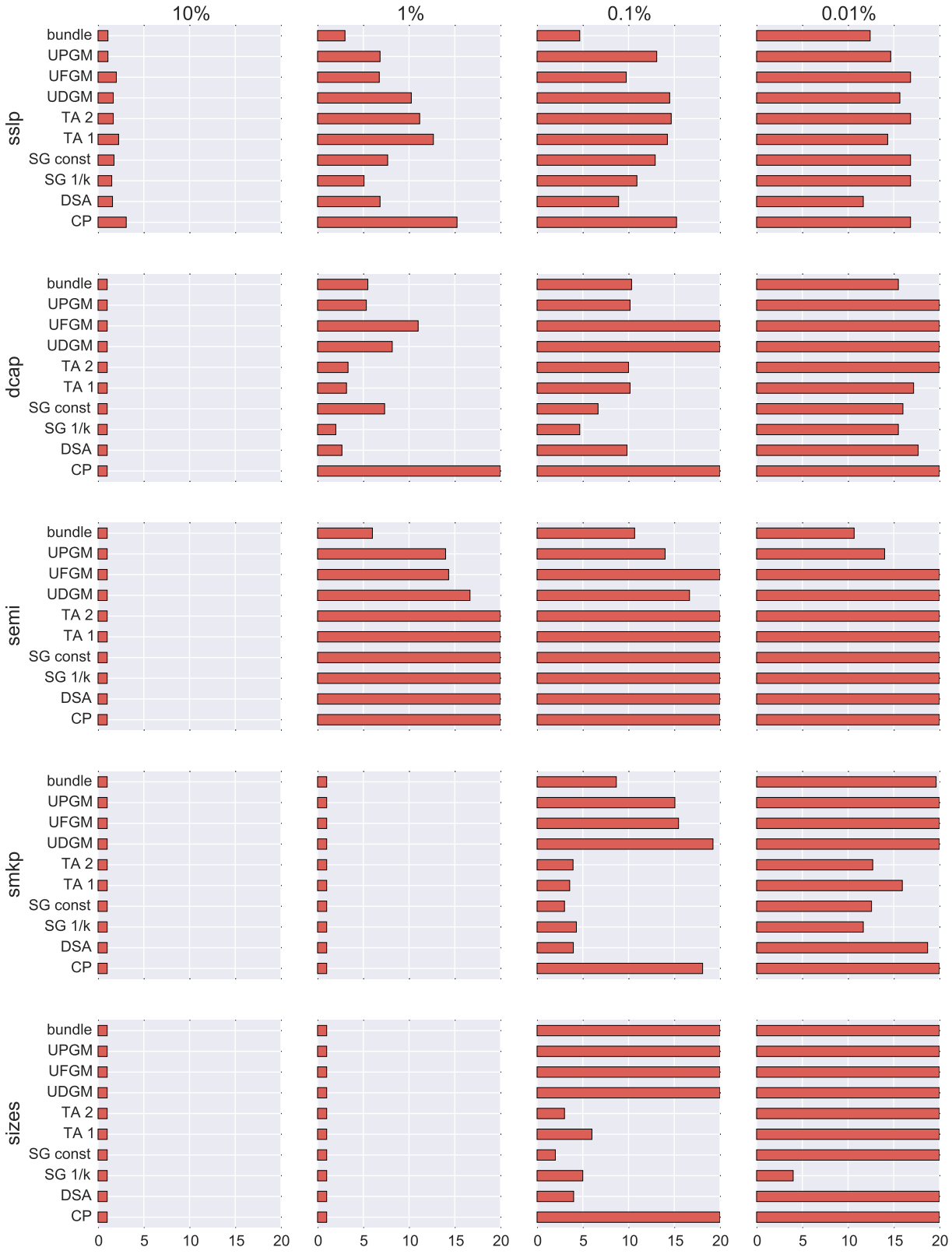

FIGURE 1. methods average rank per problem class; the lower the score the better.

4.3. **Results.** Figures 1 and 2 summarize the results of our experiments. Figure 1 displays the rank of each method averaged over all instances of a particular class, while Figure 2 is the global average over all instances. The lower the score the better.

FIGURE 2. methods average rank over all instances in the study; the lower the score the better.

The most striking result is that all methods score well on the 10% threshhold. This is because in almost all instances, except for some instances in SSLP, the initial point $\lambda_0 = \Pi_\Lambda(0)$ already satisfies this threshold. This may result from the peculiar structure of $\mathcal{D}$ when duality is applied to two stage stochastic instances. Dual variables are used to penalize mismatches between copies of first stage decisions, see Eq. (2), and our experiments indicate that, even with zero penalty, copies already match to a good degree. Thus, the performance of the methods investigated in this paper could be very different for other structures of $\mathcal{D}$.

We also found that in general, the traditional algorithms SG const and SG $1/k$, and the quasi monotone subgradient methods are the most competitive on our scoring system. They are also the simplest algorithms to implement. The only class in which they fail to find solutions at 1% or better is SEMI – one of the hardest classes in our datasets. For this class, universal and bundle methods perform best. Generally, universal methods perform reasonably well for SSLP, DCAP and SEMI, but are not the methods of choice for SMKP and SIZES.

The only algorithm that is consistently found to be the inferior choice across almost all instances is cutting planes. Our experiments revealed further problems with this method that are not included in the results of this paper. At initialization, a bounded box has to be decided upon, within which to search for an optimizer. The method then typically visits several vertices of this box in its search. The problem is that, at least in the datasets we studied, these vertices result in much harder inner problems and the corresponding values of $d(\lambda_k)$ are far from the optimum. These problems are exacerbated by a high dimensionality of the dual space (large n_λ).

5. Conclusions

References

1. S. Ahmed, R. Garcia, N. Kong, L. Ntaimo, G. Parija, and S. Sen F. Qiu, *SIPLIB: A stochastic integer programming test problem library*, http://www2.isye.gatech.edu/~sahmed/siplib/, 2015.
2. Shabbir Ahmed and Renan Garcia, *Dynamic capacity acquisition and assignment under uncertainty*, Annals of Operations Research **124** (2003), no. 1-4, 267–283.
3. Gustavo Angulo, Shabbir Ahmed, and Santanu S Dey, *Improving the integer l-shaped method*, INFORMS Journal on Computing **28** (2016), no. 3, 483–499.
4. Kurt M. Anstreicher and Laurence A. Wolsey, *Two "well-known" properties of subgradient optimization*, Math. Program. **120** (2009), no. 1, 213–220.
5. Francisco Barahona, Stuart Bermon, Oktay Gunluk, and Sarah Hood, *Robust capacity planning in semiconductor manufacturing*, Naval Research Logistics **52** (2005), no. 5, 459–468.
6. Alexandre Belloni, *Introduction to bundle methods, lecture notes*, February 2005.
7. Dimitri P. Bertsekas, *Nonlinear programming*, 2nd ed., Athena Scientific, September 1999.
8. Dimitri P. Bertsekas, G. Lauer, N. Sandell, and T. Posbergh, *Optimal short-term scheduling of large-scale power systems*, IEEE Transactions on Automatic Control **28** (1983), no. 1, 1–11.
9. Dimitri P Bertsekas and Athena Scientific, *Convex optimization algorithms*, Athena Scientific Belmont, 2015.

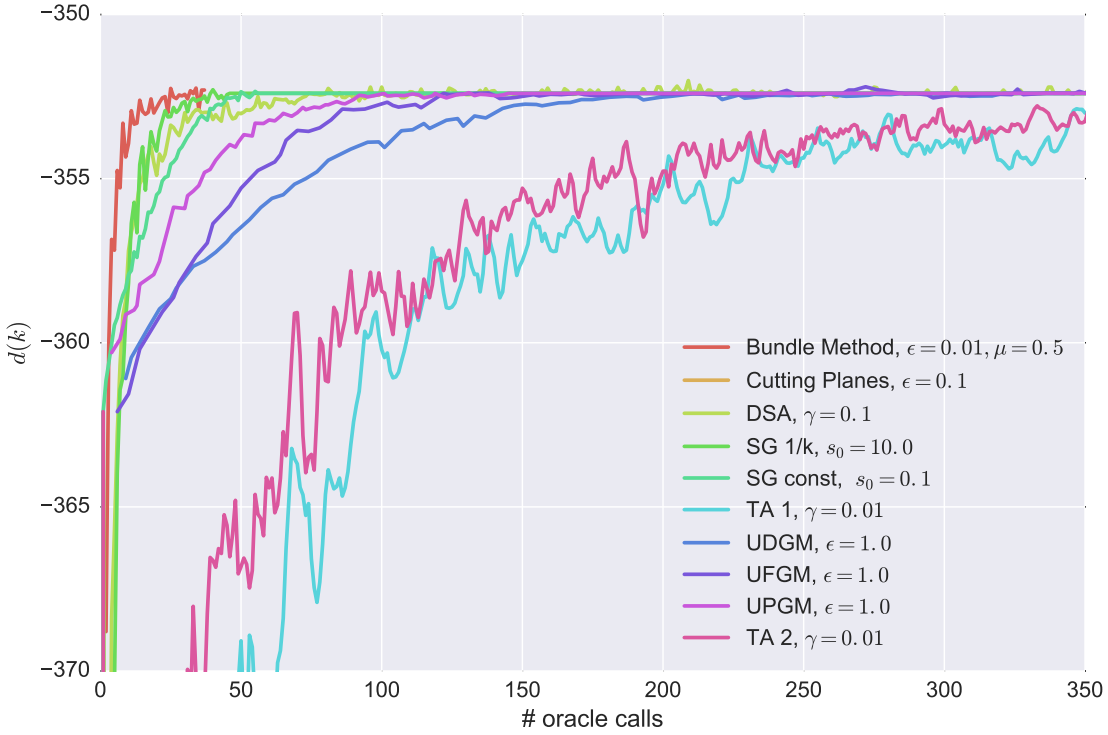

FIGURE 3.

10. John R Birge and MAH Dempster, *Stochastic programming approaches to stochastic scheduling*, Journal of Global Optimization **9** (1996), no. 3-4, 417–451.
11. John R Birge and Francois Louveaux, *Introduction to stochastic programming*, Springer Science & Business Media, 2011.
12. Stephen Boyd and Almir Mutapcic, *Subgradient methods*, Lecture notes of EE364b, Stanford University, Winter Quarter **2007** (2006).
13. Claus C Carøe, Andrzej Ruszczynski, and Rüdiger Schultz, *Unit commitment under uncertainty via two-stage stochastic programming*, (1997).
14. Claus C. Caroe and Ruediger Schultz, *Dual decomposition in stochastic integer programming*, Operations Research Letters **24** (1999), no. 1, 37–45.
15. Darinka Dentcheva and Werner Römisch, *Duality gaps in nonconvex stochastic optimization*, Mathematical Programming **101** (2004), no. 3, 515–535.
16. H.I. Gassmann, *The SMPS format for stochastic linear programs*, http://myweb.dal.ca/gassmann/smps2.htm, 2005.
17. A. M. Geoffrion, *Lagrangean relaxation for integer programming*, Mathematical Programming Studies, vol. 2, Springer Berlin Heidelberg, 1974.
18. Inc. Gurobi Optimization, *Gurobi optimizer reference manual*, http://www.gurobi.com, 2015.
19. Michael Held and Richard M Karp, *The Traveling-Salesman problem and minimum spanning trees*, Operations Research **18** (1970), no. 6, 1138–1162.
20. Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal, *Convex analysis and minimization algorithms i: fundamentals*, vol. 305, Springer science & business media, 2013.
21. Soheila Jorjani, Carlton H Scott, and David L Woodruff, *Selection of an optimal subset of sizes*, International journal of production research **37** (1999), no. 16, 3697–3710.
22. Claude Lemarechal, *Lagrangian relaxation*, Computational Combinatorial Optimization (Michael Juenger and Denis Naddef, eds.), vol. 2241, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 112–156.
23. Claude Lemaréchal, Arkadii Nemirovskii, and Yurii Nesterov, *New variants of bundle methods*, Mathematical programming **69** (1995), no. 1-3, 111–147.
24. John M Mulvey and Andrzej Ruszczyński, *A new scenario decomposition method for large-scale stochastic optimization*, Operations research **43** (1995), no. 3, 477–490.
25. Angelia Nedic and Asuman Ozdaglar, *Approximate primal solutions and rate analysis for dual subgradient methods*, SIAM J. on Optimization **19** (2009), no. 4, 1757–1780.
26. Yu Nesterov, *Universal gradient methods for convex optimization problems*, Mathematical Programming **152** (2015), no. 1-2, 381–404.

27. Yu Nesterov and Vladimir Shikhman, *Quasi-monotone subgradient methods for nonsmooth convex minimization*, Journal of Optimization Theory and Applications **165** (2015), no. 3, 917–940.
28. Lewis Ntaimo and Suvrajeet Sen, *The million-variable march for stochastic combinatorial optimization*, Journal of Global Optimization **32** (2005), no. 3, 385–400.
29. Stefan Richter, *A computational complexity certification of gradient methods for real-time model predictive control*, 2012.
30. R Tyrrell Rockafellar and Roger J-B Wets, *Scenarios and policy aggregation in optimization under uncertainty*, Mathematics of operations research **16** (1991), no. 1, 119–147.
31. Charles H Rosa and Andrzej Ruszczyński, *On augmented lagrangian decomposition methods for multistage stochastic programs*, Annals of Operations Research **64** (1996), no. 1, 289–309.
32. Andrzej P Ruszczyński, *Nonlinear optimization*, vol. 13, Princeton university press, 2006.
33. G Sand, S Engell, A Märkert, R Schultz, and C Schulz, *Approximation of an ideal online scheduler for a multiproduct batch plant*, Computers & chemical engineering **24** (2000), no. 2, 361–367.
34. N.Z. Shor, K.C. Kiwiel, and A. Ruszczynski, *Minimization methods for non-differentiable functions*, vol. 121, Springer Verlag, 1985.