Clustering-Based Preconditioning for Stochastic Programs

Yankai Cao · Carl D. Laird · Victor M. Zavala

Abstract We present a clustering-based preconditioning strategy for KKT systems arising in stochastic programming within an interior-point framework. The key idea is to perform adaptive clustering of scenarios (inside-the-solver) based on their influence on the problem as opposed to cluster scenarios based on problem data alone, as is done in existing (outside-the-solver) approaches. We derive spectral and error properties for the preconditioner and demonstrate that scenario compression rates of up to 87% can be obtained, leading to dramatic computational savings. In addition, we demonstrate that the proposed preconditioner can avoid scalability issues of Schur decomposition in problems with large first-stage dimensionality.

Keywords preconditioning · interior-point · stochastic · large-scale · clustering.

1 Preliminaries

We consider two-stage stochastic programs of the form

$$\min \left(\frac{1}{2}x_0^T Q_0 x_0 + d_0^T x_0\right) + \sum_{s\in\mathcal{S}} \left(\frac{1}{2}x_s^T Q_s x_s + d_s^T x_s\right) \tag{1a}$$

$$\text{s.t.} \quad W_0 x_0 = b_0, \quad (y_0) \tag{1b}$$

$$T_s x_0 + W_s x_s = b_s, \quad (y_s),\ s\in\mathcal{S} \tag{1c}$$

$$x_0 \geq 0, \quad (\nu_0) \tag{1d}$$

$$x_s \geq 0, \quad (\nu_s),\ s\in\mathcal{S}. \tag{1e}$$

Preprint Number ANL/MCS-P3050-1112

Yankai Cao · Carl D. Laird
School of Chemical Engineering, Purdue University
480 Stadium Mall Drive, West Lafayette, IN 47907
Tel.: +1-765-494-0085
Fax: +1-765-494-0805
E-mail: {cao142,lairdc}@purdue.edu

Victor M. Zavala
Mathematics and Computer Science Division, Argonne National Laboratory
9700 South Cass Avenue, Argonne, IL 60439
Tel.: +1-630-252-3343
Fax: +1-630-252-5986
E-mail: vzavala@mcs.anl.gov

Here, $\mathcal{S} := \{1..n_S\}$, where n_S is the number of scenarios, $x_0, \nu_0 \in \Re^{n_0}$, $x_s, \nu_s \in \Re^{n_s}$, $y_0 \in \Re^{m_0}$, and $y_s \in \Re^{m_s}$. The total number of variables is $n := n_0 + \sum_{s\in\mathcal{S}} n_s$, of equality constraints is $m := m_0 + \sum_{s\in\mathcal{S}} m_s$, and of inequalities is n. We refer to (x_0, y_0, ν_0) as the first-stage variables and to (x_s, y_s, ν_s), $s \in \mathcal{S}$, as the second-stage variables. We refer to equation (1a) as the cost function. The *data* defining problem (1) is given by the cost coefficients d_0, Q_0, Q_s, d_s, the right-hand side coefficients b_0, b_s, and the matrix coefficients T_s, W_s. We refer to Q_s, d_s, b_s, T_s, W_s as the *scenario data*.

As is typical in stochastic programming, the number of scenarios can be large and limits the scope of existing off-the-shelf solvers. In this work, we present strategies that cluster scenarios at the linear algebra level to mitigate complexity. We start by presenting some basic notation.

The Lagrange function of (1) is given by

$$\mathcal{L}(x,y,\nu) = \frac{1}{2}x_0^T Q_0 x_0 + d_0^T x_0 + y_0^T(W_0 x_0 - b_0) - \nu_0^T x_0 + \sum_{s\in\mathcal{S}} \left(\frac{1}{2}x_s^T Q_s x_s + d_s^T x_s + y_s^T(T_s x_0 + W_s x_s - b_s) - \nu_s^T x_s\right). \tag{2}$$

Here, $x := [x_0^T, x_1^T, ..., x_S^T]$, $y^T := [y_0^T, y_1^T, ..., y_S^T]$, and $\nu^T := [\nu_0^T, \nu_1^T, ..., \nu_S^T]$. In a primal-dual interior-point (IP) setting we seek to solve nonlinear systems of the form

$$\nabla_{x_0}\mathcal{L} = 0 = Q_0 x_0 + d_0 + W_0^T y_0 - \nu_0 + \sum_{s\in\mathcal{S}} T_s^T y_s \tag{3a}$$
$$\nabla_{x_s}\mathcal{L} = 0 = Q_s x_s + d_s + W_s^T y_s - \nu_s,\ s \in \mathcal{S} \tag{3b}$$
$$\nabla_{y_0}\mathcal{L} = 0 = W_0 x_0 - b_0 \tag{3c}$$
$$\nabla_{y_s}\mathcal{L} = 0 = T_s x_0 + W_s x_s - b_s,\ s \in \mathcal{S} \tag{3d}$$
$$0 = X_0 V_0 e - \mu e \tag{3e}$$
$$0 = X_s V_s e - \mu e,\ s \in \mathcal{S}, \tag{3f}$$

with the implicit condition $x_0, \nu_0, x_s, \nu_s \geq 0$. Here, $\mu \geq 0$, $e \in \Re^n$ is a vector of ones, $X_0 := \text{diag}(x_0)$, $X_s := \text{diag}(x_s)$, $V_0 := \text{diag}(\nu_0)$, and $V_s := \text{diag}(\nu_s)$. We define $\alpha_0 := X_0 V_0 e - \mu e$ and $\alpha_s := X_s V_s e - \mu e,\ s \in \mathcal{S}$. The search step is obtained by solving the linear system

$$Q_0\Delta x_0 + W_0^T\Delta y_0 + \sum_{s\in\mathcal{S}} T_s^T\Delta y_s - \Delta\nu_0 = -\nabla_{x_0}\mathcal{L} \tag{4a}$$
$$Q_s\Delta x_s + W_s^T\Delta y_s - \Delta\nu_s = -\nabla_{x_s}\mathcal{L},\ s \in \mathcal{S} \tag{4b}$$
$$W_0\Delta x_0 = -\nabla_{y_0}\mathcal{L} \tag{4c}$$
$$T_s\Delta x_0 + W_s\Delta x_s = -\nabla_{y_s}\mathcal{L},\ s \in \mathcal{S} \tag{4d}$$
$$X_0\Delta\nu_0 + V_0\Delta x_0 = -\alpha_0 \tag{4e}$$
$$X_s\Delta\nu_s + V_s\Delta x_s = -\alpha_s,\ s \in \mathcal{S}. \tag{4f}$$

After eliminating the bound multipliers from the inear system we obtain

$$CPH_0\Delta x_0 + W_0^T\Delta y_0 + \sum_{s\in\mathcal{S}} T_s^T\Delta y_s = -r_{x_0} \tag{5a}$$
$$H_s\Delta x_s + W_s^T\Delta y_s = -r_{x_s},\ s \in \mathcal{S} \tag{5b}$$
$$W_0\Delta x_0 = -r_{y_0} \tag{5c}$$
$$T_s\Delta x_0 + W_s\Delta x_s = -r_{y_s},\ s \in \mathcal{S}, \tag{5d}$$

where

$$\Delta\nu_0 = -X_0^{-1}V_0\Delta x_0 - X_0^{-1}\alpha_0 \tag{6a}$$
$$\Delta\nu_s = -X_s^{-1}V_s\Delta x_s - X_s^{-1}\alpha_s,\ s \in \mathcal{S}, \tag{6b}$$

and

$$H_0 := Q_0 + X_0^{-1}V_0 \tag{7a}$$
$$H_s := Q_s + X_s^{-1}V_s,\ s \in \mathcal{S}. \tag{7b}$$

We also have that $r_{x_0} := \nabla_{x_0}\mathcal{L} - X_0^{-1}\alpha_0$, $r_{x_s} := \nabla_{x_s}\mathcal{L}_s - X_s^{-1}\alpha_s$, $r_{y_0} := \nabla_{y_0}\mathcal{L}$, and $r_{y_s} := \nabla_{y_s}\mathcal{L}$. System (5) has the arrowhead form

$$\begin{bmatrix} K_1 & & & B_1 \\ & K_2 & & B_2 \\ & & \ddots & \vdots \\ & & K_S & B_S \\ B_1^T & B_2^T & \dots & B_S^T \ K_0 \end{bmatrix} \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \vdots \\ \Delta w_S \\ \Delta w_0 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_S \\ r_0 \end{bmatrix}, \tag{8}$$

where $\Delta w_0^T := [\Delta x_0^T, \Delta y_0^T]$, $\Delta w_s^T := [\Delta x_s^T, \Delta y_s^T]$, $r_0^T := [r_{x_0}^T, r_{y_0}^T]$, $r_s^T := [r_{x_s}^T, r_{y_s}^T]$, and

$$K_0 := \begin{bmatrix} H_0 & W_0^T \\ W_0 & 0 \end{bmatrix},\ K_s := \begin{bmatrix} H_s & W_s^T \\ W_s & 0 \end{bmatrix},\ B_s := \begin{bmatrix} 0 & 0 \\ T_s & 0 \end{bmatrix}. \tag{9}$$

We refer to the linear system (8) as the *KKT system*. We assume that each scenario block matrix $K_s,\ s \in \mathcal{S}$ is nonsingular.

We use the following notation to define a block-diagonal matrix M composed of blocks $M_1, M_2, M_3, ...$:

$$M = \text{blkdiag}\{M_1, M_2, M_3, ...\}. \tag{10}$$

In addition, we use the following notation to define a matrix B that stacks (row-wise) the blocks $B_1, B_2, B_3...$:

$$B = \text{rowstack}\{B_1, B_2, B_3, ...\}. \tag{11}$$

We apply the same rowstack notation for vectors. We use the notation $v^{(k)}$ to indicate the $k-th$ entry of vector v. We use vec(M) to denote the row-column vectorization of matrix M and we define $\sigma_{min}(M)$ as the smallest singular value of matrix M. We use $\|\cdot\|$ to denote the Euclidean norm for vectors and the Frobenius norm for matrices, and we recall that $\|M\| = \|\text{vec}(M)\|$ for matrix M.

2 Clustering Setting

In this section, we review work on scenario reduction and highlight the differences and contributions of our work. We then present our clustering-based preconditioner for the KKT system (8).

2.1 Related Work and Contributions

Scenario clustering or aggregation is a strategy commonly used in stochastic programming to reduce computational complexity. We can classify these strategies as outside-the-solver and inside-the-solver strategies. Outside-the-solver strategies perform clustering on the scenario data (right-hand sides, matrices, and gradients) prior to the solution of the problem [8,14,12, 5]. This approach can provide lower bounds and error bounds for linear programs (LPs) and this feature can be exploited in branch-and-bound procedures [5,1,20,25].

Outside-the-solver clustering approaches give rise to several inefficiencies, however. First, several optimization problems might need to be solved in order to refine the solution. Second, these approaches focus on the problem data and thus *do not capture the effect of the data on the particular problem at hand*. This is an important inefficiency because scenarios that are close to each other (in terms of data) might have very different impact on the cost function if they are close to the constraint boundary. Conversely, two scenarios that are far apart (in terms of data) might have similar contributions to the cost function. We also highlight that many scenario generation procedures require knowledge of the underlying probability distributions [9,12] which are often not available in closed form (e.g., weather forecasting) [24,16].

In this work, we seek to overcome these inefficiencies by performing clustering adaptively inside-the-solver. In an interior-point setting this can be done by creating a preconditioner for the KKT system (8) by *clustering* the scenario blocks. A key advantage of this approach is that a single optimization problem is solved and the clusters are refined only if the preconditioner is not sufficiently accurate. In addition, this approach provides a mechanism to capture the influence of the data on the particular problem at hand. Another advantage is that it can enable sparse preconditioning of Schur complement systems. This is beneficial in situations where the number of first-stage variables is large and thus direct Schur complement decomposition is expensive. Moreover, our approach does not require any knowledge of the underlying probability distributions generating the scenario data. Thus, it can be applied to problems in which simulators are used to generate scenarios (e.g., weather forecasting), and it can be applied to problem classes that exhibit similar structures such as support vector machines [10,13] and scenario-based robust optimization [4]. Our proposed clustering approach can also be used in combination with outside-the-solver scenario aggregation procedures.

Related work on inside-the-solver scenario reduction strategies includes stochastic Newton methods [3]. These approaches sample scenarios to create a smaller representation of the KKT system. Existing approaches, however, cannot handle constraints. Scenario and constraint reduction approaches for IP solvers have been presented in [6,22,18,7]. In [6,22], scenarios that have little influence on the step computation are eliminated from the KKT system. This influence is measured in terms of the magnitude of the constraint multipliers or in terms of the products $X_s^{-1}V_s$. In those works, it was found that a large proportion of scenarios or constraints can be eliminated without compromising convergence. The elimination potential can be limited in early iterations, however, because it is not clear which scenarios have strong or weak influence on the solution. In addition, these approaches eliminate the scenarios from the problem formulation, and thus special safeguards are needed to guarantee convergence. Our proposed clustering approach does not eliminate the scenarios from the problem formulation; instead, the scenario space is compressed to construct preconditioners.

In [18] preconditioners for Schur systems are constructed by sampling the full scenario set. A shortcoming of this sampling approach is that scenario outliers with strong influence might not be captured in the preconditioner. In addition, this approach still requires a dense preconditioner for the Schur complement, which hinders scalability in problems with many first-stage variables. Our preconditioning approach enables sparse preconditioning and thus avoids form-

ing and factorizing dense Schur complements. In addition, compared with approaches in [6,22, 18], our approach clusters scenarios instead of eliminating them (either by sampling or by measuring strong/weak influence). This enables us to capture scenario redundancies and outliers. In [7], scenarios are clustered to solve a reduced problem and the solution of this problem is used to warm-start the problem defined for the full scenario set. The approach can reduce the number of iterations of the full scenario problem; but the work per iteration is not reduced, as in our approach.

2.2 Clustering-Based Preconditioner

To derive our clustering-based preconditioner, we partition the full scenario set $\mathcal{S}$ into C clusters, where $C \leq S$. For each cluster $i \in \mathcal{C} := \{1..C\}$ we define a subset $\mathcal{S}_i \subseteq \mathcal{S}$ with $\omega_i := |\mathcal{S}_i|$ scenarios satisfying

$$\bigcup_{i \in \mathcal{S}} \mathcal{S}_i = \mathcal{S} \tag{12a}$$

$$\mathcal{S}_i \bigcap \mathcal{S}_j = \emptyset,\ i, j \in \mathcal{C}, j \neq i. \tag{12b}$$

For each scenario $s \in \mathcal{S}$, we define an index pair $(i, j),\ i \in \mathcal{C}, j \in \mathcal{C}_i := \{1..\omega_i\}$, to indicate that scenario s belongs to cluster i and is located at the local position j. We use this to define the ordered scenario set

$$\Omega := \{(1,1), (1,\omega_1), (2,1), ..., (2,\omega_2), ..., (C,1), (C,\omega_C)\}. \tag{13}$$

For each cluster $i \in \mathcal{C}$ we pick an index $j_i \in \mathcal{S}_i$ to represent the cluster and we use these indexes to define the compressed set $\mathcal{R} := \{j_1, j_2, .., j_C\}$. Note that $|\mathcal{R}| = C$.

We define the binary indicator $\kappa_{s,i},\ s \in \mathcal{S}, i \in \mathcal{C}$, satisfying

$$\kappa_{s,i} = \begin{cases} 1 & \text{if } s \in \mathcal{S}_i \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Using this notation, we have that for arbitrary vectors $v_{j_i}, v_{(i,j)}, i \in \mathcal{C}, j \in \mathcal{C}_i$, the following identities hold:

$$\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} \|v_{j_i} - v_{(i,j)}\| = \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{C}} \kappa_{s,i} \|v_{j_i} - v_s\| \tag{15a}$$

$$\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} v_{(i,j)} = \sum_{s \in \mathcal{S}} v_s \tag{15b}$$

$$\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} v_{j_i} = \sum_{i \in \mathcal{C}} \omega_i v_{j_i}. \tag{15c}$$

At this point, we have yet to define appropriate procedures for obtaining the cluster information $\mathcal{S}, \mathcal{R}, \mathcal{S}_i, \mathcal{C}_i, \omega_i$ and $\kappa_{s,i}$. These are discussed in Section 3.

Consider now the ordered representation of the KKT system (8),

$$\underbrace{\begin{bmatrix} K_\Omega & B_\Omega \\ B_\Omega^T & K_0 \end{bmatrix}}_{:=K} \underbrace{\begin{bmatrix} q_\Omega \\ q_0 \end{bmatrix}}_{:=q} = \underbrace{\begin{bmatrix} t_\Omega \\ t_0 \end{bmatrix}}_{:=t}, \tag{16}$$

where

$$K_\Omega := \text{blkdiag}\left\{K_{(1,1)}, K_{(1,2)}, ..., K_{(1,\omega_1)}, ..., K_{(C,\omega_C)}\right\} \tag{17a}$$
$$B_\Omega := \text{rowstack}\left\{B_{(1,1)}, B_{(1,2)}, ..., B_{(1,\omega_1)}, ..., B_{(C,\omega_C)}\right\} \tag{17b}$$
$$q_\Omega := \text{rowstack}\left\{q_{(1,1)}, q_{(1,2)}, ..., q_{(1,\omega_1)}, ..., q_{(C,\omega_C)}\right\} \tag{17c}$$
$$t_\Omega := \text{rowstack}\left\{t_{(1,1)}, t_{(1,2)}, ..., t_{(1,\omega_1)}, ..., t_{(C,\omega_C)}\right\}. \tag{17d}$$

Here, (t_0, t_Ω) are arbitrary right-hand side vectors and (q_0, q_Ω) are solution vectors. If the solution vector (q_0, q_Ω) does not exactly solve (16), it will induce a residual vector that we define as $\epsilon_r^T := [\epsilon_{r_0}^T, \epsilon_{r_\Omega}^T]$ with

$$\epsilon_{r_0} := K_0 q_0 + B_\Omega^T q_\Omega - t_0 \tag{18a}$$
$$\epsilon_{r_\Omega} := K_\Omega q_\Omega + B_\Omega q_0 - t_\Omega. \tag{18b}$$

The Schur system of (16) is given by

$$\underbrace{(K_0 - B_\Omega^T K_\Omega^{-1} B_\Omega)}_{:=Z} q_0 = \underbrace{t_0 - B_\Omega^T K_\Omega^{-1} t_\Omega}_{:=t_Z}. \tag{19}$$

Because K_Ω is block-diagonal, we have that

$$Z = K_0 - \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}_i} B_{(i,j)}^T K_{(i,j)}^{-1} B_{(i,j)} \tag{20a}$$
$$t_Z = t_0 - \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}_i} B_{(i,j)}^T K_{(i,j)}^{-1} t_{(i,j)}. \tag{20b}$$

We now define the following:

$$K_\mathcal{R}^\omega := \text{blkdiag}\left\{\omega_1 K_{j_1}, \omega_2 K_{j_2}, ..., \omega_C K_{j_C}\right\} \tag{21a}$$
$$K_\mathcal{R}^{1/\omega} := \text{blkdiag}\left\{1/\omega_1 K_{j_1}, 1/\omega_2 K_{j_2}, ..., 1/\omega_C K_{j_C}\right\} \tag{21b}$$
$$B_\mathcal{R} := \text{rowstack}\left\{B_{j_1}, B_{j_2}, ..., B_{j_C}\right\} \tag{21c}$$
$$q_\mathcal{R} := \text{rowstack}\left\{q_{j_1}, q_{j_2}, ..., q_{j_C}\right\} \tag{21d}$$
$$t_\mathcal{R} := \text{rowstack}\left\{t_{j_1}, t_{j_2}, ..., t_{j_C}\right\}. \tag{21e}$$

In other words, $K_\mathcal{R}^\omega$ is a block-diagonal matrix in which each block entry K_{j_i} is weighted by the scalar weight ω_i and $K_\mathcal{R}^{1/\omega}$ is a block-diagonal matrix in which each block entry K_{j_i} is weighted by $1/\omega_i$. Note that

$$(K_\mathcal{R}^{1/\omega})^{-1} = (K_\mathcal{R}^{-1})^\omega. \tag{22}$$

We now present the *clustering-based preconditioner* (CP),

$$\begin{bmatrix} K_\mathcal{R}^{1/\omega} & B_\mathcal{R} \\ B_\mathcal{R}^T & K_0 \end{bmatrix} \begin{bmatrix} q_\mathcal{R} \\ q_0 \end{bmatrix} = \begin{bmatrix} t_\mathcal{R} \\ t_0 + t_{CP} \end{bmatrix} \tag{23a}$$
$$K_{(i,j)} q_{(i,j)} = t_{(i,j)} - B_{(i,j)} q_0, \quad i \in \mathcal{C},\ j \in \mathcal{C}_i, \tag{23b}$$

where

$$t_{CP} := \sum_{i\in\mathcal{C}} \omega_i B_{j_i}^T K_{j_i}^{-1} t_{j_i} - \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}_i} B_{(i,j)}^T K_{(i,j)}^{-1} t_{(i,j)} \tag{24}$$

is a correction term that is used to establish consistency between CP and the KKT system. In particular, the Schur system of (23a) is

$$\begin{aligned}\bar{Z}q_0 &= t_0 + t_{CP} - B_{\mathcal{R}}^T (K_{\mathcal{R}}^{1/\omega})^{-1} t_{\mathcal{R}} \\ &= t_0 + t_{CP} - \sum_{i\in\mathcal{C}} \omega_i B_{j_i}^T K_{j_i}^{-1} t_{j_i} \\ &= t_0 - \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}_i} B_{(i,j)}^T K_{(i,j)}^{-1} t_{(i,j)} \\ &= t_Z, \end{aligned} \tag{25}$$

with

$$\begin{aligned}\bar{Z} &= K_0 - \sum_{i\in\mathcal{C}} \omega_i B_{j_i}^T K_{j_i}^{-1} B_{j_i} \\ &= K_0 - \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}_i} B_{j_i}^T K_{j_i}^{-1} B_{j_i}. \end{aligned} \tag{26}$$

Consequently, the Schur system of the preconditioner and of the KKT system have the same right-hand side. This property will be used to derive spectral properties and error bounds.

The use of the weighted matrix $K_{\mathcal{R}}^{\omega}$ and the correction term t_{CP} induces consistency between the right-hand sides of the Schur systems of the CP (25) and of the KKT system (19). In Section 3 we will see that this is key to establishing spectral and error properties for the preconditioner.

The main idea behind the CP is to compress the KKT system (16) into the smaller compressed system (23a), which is cheaper to factorize (e.g., using an LBL^T approach). We solve this smaller system to obtain q_0, and we recover q_Ω from (23b) by factorizing the individual blocks $K_{(i,j)}$. We refer to the coefficient matrix of (23a) as the *compressed matrix*.

In the following, we assume that the Schur complements Z and $\bar{Z}$ are nonsingular. The nonsingularity of Z together with the assumption that all the blocks $K_{(i,j)}$ are nonsingular implies (from the Schur complement theorem) that matrix K is nonsingular and thus the KKT system has a unique solution. The nonsingularity of $\bar{Z}$ together with the assumption that all the blocks $K_{(i,j)}$ are nonsingular implies that the compressed matrix is nonsingular and thus the CP has a unique solution. Note that we could have also assumed nonsingularity of matrix K directly and this, together with the nonsingularity of the blocks $K_{(i,j)}$, would imply nonsingularity of Z (this also from the Schur complement theorem). The same applies if we assume nonsingularity of the compressed matrix, which would imply nonsingularity of $\bar{Z}$.

Although Schur decomposition is a popular approach for solving structured KKT systems, it suffers from poor scalability with the dimension of q_0. The reason is that the Schur complement needs to be formed (this requires as many backsolves with the factors of $K_{(i,j)}$ as the dimension of q_0) and factorized (this requires a factorization of a dense matrix of dimension q_0). We elaborate on these scalability issues in Section 4. We thus highlight that the Schur system representations are used only for analyzing the CP.

Our preconditioning setting is thus the following. At each IP iteration k, we seek to compute a step by solving the KKT system (8). We do so by finding a solution vector $(\Delta w_0, \Delta w_\Omega)$ of the ordered KKT system (8) for the right-hand side (r_0, r_Ω) using an iterative linear algebra solver such as GMRES, QMR, or BICGSTAB. Here, (r_0, r_Ω) are the right-hand side vectors of the KKT system (8) in ordered form. Each minor iteration of the iterative linear algebra solver is denoted by $\ell = 0, 1, 2, ...,$. We denote the initial guess of the solution vector of (8) as $(\Delta w_0^\ell, \Delta w_\Omega^\ell)$ with $\ell = 0$. At each minor iterate ℓ, the iterative solver will request the application of the CP to a given vector $(t_0^\ell, t_\Omega^\ell)$, and the solution vectors $(q_0^\ell, q_\Omega^\ell)$ of (23) are returned to the iterative

linear algebra solver. Perfect preconditioning occurs when we solve (8) instead of (23) with the right-hand sides $(t_0^\ell, t_\Omega^\ell)$.

3 Preconditioner Properties

In this section we establish properties for the CP, and we use these to guide the design of appropriate clustering strategies.

We first note that any solution of the CP system (23a)-(23b) solves the perturbed KKT system,

$$\underbrace{\begin{bmatrix} K_\Omega & B_\Omega \\ B_\Omega^T & K_0 + E_Z \end{bmatrix}}_{:=\bar{K}} \begin{bmatrix} q_\Omega \\ q_0 \end{bmatrix} = \begin{bmatrix} t_\Omega \\ t_0 \end{bmatrix}, \tag{27}$$

where

$$E_Z := \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}_i} B_{(i,j)}^T K_{(i,j)}^{-1} B_{(i,j)} - \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}_i} B_{j_i}^T K_{j_i}^{-1} B_{j_i}, \tag{28}$$

and E_Z satisfies $\bar{Z} + E_Z = Z$. This mathematical equivalence between the CP system (23a)-(23b) and (27) can be established by constructing the Schur system of (27) and noticing that it is equivalent to (25). Moreover, the steps for the second-stage variables are the same. This equivalence enables us to establish the following result.

Lemma 1 *The preconditioned matrix $\bar{K}^{-1}K$ has $(n+m-n_0-m_0)$ unit eigenvalues, and the remaining (n_0+m_0) eigenvalues are bounded as*

$$|\lambda(\bar{K}^{-1}K) - 1| \leq \frac{1}{\sigma_{min}(\bar{Z})}\|E_Z\|.$$

Proof: The eigenvalues λ and eigenvectors $w := (w_\Omega, w_0)$ of $\bar{K}^{-1}K$ satisfy $\bar{K}^{-1}Kw = \lambda w$, and thus $Kw = \lambda \bar{K} w$. Consequently,

$$\begin{aligned} K_\Omega w_\Omega + B_\Omega w_0 &= \lambda(K_\Omega w_\Omega + B_\Omega w_0) \\ B_\Omega^T w_\Omega + K_0 w_0 &= \lambda B_\Omega^T w_\Omega + \lambda(K_0 + E_Z) w_0. \end{aligned}$$

From the first relationship we have $n + m - n_0 - m_0$ unit eigenvalues. Applying Schur decomposition to the eigenvalue system, we obtain

$$\begin{aligned} Z w_0 &= \lambda(Z + E_Z) w_0 \\ &= \lambda \bar{Z} w_0. \end{aligned}$$

We can thus express the remaining $n_0 + m_0$ eigenvalues of $\bar{K}^{-1}K$ as $\lambda = 1 + \epsilon_Z$ to obtain

$$\begin{aligned} |\epsilon_Z| &= \frac{\|E_Z w_0\|}{\|\bar{Z} w_0\|} \\ &\leq \frac{1}{\sigma_{min}(\bar{Z})}\|E_Z\|. \end{aligned}$$

The proof is complete. □

From the definition of E_Z we note that the following bound holds:

$$|\lambda(\bar{K}^{-1}K) - 1| \leq \frac{1}{\sigma_{min}(\bar{Z})} \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} \left\| B_{(i,j)}^T K_{(i,j)}^{-1} B_{(i,j)} - B_{j_i}^T K_{j_i}^{-1} B_{j_i} \right\|. \tag{30}$$

Thus, we can improve the spectrum of $\bar{K}^{-1}K$ by choosing clusters that minimize $\|E_Z\|$. This approach, however, would require expensive matrix operations. An interesting and tractable exception occurs in the special case in which $Q_{(i,j)} = Q$, $W_{(i,j)} = W$, and $T_{(i,j)} = T,\ i \in \mathcal{C}, j \in \mathcal{C}_i$. This case arises when the scenario data corresponds to the right-hand sides b_s and the cost coefficients d_s of (1) and is common in applications. In this case we have that E_Z reduces to

$$E_Z = \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} B^T \left(K_{(i,j)}^{-1} - K_{j_i}^{-1} \right) B. \tag{31}$$

We also have that $K_{(i,j)}$ and K_{j_i} differ only in the diagonal matrices $X_{(i,j)}^{-1} V_{(i,j)}$ and $X_{j_i}^{-1} V_{j_i}$. We thus have,

$$K_{(i,j)} - K_{j_i} = \begin{bmatrix} (X_{(i,j)}^{-1} V_{(i,j)} - X_{j_i}^{-1} V_{j_i}) & 0 \\ 0 & 0 \end{bmatrix}. \tag{32}$$

We define the vectors,

$$\gamma_{(i,j)} = \text{vec}(X_{(i,j)}^{-1} V_{(i,j)}), i \in \mathcal{C}, j \in \mathcal{C}_i \tag{33a}$$

$$\gamma_{j_i} = \text{vec}(X_{j_i}^{-1} V_{j_i}), i \in \mathcal{C}, \tag{33b}$$

and establish the following result.

Theorem 1 *Assume that $Q_{(i,j)} = Q$, $W_{(i,j)} = W$, and $T_{(i,j)} = T,\ i \in \mathcal{C}, j \in \mathcal{C}_i$ holds. Let vectors $\gamma_{(i,j)}, \gamma_{j_i}$ be defined as in* (33)*. The preconditioned matrix $\bar{K}^{-1}K$ has $(n+m-n_0-m_0)$ unit eigenvalues, and there exists a constant $c_K > 0$ such that the remaining $(n_0 + m_0)$ eigenvalues are bounded as*

$$|\lambda(\bar{K}^{-1}K) - 1| \leq \frac{c_K}{\sigma_{min}(\bar{Z})} \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{C}} \kappa_{s,i} \|\gamma_{j_i} - \gamma_s\|.$$

Proof: From Lemma 1 we have that $n_0 + m_0$ eigenvalues λ of $\bar{K}^{-1}K$ are bounded as $|\lambda - 1| \leq \frac{1}{\sigma_{min}(\bar{Z})} \|E_Z\|$. We define the error matrix,

$$E_{(i,j)} := K_{(i,j)} - K_{j_i},$$

and use (31) and (32) to obtain the bound,

$$\begin{aligned} \|E_Z\| &\leq \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} \|B^T B\| \|K_{(i,j)}^{-1} - K_{j_i}^{-1}\| \\ &= \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} \|B^T B\| \|(K_{j_i} + E_{(i,j)})^{-1} - K_{j_i}^{-1}\|. \end{aligned}$$

We have that

$$\begin{aligned} (K_{j_i} + E_{(i,j)})^{-1} - K_{j_i}^{-1} &= -K_{j_i}^{-1} (K_{j_i} + E_{(i,j)})^{-1} E_{(i,j)} \\ &= -K_{j_i}^{-1} K_{(i,j)}^{-1} E_{(i,j)}. \end{aligned}$$

This can be verified by multiplying both sides by $K_{j_i} + E_{(i,j)}$ and by exploiting symmetry of the matrices. We thus have

$$\begin{aligned}\|E_Z\| &\leq \sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{C}_i} \|B^TB\|\|(K_{j_i}+E_{(i,j)})^{-1} - K_{j_i}^{-1}\| \\ &\leq \sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{C}_i} \|B^TB\|\|K_{(i,j)}^{-1}K_{j_i}^{-1}\|\|E_{(i,j)}\| \\ &\leq c_K \sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{C}_i} \|\text{vec}(X_{j_i}^{-1}V_{j_i}) - \text{vec}(X_{(i,j)}^{-1}V_{(i,j)})\|,\end{aligned}$$

with $c_K := \sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{C}_i}\|B^TB\|\|(K_{(i,j)}K_{j_i})^{-1}\|$. The existence of c_K follows from the nonsingularity of $K_{(i,j)}$ and K_{j_i}. The proof is complete. □

We now develop a bound of the preconditioning error for the general case in which the scenario data also defines the coefficient matrices. Notably, this bound does not require the minimization of the error $\|E_Z\|$. The idea is to bound the error induced by the CP on the exact solution of the KKT system (16) (perfect preconditioner). This approach is used to characterize inexact preconditioners such as multigrid and nested preconditioned conjugate gradient [21]. We express the solution of CP obtained from (23) as $q^T = [q_\Omega^T, q_0^T]$ and that of the KKT system (16) as $q^{*T} = [q_\Omega^{*T}, q_0^{*T}]$. We define the error between q and q^* as $\epsilon := q - q^*$ and we seek to bound ϵ. If we decompose the error as $\epsilon^T = [\epsilon_\Omega^T, \epsilon_0^T]$, we have that $\epsilon_0 = q_0 - q_0^*$ and $\epsilon_\Omega = q_\Omega - q_\Omega^*$.

We recall that the Schur systems of (16) and of (23) and their respective solutions satisfy

$$Zq_0^* = t_Z \tag{34a}$$
$$\bar{Z}q_0 = t_Z. \tag{34b}$$

We also define the vectors,

$$\gamma_{(i,j)} = (B_{(i,j)}^T K_{(i,j)}^{-1} B_{(i,j)})t_Z, i \in \mathcal{C}, j \in \mathcal{C}_i \tag{35a}$$
$$\gamma_{j_i} = (B_{j_i}^T K_{j_i}^{-1} B_{j_i})t_Z, i \in \mathcal{C}. \tag{35b}$$

We now establish a bound on the error $\epsilon = q - q^*$.

Lemma 2 *There exists $c_{ZK} > 0$ such that the preconditioner error ϵ is bounded as*

$$\|\epsilon\| \leq c_{ZK}\|Zt_Z - \bar{Z}t_Z\|.$$

Proof: From $\epsilon_0 = q_0 - q_0^*$ we have $\bar{Z}\epsilon_0 = \bar{Z}q_0 - \bar{Z}q_0^*$. From (34) we have $\bar{Z}q_0 = Zq_0^* = t_Z$ and thus $\bar{Z}\epsilon_0 = Zq_0^* - \bar{Z}q_0^*$. Multiplying this expression through by Z and exploiting symmetry of Z and $\bar{Z}$, we obtain

$$\begin{aligned}\bar{Z}Z\epsilon_0 &= ZZq_0^* - \bar{Z}Zq_0^* \\ &= Zt_Z - \bar{Z}t_Z,\end{aligned}$$

where the second equality follows from (34). We recall that

$$q_\Omega^* = K_\Omega^{-1}(t_\Omega - B_\Omega q_0^*)$$
$$q_\Omega = K_\Omega^{-1}(t_\Omega - B_\Omega q_0)$$

and thus

$$\begin{aligned}\epsilon_\Omega &= K_\Omega^{-1} B_\Omega (q_0^* - q_0) \\ &= -K_\Omega^{-1} B_\Omega \epsilon_0.\end{aligned}$$

We thus have

$$\begin{aligned}\|\epsilon_0\| &\leq c_Z \|Zt_Z - \bar{Z}t_Z\| \\ \|\epsilon_\Omega\| &\leq c_{K_\Omega} \|\epsilon_0\|,\end{aligned}$$

with $c_Z := \|(\bar{Z}Z)^{-1}\|$ and $c_{K_\Omega} := \|K_\Omega^{-1} B_\Omega\|$. The existence of c_Z follows from the assumption that Z and $\bar{Z}$ are nonsingular. The existence of c_Ω follows from the assumption that the blocks $K_{(i,j)}$ are nonsingular and thus K_Ω is nonsingular. The result follows from $\|\epsilon\| \leq \|\epsilon_0\| + \|\epsilon_\Omega\|$ and by defining $c_{ZK} := c_Z(1 + c_{K_\Omega})$. □

Theorem 2 *Let vectors $\gamma_{(i,j)}, \gamma_{j_i}$ be defined as in* (35)*. The preconditioner error ϵ is bounded as*

$$\|\epsilon\| \leq c_{ZK} \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{C}} k_{s,i} \|\gamma_{j_i} - \gamma_s\|,$$

with c_{ZK} defined in Lemma 2.

Proof: From (35) and (28) we have that

$$\begin{aligned}\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} (\gamma_{(i,j)} - \gamma_{j_i}) &= Zt_Z - \bar{Z}t_Z \\ &= E_Z t_Z.\end{aligned}$$

This follows from

$$\begin{aligned}Zt_Z - \bar{Z}t_Z &= \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} B_{(i,j)}^T K_{(i,j)}^{-1} B_{(i,j)} t_Z - \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} B_{j_i}^T K_{j_i}^{-1} B_{j_i} t_Z \\ &= \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} (B_{(i,j)}^T K_{(i,j)}^{-1} B_{(i,j)} t_Z - B_{j_i}^T K_{j_i}^{-1} B_{j_i} t_Z) \\ &= \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} (\gamma_{(i,j)} - \gamma_{j_i}).\end{aligned}$$

Bounding, we have

$$\begin{aligned}\|Zt_Z - \bar{Z}t_Z\| &= \left\| \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} (\gamma_{j_i} - \gamma_{(i,j)}) \right\| \\ &\leq \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} \|\gamma_{j_i} - \gamma_{(i,j)}\| \\ &= \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{C}} \kappa_{s,i} \|\gamma_{j_i} - \gamma_s\|.\end{aligned}$$

The result follows from Lemma 2. □

Theorems 1 and 2 provide the necessary insights to derive clustering strategies. We can see that the properties of CP are related to a metric of the form

$$\mathcal{D}_C := \sum_{s\in\mathcal{S}} \sum_{i\in\mathcal{C}} \kappa_{s,i} \|\gamma_{j_i} - \gamma_s\|. \tag{36}$$

This is the *distortion metric* widely used in clustering analysis [2]. The distortion metric is (partially) minimized by k-means and hierarchical clustering algorithms to determine $\kappa_{s,i}$ and γ_{j_i}. The vectors γ_s are called *features*, and γ_{j_i} is the centroid of cluster $i \in \mathcal{C}$ (we can also pick the scenario that is closest to the centroid). The distortion metric is interpreted as the accumulated distance of the elements of the cluster relative to the centroid. If the distortion is small, then the scenarios in a cluster are similar. The distortion metric can be made arbitrarily small by increasing the number of clusters and is zero in the limit with $S = C$ because each cluster is given by one scenario.

Theorem 1 states that in the special case where the scenario data enters only in the right-hand sides and cost coefficients and if the features are defined as $\gamma_s = \text{vec}(X_s^{-1}V_s)$, the spectrum of $\bar{K}^{-1}K$ can be made arbitrarily close to one if the distortion is made arbitrarily small. This thus guarantees that the *definition of the features is consistent.* The scenarios are clustered at each IP iteration k because the matrices $X_s^{-1}V_s$ change along the search (even if the matrices Q_s, W_s, T_s do not). The clustering approach is therefore adaptive, unlike outside-the-solver scenario clustering approaches. Note that even if the data enters only in the right-hand side, our proposed clustering approach does not cluster the problem data. In fact, it does not seem possible to derive spectral and error properties for preconditioners based on clustering of problem data. Our approach focuses directly on the contributions $X_s^{-1}V_s$ and thus assumes that the problem data enters indirectly through the contributions $X_s^{-1}V_s$, which in turn affect the structural properties of the KKT matrix.

The features $\gamma_s = \text{vec}(X_s^{-1}V_s)$ have an important interpretation: these reflect the contribution of each scenario to the logarithmic barrier function. If $\|\gamma_s\| \approx 0$, then this implies that $\|\nu_s\| \approx 0$; therefore there is weak activity in this scenario, and the scenario is not relevant. On the other hand, when $\|\gamma_s\| \gg 0$, we have that the scenario has strong activity and is thus relevant. Instead of eliminating the scenarios with weak activity, as proposed in [22,11], we cluster scenarios with similar activities. This approach allows us to eliminate redundancies in both active and inactive scenarios and to capture outliers. In addition, this strategy bypasses the need to specify a suitable threshold to classify weak and strong activity.

Theorem 2 provides a mechanism to obtain clusters for the general case in which the scenario data also defines the coefficient matrices. The result states that we can bound the preconditioning error using the Schur complement error $E_Z = Z - \bar{Z}$ projected on the right-hand side vector t_Z. Consequently, the error can be bounded by the distortion metric with features defined in (35). This implies that the error can be made arbitrarily small if the distortion is made arbitrarily small and it is not necessary to perform major matrix operations.

The error bound of Theorem 2 requires that clustering tasks and the factorization of the compressed matrix be performed at each minor iteration ℓ of the iterative linear algebra solver. The reason is that the features (35) change with t_Z^ℓ. Performing these tasks at each minor iteration, however, is expensive. Consequently, we perform these tasks only at the first minor iteration $\ell = 0$. If the initial guess of the solution vector of the KKT system is set to zero ($\Delta w_0^\ell = 0$ and $\Delta w_\Omega^\ell = 0$) and if GMRES, QMR, or BICGSTAB schemes are used, this is equivalent to performing by clustering using the features

$$\gamma_{(i,j)} = (B_{(i,j)}^T K_{(i,j)}^{-1} B_{(i,j)}) r_Z, i \in \mathcal{C}, j \in \mathcal{C}_i \tag{37a}$$

$$\gamma_{j_i} = (B_{j_i}^T K_{j_i}^{-1} B_{j_i}) r_Z, i \in \mathcal{C}. \tag{37b}$$

where

$$\begin{aligned} r_Z &= t_Z^0 \\ &= r_0 - \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} B_{(i,j)}^T K_{(i,j)}^{-1} r_{(i,j)} \end{aligned} \tag{38}$$

is the right-hand side of the Schur system corresponding to the KKT system (8).

4 Numerical Results

In this section we discuss implementation issues of CP and present numerical results for benchmark problems in the literature and a large-scale stochastic market clearing problem.

We summarize the procedure for computing the step $(\Delta x_k, \Delta y_k, \Delta \nu_k)$ at each IP iteration k in the following scheme.

Step Computation Scheme

1. **Initialization**. Given iterate (x_k, y_k, ν_k), number of clusters C, tolerance τ_k, and maximum number of linear solver iterates m_{it}.
2. **Get Clustering Information.**
   - 2.0. Compute features $\gamma_s,\ s \in \mathcal{S}$ as in (33) or (37).
   - 2.1. Obtain $\kappa_{s,i}$ and γ_{j_i} using K-means or hierarchical clustering.
   - 2.2. Use $\kappa_{s,i}$ to construct $\mathcal{C}$, $\mathcal{R}$, Ω, $\mathcal{C}_i$, and ω_i.
   - 2.3. Construct and factorize compressed matrix

$$\begin{bmatrix} K_{\mathcal{R}}^{1/\omega} & B_{\mathcal{R}} \\ B_{\mathcal{R}}^T & K_0 \end{bmatrix}$$

     and factorize scenario matrices $K_{(i,j)},\ i \in \mathcal{C}, j \in \mathcal{C}_i$.
3. **Get Step.**
   - 3.1. Call iterative linear solver to solve KKT system (16) with right-hand sides (r_0, r_Ω), set $\ell = 0$, and initial guess $\Delta w_0^\ell = 0$ and $\Delta w_\Omega^\ell = 0$. At each minor iterate $\ell = 0, 1, ...,$ of the iterative linear solver, DO:
     - 3.1.1. Use factorization of compressed matrix and of K_Ω to solve CP (23a)-(23b) for right-hand sides $(t_0^\ell, t_\Omega^\ell)$ and RETURN solution $(q_0^\ell, q_\Omega^\ell)$.
     - 3.1.2. From (18), get ϵ_r^ℓ using solution vector $(\Delta w_0^\ell, \Delta w_\Omega^\ell)$ and right-hand side vectors (r_0, r_Ω). If $\|\epsilon_r^\ell\| \leq \tau_k$, TERMINATE.
     - 3.1.3. If $\ell = m_{it}$, increase C, and RETURN to Step 1.
   - 3.2. Recover $(\Delta x_k, \Delta y_k)$ from $(\Delta w_0^\ell, \Delta w_\Omega^\ell)$.
   - 3.3. Recover $\Delta \nu_k$ from (6).

We call our clustering-based IP implementation `IP-CLUSTER`. In this implementation we use the primal-dual IP framework of Mehrotra [17]. We use the matrix templates and direct linear algebra routines of the `BLOCK-TOOLS` library. This library is specialized to block matrices such as the KKT matrices used in this work and greatly facilitated the implementation. Within `BLOCK-TOOLS`, we use its MA57 interface to perform all direct linear algebra operations. We use the `GMRES` implementation within the `PETSc` library (`http://www.mcs.anl.gov/petsc`) to perform all iterative linear algebra operations. We have implemented serial and parallel versions of CP. The parallel version performs the factorizations of (23b) in parallel and exploits the block-bordered-diagonal structure of the KKT matrix to perform matrix-vector operations in parallel. We use the K-means and hierarchical clustering implementations

of the C-Clustering library (http://bonsai.hgc.jp/~mdehoon/software/cluster/software.htm). To implement the market clearing models we use an interface to AMPL to create individual instances (.nl files) for each scenario and indicate first-stage variables and constraints using the suffix capability.

4.1 Benchmark Problems

We consider stochastic variants of problems obtained from the CUTEr library and benchmark problems (SSN, GBD, LANDS, 20TERM) reported in [15]. The deterministic CUTEr QP problems have the form

$$\min \frac{1}{2} y^T Q y + d^T y, \text{ s.t. } Ay = b,\ y \geq 0. \tag{39}$$

We generate a stochastic version of this problem by defining b as a random vector. We create scenarios for this vector $b_s, s \in \mathcal{S}$ using the nominal value b as mean and a standard deviation $\pm\sigma = 0.5b$. We then formulate the two-stage stochastic program:

$$\min\ e^T y_0 + \sum_{s\in\mathcal{S}} \frac{1}{2} y_s^T Q y_s + d^T y_s \tag{40a}$$

$$\text{s.t.} \quad Ay_s = b_s, s \in \mathcal{S} \tag{40b}$$

$$y_s + y_0 \geq 0,\ s \in \mathcal{S} \tag{40c}$$

$$y_0 \geq 0. \tag{40d}$$

We first demonstrate the quality of CP in terms of the number of GMRES Iterations. For all cases, we assume a scenario compression rate of 75% (only 25% of the scenarios are used in the compressed matrix), and we solve the problems to a tolerance of 1×10^{-6}. In all the numerical experiments we do not adaptively modify the number of clusters. We set the maximum number of GMRES iterations to 100 (if the preconditioner reaches this limit the IP solver is terminated). This is done in order to profile scalability of the preconditioner in a more systematic manner.

For this first set of results, we cluster the scenarios using the features (35) and hierarchical clustering. The results are presented in Table 1. The performance of CP is satisfactory in all instances, requiring fewer than 20 GMRES iterations per IP iteration (this is labeled as GMRES/IP Iter). We attribute this to the particular structure of CP, which enable us to pose it in the equivalent form (27) and to derive favorable spectral properties and error bounds. To support these observations, we have also experimented with a *naive* preconditioner of the form:

$$\begin{bmatrix} \bar{K}_\Omega & \bar{B}_\Omega \\ \bar{B}_\Omega^T & K_0 \end{bmatrix} \begin{bmatrix} q_\Omega \\ q_0 \end{bmatrix} = \begin{bmatrix} t_\Omega \\ t_0 \end{bmatrix}, \tag{41}$$

where,

$$\bar{K}_\Omega := \text{blkdiag}\left\{K_{j_1}, ..., K_{j_1}, K_{j_2}, ..., K_{j_2}, ..., K_{j_C}, ..., K_{j_C}\right\} \tag{42a}$$

$$\bar{B}_\Omega := \text{rowstack}\left\{B_{j_1}, ..., B_{j_1}, B_{j_2}, ..., B_{j_2}, ..., B_{j_C}, ..., B_{j_C}\right\}. \tag{42b}$$

The naive preconditioner replaces the block matrix elements of the cluster with those of the scenario representing the cluster. The right-hand sides of the naive preconditioner, however, are consistent with those of the KKT system. The Schur system of the naive preconditioner has the form

$$\bar{Z} q_0 = t_0 - \sum_{i\in\mathcal{C}} \sum_{i\in\mathcal{C}_i} B_{j_i}^T K_{j_i}^{-1} t_{(i,j)}. \tag{43}$$

Note that this system has the same Schur matrix as that of the Schur system of CP (25). The naive preconditioner, however, does not have the same right-hand side of the Schur system of CP and of the KKT system (see (34)). Moreover, the steps for the second-stage variables obtained with the naive preconditioner are:

$$K_{j_i} q_{(i,j)} = t_{(i,j)} - B_{j_i} q_0, \quad i \in \mathcal{C},\ j \in \mathcal{C}_i. \tag{44}$$

By comparing (44) with (23b) we can see that the recovery of second-stage variables in the naive approach does not use the actual second-stage matrices $K_{(i,j)}, B_{(i,j)}$ corresponding to each scenario as is done in the CP approach. The structural deficiencies of the naive preconditioner prevent us from obtaining the error bounds of Lemma 2 and Theorem 2 and highlight the importance of CP structure. In Table 1 we can in fact see that the performance of the naive preconditioner is not competitive. In particular, the naive approach only outperforms the CP approach in a couple of instances.

Table 1 Performance of naive and CP preconditioners naive in benchmark problems.

			Naive (75%)			CP (75%)		
Problem	S	n	IP Iter	GMRES Iter	GMRES/IP Iter	IP Iter	GMRES Iter	GMRES/IP Iter
HS53	100	1,010	27	911	33	19	113	5
LOTSCHD	100	1,212	24	626	26	25	203	8
HS76	100	707	19	152	8	23	98	4
HS118	100	5,959	47	1499	31	47	409	8
QPCBLEND	100	11,514	57	258	4	57	253	4
ZECEVIC2	100	606	27	451	16	29	111	3
QPTEST	100	505	23	569	24	23	108	4
SSN	100	70,689	114	738	6	114	1857	16
GBD	1000	10,017	24	627	26	24	144	6
LANDS	1000	12,004	29	481	16	29	115	3
20TERM	100	76,463	57	581	10	57	976	17

In Table 2 we compare the performance of CP with that of the unpreconditioned strategy (compression of 100%) and naive strategy for a single problem instance. We perform this comparison to illustrate that the matrices of the benchmark problems are nontrivial and preconditioning is indeed needed. We use the notation x% to indicate the compression rate (i.e., the preconditioner uses 100-x% of the scenarios). A compression of 0% indicates the entire scenario set is used for the preconditioner (ideal).

In Table 3 we compare the effect of the compression rate and the number of scenarios on the performance of CP for problem 20TERM. For a fixed number of scenarios, the performance of CP deteriorates as we increase the compression rate. The performance is improved, however, for a fixed compression rate as we increase the number of scenarios. We have also found that the deterioration of performance due to increasing compression rates becomes less pronounced as we increase the number of scenarios. The reason is that more redundancy is observed as we increase the number of scenarios and, consequently, compression potential increases. This behavior has been found in several instances and indicates that once can deal with problems with a large number of scenarios.

In Table 4 we isolate the effect of different clustering strategies on preconditioning performance. We perform clustering using features (33) (we label this as $X^{-1}V$) and (37) (we label this as r_Z). In these tests we also illustrate the computational performance of CP compared with full factorization for two large instances. Instance QSC2015 has 63,717 variables, while instance

Table 2 Performance of preconditioned and unpreconditioned strategies.

Problem	S	n	Compression	IP Iter	GMRES Iter	GMRES/IP Iter
HS53	100	1,010	100%	19	12861	676
			75% (Naive)	27	911	33
			75% (CP)	19	113	5

Table 3 Effect of compression rates on 20TERM problem.

S	n	Compression	IP Iter	GMRES/IP Iter
100	76,463	50%	57	10
		75%	57	19
		87%	57	17
200	152,863	50%	72	9
		75%	72	14
		87%	72	18
400	305,663	50%	92	20
		75%	92	21
		87%	92	23
800	611,263	50%	97	25
		75%	97	25
		87%	97	27

AUG3DC has 131,682 variables. We use θ_{tot} to denote the total solution time, θ_{fact} to denote the time spent in factorizing the compressed matrix and the block matrices, θ_{clus} to denote the time spent performing clustering operations, θ_{gmres} to denote the time spent in GMRES Iterations (without considering factorization of preconditioner). As can be seen from Table 4, the performance of CP is identical for both clustering strategies, and the solution times are dramatically reduced.

Table 4 Performance of different clustering strategies for benchmark problems.

Problem	Compress.	Clustering	IP Iter	θ_{tot}	θ_{fact}	θ_{clus}	θ_{gmres}	GMRES Iter	GMRES/IP Iter
QSC205	0%		110	1331	1321				
	50%	$X^{-1}V$	110	220	157	5	42	747	6
	75%	$X^{-1}V$	110	91	25	6	45	933	8
	50%	r_Z	110	229	161	5	43	747	6
	75%	r_Z	110	89	24	5	43	924	8
AUG3DC	0%		11	1427	1423				
	50%	$X^{-1}V$	11	96	84	0.3	6	26	2
	75%	$X^{-1}V$	11	24	13	0.3	6	27	2
	50%	r_Z	11	93	80	0.3	6	26	2
	75%	r_Z	11	25	13	0.3	6	27	2

4.2 Stochastic Market Clearing Problem

To demonstrate the computational efficiency of the preconditioner, we solve the stochastic market-clearing model presented in [19] for the entire Illinois power grid system [23]. The system is

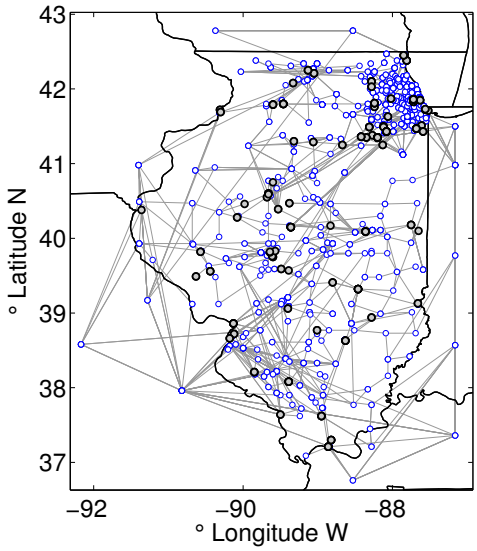

Fig. 1 Illinois transmission system. Dark dots are generation nodes and blue dots are demand nodes.

illustrated in Figure 1. The stochastic programming formulation is given by

$$\min_{x_i, X_i(\omega)} \sum_{i \in \mathcal{G}} \left(p_i x_i + \mathbb{E}_\omega \left[p_i^+ (X_i(\omega) - x_i)_+ - p_i^- (X_i(\omega) - x_i)_- \right] \right)$$

$$\text{s.t. } \tau_n(f) + \sum_{i \in \mathcal{G}(n)} x_i = d_n, \ n \in \mathcal{N} \tag{45a}$$

$$\tau_n(F(\omega)) - \tau_n(f) + \sum_{i \in \mathcal{G}(n)} (X_i(\omega) - x_i) = 0, \ \ n \in \mathcal{N}, \omega \in \Omega \tag{45b}$$

$$f, F(\omega) \in \mathcal{F}, \ \ \omega \in \Omega \tag{45c}$$

$$(x_i, X_i(\omega)) \in \mathcal{X}_i(\omega), \ \ i \in \mathcal{G}, \omega \in \Omega. \tag{45d}$$

Here, $\mathcal{N}$ denotes the set of network nodes and $\mathcal{L}$ the set of transmission lines. The set of all suppliers is denoted by $\mathcal{G}$. Subsets $\mathcal{G}(n)$ denote the set of players connected to node n. The forward (first-stage) dispatched quantities for players are x_i, and the spot (second-stage) quantities under scenario ω are $X_i(\omega)$. The forward power flow through line $\ell \in L$ is denoted by f_ℓ, and f denotes the vector of all line flows. Similarly, $F(\omega)$ denotes the vector of line flows $F_\ell(\omega)$ for each scenario ω. The demand is assumed to be deterministic and inelastic and is represented by $d_n, \ n \in N$. The sets $\mathcal{F}$ and $\mathcal{X}_i(\omega)$ are polyhedral and defined by lower and upper bounds for the flows and dispatch quantities. The objective of the market clearing problem is to minimize the forward dispatch cost plus the expected recourse dispatch cost. Here $[y]_+ = \max\{y, 0\}$ and $[y]_- = \max\{-y, 0\}$. The coefficients p_i denote the bid price, and p_i^+ and p_i^- are price bids for corrections of the generators. A supplier i asks $p_i^+ > p_i$ to sell additional power or asks $p_i^- < p_i$ to buy power from the system (e.g., reduces output). The scenarios ω characterize the randomness in the model due to unpredictable supply capacities (in this case wind power). We use a sample-average approximation of the problem to obtain a deterministic equivalent.

The market clearing model has *large first-stage dimensionality*. For the Illinois system we have 11,891 first-stage variables, 6,704 equality constraints, and 11,891 inequality constraints. This

Table 5 Performance of Schur decomposition approach.

S	n	IP Iter	θ_{tot}	$\theta_{factschur}$	$\theta_{formschur}$	$\theta_{factblock}$
1	30,472	55	31280	27236	4023	4

gives a Schur complement with saddle-point structure of dimension 64,199, which is sparse but contains a dense block of dimension 11,891.

In Table 5 we present the solution times for this problem using a Schur decomposition strategy for *a single scenario*. Here, θ_{tot} is the total solution time, $\theta_{factschur}$ is the time spent factorizing the Schur complement, $\theta_{formschur}$ is the time spent forming the Schur complement, and $\theta_{factblock}$ is the time spent factorizing the scenario block. All times are reported in seconds. The solution time for this problem is 3.61 hr, with 25% of the time spent forming the Schur complement and 75% spent factorizing the Schur complement. Note that if more scenarios are added, the time spent forming and factorizing the Schur complement will dominate (even if the scenarios can be parallelized). Iterative strategies applied to the Schur complement system can avoid the time spent forming the Schur complement but not the factorization time because a preconditioner with a large dense block still needs to be factorized [18].

We now assess the serial performance of CP. The results are presented in Table 6. By comparing Tables 5 and 6 we can see that the full sparse factorization approach will be as efficient as Schur decomposition for problems with up to 64 scenarios. In other words, it would be faster to factorize the full sparse KKT system than forming and factorizing the large Schur complement. The fast growth in solution time of the full factorization approach is remarkable, however. We attribute this to the tight connectivity induced by the network constraints which introduce significant fill-in. The CP reduces the solution times of full factorization by a factor of 2 for the problem with 32 scenarios and by a factor of 8 for the problem with 64 scenarios. We highlight that CP is highly effective, requiring on average 5 to 10 GMRES Iterations per IP iteration for compression rates of 50% and 11 to 13 iterations for compression rates of 75%. We also observe that the performance of different clustering strategies is similar.

For the problem with 64 scenarios we can see that the solution time of CP increases as we increase the compression rate from 75% to 87% even if the factorization time is dramatically reduced. This is because the time spent in GMRES to perform backsolves and matrix-vector operations dominates the factorization time. This is mitigated by using a parallel implementation of CP. The results are presented in Table 7. We can see that the solution time spent in GMRES to perform backsolves and matrix-vector operations is dramatically reduced by exploiting the block-bordered-diagonal structure of the KKT matrix. *This enables us to solve a market clearing problem with over 1.2 million variables in 10 minutes as opposed to 9 hours using the full factorization approach. This represents a speed up factor of 42*. By comparing the parallel results with those of Table 5 we can also see that the Schur complement approach is not competitive because of the time needed to form and factorize the Schur complement (this holds even for a single scenario).

From Table 7 we can see that scalability slows down as we increase the number of processes. This is because the remaining serial components (beyond backsolves and matrix-vector operations) of CP start dominating. This mainly include operations inside the GMRES algorithm itself. We are currently investigating ways to parallelize these operations.

In Table 7 we also present experiments using a compression rate of 94%. We performed these experiments to explore the performance limit of CP. We can see that the performance of CP deteriorates in terms of total solution time because the number of GMRES Iterations (and thus time) increases. Consequently, it does not pay off to cluster the KKT system further. We highlight, however, that the deterioration of CP in terms of GMRES Iterations is *graceful*. In

Table 6 Serial performance of CP preconditioner against full factorization for stochastic market clearing problem.

S	n	Compress.	Cluster.	IP Iter	θ_{tot}	θ_{fact}	θ_{clus}	θ_{gmres}	GMRES Iter	GMRES/IP Iter
		0%		57	473	452				
16	309,187	50%	$X^{-1}V$	57	544	119	0.4	325	631	11
		50%	r_Z	57	508	117	0.15	296	519	9
		0%		65	3480	3414				
		50%	$X^{-1}V$	65	1477	661	8	606	574	8
32	606,483	75%	$X^{-1}V$	65	1479	145	8	1141	1194	18
		50%	r_Z	65	1347	672	3	459	398	6
		75%	r_Z	65	1131	150	3	769	804	12
		0%		64	28022	27883				
		50%	$X^{-1}V$	64	5163	3513	29	1292	660	10
		75%	$X^{-1}V$	64	2878	656	29	1844	902	14
64	1,201,075	87%	$X^{-1}V$	64	2499	135	29	1990	1040	16
		50%	r_Z	64	5238	3492	12	1349	666	10
		75%	r_Z	64	3003	659	12	1924	937	14
		87%	r_Z	64	2440	115	12	1944	1147	17

Table 7 Parallel performance of CP preconditioner against full factorization for stochastic market clearing problem.

S	n	Processors	Compress.	Cluster.	IP Iter	θ_{tot}	θ_{fact}	θ_{clus}	θ_{gmres}	GMRES Iter	GMRES/IP Iter
		1	0%		64	**28022**	27883				
		1	87%	r_Z	64	2440	115	12	1944	1147	17
		2	87%	r_Z	64	1211	116	12	892	1025	16
		4	87%	r_Z	64	817	134	12	592	919	14
		8	87%	r_Z	64	**658**	152	12	398	905	14
64	1,201,075	1	94%	r_Z	64	3223	49	12	2764	1489	23
		2	94%	r_Z	64	1558	43	12	1306	1471	22
		4	94%	r_Z	64	993	49	12	801	1420	22
		8	94%	r_Z	64	**733**	54	12	570	1409	22

particular, it is remarkable that, on average, the linear system can be solved in 22 GMRES Iterations even if only four scenarios are used. This behavior indicates that the computation of the second-stage variables in (23b) plays a key role in the performance of CP. The naive preconditioner does not work in any of the market clearing cases presented (reaches the GMRES iteration limit of 100).

5 Conclusions and Future Work

We have presented a preconditioning strategy for stochastic programs using clustering techniques. This inside-the-solver clustering strategy can be used as an alternative to (or in combination with) outside-the-solver scenario aggregation and clustering strategies. A practical feature of performing inside-the-solver clustering is that no information on probability distributions is required and a single problem is solved. We have demonstrated that the preconditioners can be implemented in sparse form and dramatically reduce computational time compared to full factorizations of the KKT system. We have also demonstrated that the sparse form enables the solution of problems with large first-stage dimensionality that cannot be addressed with Schur decomposition. Scenario compression rates of up to 87% have been observed in large problem instances. As part of future work, we will investigate the performance of the preconditioner in a nonlinear programming setting and we will investigate extensions to multi-stage stochastic programs.

Acknowledgements This material was based upon work supported by the U.S. Department of Energy, Office of Science, under Contract No. DE-AC02-06CH11357. Funding from the Office of Science under the Early Career program is acknowledged. Victor M. Zavala thanks Jacek Gondzio for providing feedback on a previous version of the manuscript.

References

1. J. Birge. Aggregation bounds in stochastic linear programming. *Mathematical Programming*, 31:25–41, 1985.
2. C.M. Bishop et al. *Pattern recognition and machine learning*, volume 4. Springer, New York, 2006.
3. R. H Byrd, G. M. Chin, W. Neveitt, and J. Nocedal. On the use of stochastic hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3):977–995, 2011.
4. G. C. Calafiore and M. C. Campi. The scenario approach to robust control design. *Automatic Control, IEEE Transactions on*, 51(5):742–753, 2006.
5. M.S. Casey and S. Sen. The scenario generation algorithm for multistage stochastic linear programming. *Mathematics of Operations Research*, 30(3):615–631, 2005.
6. N. Chiang and A. Grothey. Solving security constrained optimal power flow problems by a structure exploiting interior point method. *Submitted For Publication*, 2012.
7. M. Colombo, J. Gondzio, and A. Grothey. A warm-start approach for large-scale stochastic linear programs. *Mathematical Programming*, 127(2):371–397, 2011.
8. W. L. de Oliveira, C. Sagastizábal, D. Penna, M. Maceira, and J. M. Damázio. Optimal scenario tree reduction for stochastic streamflows in power generation planning problems. *Optimization Methods and Software*, 25(6):917–936, 2010.
9. J. Dupačová, N. Gröwe-Kuska, and W. Römisch. Scenario reduction in stochastic programming. *Mathematical programming*, 95(3):493–511, 2003.
10. M. C. Ferris and T. S. Munson. Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804, 2002.
11. J. Gondzio and A. Grothey. Reoptimization with the primal-dual interior point method. *SIAM Journal on Optimization*, 13:842–864, 2003.
12. H. Heitsch and W. Römisch. Scenario tree reduction for multistage stochastic programs. *Computational Management Science*, 6:117–133, 2009.
13. J. Jung, D. P. Oleary, and A. L. Tits. Adaptive constraint reduction for training support vector machines. *Electronic Transactions on Numerical Analysis*, 31:156–177, 2008.
14. J. M. Latorre, S. Cerisola, and A. Ramos. Clustering algorithms for scenario tree generation: Application to natural hydro inflows. *European Journal of Operational Research*, 181(3):1339 – 1353, 2007.
15. J. Linderoth, A. Shapiro, and S. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241, 2006.
16. M. Lubin, C. G. Petra, M. Anitescu, and V. M. Zavala. Scalable stochastic optimization of complex energy systems. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–10. IEEE, 2011.
17. S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2:575–601, 1992.
18. C. Petra and M. Anitescu. A preconditioning technique for schur complement systems arising in stochastic optimization. *Computational Optimization and Applications*, 52:315–344, 2012.
19. G. Pritchard, G. Zakeri, and A. Philpott. A single-settlement, energy-only electric power market for unpredictable and intermittent participants. *Operations Research*, 58(4-part-2):1210–1219, 2010.
20. C. M. Shetty and R. W. Taylor. Solving large-scale linear programs by aggregation. *Computers & Operations Research*, 14(5):385 – 393, 1987.
21. D. B. Szyld and J. A. Vogel. Fqmr: A flexible quasi-minimal residual method with inexact preconditioning. *SIAM Journal on Scientific Computing*, 23(2):363–380, 2001.
22. A. Tits, P. Absil, and W. Woessner. Constraint reduction for linear programs with many inequality constraints. *SIAM Journal on Optimization*, 17(1):119–146, 2006.
23. V. M. Zavala, A. Botterud, E. M. Constantinescu, and J. Wang. Computational and economic limitations of dispatch operations in the next-generation power grid. *IEEE Conference on Innovative Technologies for and Efficient and Reliable Power Supply*, 2010.
24. V. M. Zavala, E. M. Constantinescu, T. Krause, and M. Anitescu. On-line economic optimization of energy systems using weather forecast information. *Journal of Process Control*, 19(10):1725–1736, 2009.
25. P.H. Zipkin. Bounds for row-aggregation in linear programming. *Operations Research*, 28(4):903–916, 1980.