Scenario grouping and decomposition algorithms for chance-constrained programs

Yan Deng∗ Shabbir Ahmed† Jon Lee‡ Siqian Shen§

Abstract

A lower bound for a finite-scenario chance-constrained problem is given by the quantile value corresponding to the sorted optimal objective values of scenario subproblems. This quantile bound can be improved by grouping subsets of scenarios at the expense of larger subproblems. The quality of the bound depends on how the scenarios are grouped. We formulate a mixed-integer bilevel program that optimally groups scenarios to tighten the quantile bounds. For general chance-constrained programs we propose a branch-and-cut algorithm to optimize the bilevel program, and for chance-constrained linear programs, we derive a mixed-integer linear programming reformulation. We also propose several heuristics for grouping similar or dissimilar scenarios. Our computational results show that optimal grouping bounds are much tighter than heuristic bounds, resulting in smaller root node gaps and better performance of the scenario decomposition algorithm for chance-constrained 0-1 programs. Moreover, the bounds from feasible grouping solutions obtained after solving the optimal grouping model for 20%-50% of the total time are sufficiently tight, having gaps under 10% of the corresponding optimal grouping bounds. They outperform heuristic grouping bounds both in tightness and solving time, and can be significantly strengthened using larger group size.

Key words: chance-constrained programming; quantile bounds; scenario grouping; mixed-integer programming; branch-and-cut; scenario decomposition

∗Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA 48109; email: yandeng@umich.edu
†School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA 30332; email: sahmed@isye.gatech.edu
‡Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA 48109; email: jonxlee@umich.edu
§Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA 48109; email: siqian@umich.edu
1 Introduction

We consider a generic chance-constrained program:

\[
\begin{align*}
\text{min} & \quad c^T x \\
\text{s.t.} & \quad \mathbb{P}\{x \in F(\xi)\} \geq 1 - \epsilon \\
& \quad x \in \mathcal{X} \subseteq \mathbb{R}^d,
\end{align*}
\]

(1a) \hspace{1cm} (1b) \hspace{1cm} (1c)

where \(x\) represents a \(d\)-dimensional decision vector, \(\xi\) is a multivariate random vector, and \(c \in \mathbb{R}^d\) is a given cost parameter. We denote \(F(\xi) \subseteq \mathbb{R}^d\) as some region parameterized by \(\xi\), and the decision vector \(x\) must fall into it with at least \(1 - \epsilon\) probability \((0 < \epsilon < 1)\). The set \(\mathcal{X}\) represents a deterministic feasible region, which can be either continuous or discrete.

The above chance-constrained programs are often used for guaranteeing quality of service and bounding the risk of the occurrence of undesirable outcomes in many application areas. However, they are generally nonconvex unless under specially distributed random parameter \(\xi\) (Prékopa, 1970). Luedtke and Ahmed (2008) employ the Sample Average Approximation (SAA) method to reformulate generic chance-constrained programs as approximate mixed-integer linear programs (MILPs) based on finite samples of the random vector \(\xi\). They analyze the objective bounds and solution feasibility dependent on sample size, confidence level, and approximate risk level used in the SAA approach. Pagnoncelli et al. (2009); Luedtke et al. (2010); Küçükyavuz (2012); Luedtke (2014) study effective valid inequalities, and/or decomposition-based branch-and-cut algorithms for optimizing the SAA-based MILP formulations of chance-constrained programs. It is worth noting that the SAA approach can easily adapt to any types of random vector \(\xi\) that follows either a discrete or a continuous distribution.

Assuming finite support of the random vector \(\xi\), Watson et al. (2010) generalize the scenario decomposition method (Rockafellar and Wets, 1991) to solve chance-constrained programs. In general, scenario decomposition algorithms reformulate a stochastic program by making scenario-based copies of the here-and-now decision (i.e., variable \(x\) in the chance-constrained model (1)), and adding “nonanticipativity constraints” to require these copies to take the same value. Then, a Lagrangian relaxation can be attained by relaxing the nonanticipativity constraints and can be fully decomposed into scenario-independent subproblems. In addition to relaxing nonanticipativity constraints, Watson et al. (2010) also relax the knapsack constraint that couples indicator binary variables defined for each scenario in the MILP reformulation. The basic scenario decomposition approach has been generalized to solve various types of two/multi-stage stochastic linear/integer programs in the past decades, including representative work by Ahmed (2013); Carøe and Schultz (1999); Dentcheva and Römisch (2004); Miller and Ruszczyński (2011); Collado et al. (2012); Deng et al. (2016) with different variable settings and risk preference setups.

Ahmed et al. (2016) provide a comprehensive review of the literature on modeling and solving chance-constrained programs with finite support, mainly by applying integer programming techniques. The authors propose new Lagrangian dual formulations to derive lower bounds for the optimal objective value of the chance-constrained program (1), which can be incorporated in
a branch-and-cut scheme. They also develop a scenario decomposition algorithm and discuss potential “scenario-grouping” strategies to derive more efficient algorithmic variants.

In this paper, we base our approaches on quantile bounds for chance-constrained programs (see Song et al., 2014; Ahmed et al., 2016), and continue investigating effective ways of grouping scenarios to achieve better computational performance of decomposition algorithms, especially scenario decomposition. Unlike “random scenario grouping” in the current literature (e.g., Ahmed et al., 2016; Gade et al., 2016), we formulate a mathematical optimization model to guide the 0-1 decisions of grouping scenarios to result in the tightest quantile bounds. To our best knowledge, the optimization driven scenario grouping method has only been discussed by Ryan et al. (2016) but for solving expectation-based stochastic programs. However, the optimal scenario grouping model in our paper is not a trivial extension from the MILP approximate model solved in Ryan et al. (2016), but becomes a mixed-integer bilevel model in which we derive an inner optimization of group-based subproblems for computing quantile bounds. We further investigate exact solution methods and special cases for solving the bilevel scenario grouping model. We also develop heuristic methods for grouping similar or dissimilar scenarios. In the computational studies, we compare optimality gaps and bounds given by different scenario grouping strategies, via solving several test sets of chance-constrained programming instances. We find that optimal grouping bounds are much tighter, yield smaller root node gaps and significant improvement of the scenario decomposition algorithm for optimizing instances of chance-constrained 0-1 programs as compared to the non-grouping quantile bounds and heuristic grouping bounds. The optimal grouping bounds can be much more strengthened if we increase the group sizes and can become very tight even we use moderate-sized scenario groups. We also compute the quantile bounds using the best feasible solutions given by solving the optimal grouping model for only 20%-50% of the total solution time and find out that they are sufficiently tight with gaps being under 10% of the corresponding optimal grouping bounds.

The rest of this paper is organized as follows. In Section 2, we describe quantile bounds for chance-constrained programs and review the most relevant literature on scenario grouping. In Section 3, we formulate an optimization model for grouping scenarios to achieve the tightest quantile bound given fixed group size, and discuss exact solution approaches for general and special cases. In Section 4, we develop several heuristics for grouping scenarios based on different metrics. In Section 5, we use scenario grouping in scenario decomposition for solving 0-1 chance-constrained programs. We then conduct computational studies on diverse sets of chance-constrained problem instances by applying different grouping methods. In general, optimal grouping bounds perform much better than original quantile bounds and heuristic grouping bounds, but could take longer time to compute. Finally, in Section 6 we conclude the paper and outline the future research.
2 Preliminaries and Literature on Scenario Grouping

Recall the chance-constrained program (1). In this paper, we assume that (i) the random vector \( \xi \) has a finite support \( \Xi = \{ \xi_1, \ldots, \xi_K \} \), where each \( i \in \{1, \ldots, K \} \) refers to a scenario; (ii) each scenario is realized with equal probability, \( 1/K \); (iii) the feasible region is nonempty. Note that Assumption (ii) is made without loss of generality, since if the scenarios have different probabilities to be realized, we can make sufficient number of copies of each scenario and increase \( K \) to have equal probabilities of \( \xi_k \), \( k = 1, \ldots, K \). The objective \( c^\top x \) is also set as a linear function without loss of generality, and our approaches can be used for handling chance-constrained programs with general objective function \( f(x) : \mathbb{R}^d \rightarrow \mathbb{R} \), for which we minimize a linear function \( t \) and let \( t = f(x) \) be part of the constraints defining set \( \mathcal{X} \).

2.1 Quantile bounds and scenario-grouping model

Letting \( I(\cdot) \) be the indicator function, we rewrite (1) in an equivalent form:

\[
\text{CCP}: \quad v^* := \min c^\top x \\
\text{s.t.} \quad \sum_{k=1}^K I(x \not\in F_k) \leq K' \\
x \in \mathcal{X},
\]

where \( F_k := \mathcal{F}(\xi_k) \) for \( k = 1, \ldots, K \) and \( K' := \lfloor \epsilon K \rfloor \). Using binary variables to model the outcomes of the indicator function in all the scenarios, we can further reformulate CCP as an MILP with a knapsack constraint (see Ahmed et al., 2016). As previously reviewed in Section 1, decomposition algorithms are commonly proposed to accelerate the computation of CCP by iteratively generating bounds or valid cuts into a relaxed master problem, from solving individual-scenario based problems, referred to as scenario subproblems that can be solved in a distributed framework.

One of the well-studied relaxation bounds for CCP is the quantile bound (see Song et al., 2014; Ahmed et al., 2016), proposed based on the idea that in order to meet the probabilistic requirement, the decision vector \( x \) must fall in sufficiently many subregions \( F_k \)'s. The optimal objective values of the \( K \) scenario subproblems are

\[
\psi_k := \min \left\{ c^\top x : x \in F_k, \ x \in \mathcal{X} \right\}, \ \forall k = 1, \ldots, K. \tag{2}
\]

These values are ordered to obtain a permutation \( \sigma \) of the set \( \{1, \ldots, K\} \) such that \( \psi_{\sigma_1} \geq \cdots \geq \psi_{\sigma_K} \). Given \( K' = \lfloor \epsilon K \rfloor \), it is clear that the \( (K'+1) \)th quantile value, \( \psi_{\sigma_{K'+1}} \), is a valid lower bound for CCP, because the decision vector \( x \) will fall in at least one \( F_k \) with \( k \in \{\sigma_1, \ldots, \sigma_{K'+1}\} \). The problem of finding such a quantile bound is given by

\[
v^Q := \max \left\{ \rho : \sum_{k=1}^K \frac{1}{K} I(\rho \leq \psi_k) > \epsilon \right\} = \max \left\{ \rho : \sum_{k=1}^K I(\rho \leq \psi_k) \geq K' + 1 \right\}. \tag{3}
\]
Another way to derive a valid lower bound for CCP is to use a relaxed formulation obtained by using scenario grouping as follows (Ahmed et al., 2016). We partition the set of scenarios, \( \{1, \ldots, K\} \), into \( N \) disjoint subsets \( G_1, \ldots, G_N \), and formulate the following scenario grouping model:

\[
\text{SGM} : \quad v_{\text{SGM}} := \min \ c^\top x \\
\text{s.t.} \quad N \sum_{n=1}^N \mathbb{I} \left( x \notin \bigcap_{k \in G_n} \mathcal{F}_k \right) \leq K' \\
\phantom{\text{s.t.}} \quad x \in \mathcal{X}.
\]

SGM is a relaxation of CCP, since the violated groups are at least as many as the violated scenarios. Moreover, we choose the group size \( N \) from \( \{K' + 1, \ldots, K\} \) for SGM to be an effective relaxation. (If \( N \leq K' \), SGM is simply \( \min \{c^\top x : x \in \mathcal{X}\} \) which does not account for the chance constraint at all.) Also note that SGM is not a relaxation, if \( \{G_1, \ldots, G_N\} \) does not form a partition of scenarios \( \{1, \ldots, K\} \), i.e., if some scenarios are shared among different groups.

Following (3) (which is a special case when \( N = K \)), we define the quantile bound of SGM as:

\[
v_{\text{QSGM}} := \max \left\{ \rho : \sum_{n=1}^N \mathbb{I} (\rho \leq \phi_n) \geq K' + 1 \right\},
\]

where we solve

\[
\phi_n := \min \left\{ c^\top x : x \in \bigcap_{k \in G_n} \mathcal{F}_k, x \in \mathcal{X} \right\}, \quad n = 1, \ldots, N.
\]

We call Model (5) \textit{group subproblems} in this paper. The grouping-based quantile bound \( v_{\text{QSGM}} \) is a valid lower bound for CCP (i.e., \( v_{\text{QSGM}} \leq v_{\text{SGM}} \leq v^* \)).

### 2.2 Literature on scenario grouping

In the above scenario grouping idea, how to group scenarios is a key question and can sometimes significantly affect the computational performance of an underlying algorithm. There have been many theoretical and computational schemes proposed for reducing the number of scenarios in stochastic programming. We review the ones that determine grouping strategies based on solution information obtained from scenario subproblems. Escudero et al. (2013) apply scenario grouping for solving stochastic mixed 0-1 programs using a dual decomposition algorithm. They focus on demonstrating the effect of the number of groups on bound tightness and computational efficacy. Crainic et al. (2014) use scenario grouping in progressive hedging algorithms for solving generic stochastic programs. They develop more refined heuristics that extract descriptive statistics of each scenario and then group scenarios based on their relative differences. In terms of bound tightness and computational efficacy, Crainic et al. (2014) show the advantages of using input parameters to characterize a scenario (as compared to using an optimal solution), and the
advantage of using groups that form a cover of scenarios (as compared to those that form a partition). They also show that grouping similar scenarios or dissimilar scenarios could lead to good results dependent on specific problem settings.

Ryan et al. (2016) propose an optimization driven grouping strategy for optimizing stochastic 0-1 programs, implemented by solving an MILP in each iteration of a scenario decomposition algorithm. The algorithm is implemented in a parallel computing framework, and the computational results demonstrate the power of optimal scenario grouping as a preprocessing step. Gade et al. (2016) also implement a subproblem bundling approach, similar to scenario grouping, in the progressive hedging algorithm for optimizing stochastic mixed-integer programs. They arbitrarily decide the size of subproblems to bundle and show that it still improves the convergence of the algorithm and their proposed lower bounds. In contrast to stochastic programming, scenario grouping in chance-constrained optimization has received limited attention. Ahmed et al. (2016) investigate continuous relaxations and dual decomposition algorithms for chance-constrained programs. They propose the scenario grouping idea and demonstrate the computational efficacy of the approach used in scenario decomposition for chance-constrained 0-1 programs. However, they arbitrarily pick scenarios to group without further investigating grouping strategies that could lead to better bounds.

In this paper, we aim to utilize scenario grouping strategies to enhance algorithms for chance-constrained programming models. Our main contribution is to develop an optimization model for grouping scenarios and strengthening quantile bounds for chance-constrained programs. The generic model is bilevel and is more complex than the single-level mixed-integer linear program solved in Ryan et al. (2016) for optimization driven scenario grouping when solving expectation-based stochastic programs. To implement the scenario grouping more efficiently, we develop both exact algorithms using branch-and-cut and heuristic approaches. They explore the features of chance-constrained programs with carefully designed metrics for evaluating the effectiveness of grouping. The goal is to balance the efforts between grouping scenarios and solving group subproblems, and to trade off between such efforts and the quality of the resulting bounds. Through extensive computational studies, we investigate (i) whether it is worthwhile grouping scenarios optimally, (ii) how to determine group sizes, and (iii) whether to group similar or dissimilar scenarios.

3 Optimization-Based Scenario Grouping

Recall that $K$ is the number of scenarios, $N$ is the number of groups as disjoint subsets of scenarios of $\{1, \ldots, K\}$, and we let $K' = \lfloor \epsilon K \rfloor$ where $\epsilon$ is the risk tolerance level in Model (1). We define decision variables $y_{kn} \in \{0, 1\}$, $k = 1, \ldots, K$, $n = 1, \ldots, N$ to indicate whether scenario $k$ is assigned to group $G_n$, such that $y_{kn} = 1$ if yes, and $= 0$ otherwise. We formulate an optimization model to group scenarios and solve the ordered objective values $\phi_1, \ldots, \phi_N$ of group subproblems (5) in which we maximize $\phi_{K' + 1}$ for $K' = \lfloor \epsilon K \rfloor$. Then following (4) we can obtain the tightest group-based quantile bound $v_{SGM}^Q$ that is equal to $\phi_{K' + 1}$. The optimization model is
\[
\text{QGP} : \quad \max \phi_{K' + 1} \tag{6a}
\]

s.t. \[
\phi_n \leq \min \left\{ c^\top x : \mathbb{I}(x \in F_k) \geq y_{kn}, \forall k = 1, \ldots, K, x \in \mathcal{X} \right\} \quad \forall n = 1, \ldots, N \tag{6b}
\]

\[
\phi_n - \phi_{n+1} \geq 0 \quad \forall n = 1, \ldots, N - 1 \tag{6c}
\]

\[
\sum_{n=1}^{N} y_{kn} = 1 \quad \forall k = 1, \ldots, K \tag{6d}
\]

\[
\sum_{k=1}^{K} y_{kn} \leq P \quad \forall n = 1, \ldots, N \tag{6e}
\]

\[
y_{kn} \in \{0, 1\} \quad \forall n = 1, \ldots, N, k = 1, \ldots, K. \tag{6f}
\]

In (6b), we use \(\phi_1, \ldots, \phi_N\) to keep the optimal objective values of group subproblems following the definitions of \(\phi_n, n = 1, \ldots, N\) in (5). That is, if scenario \(k\) is in group \(n\) \((y_{kn} = 1)\), we enforce \(\mathbb{I}(x \in F_k) = 1\) and thus \(x \in F_k\). Otherwise, the constraints of \(x \in F_k\) are relaxed in the corresponding group subproblems to allow larger \(\phi_n\)-values and thus larger quantile bound \(\phi_{K' + 1}\) maximized in the objective function (6a). To exclude symmetric grouping solutions, we order the optimal objective values of group subproblems following a nonincreasing order given in (6c). We assign every scenario a group in (6d). Moreover, we restrict each group size by an integer parameter \(P\) in (6e) with \(P \geq K/N\). Without this restriction, i.e., if \(P = K\), the model will allocate scenarios densely into \(K' + 1\) groups and each group subproblem could be difficult to solve (see Ryan et al., 2016).

In the rest of the section, we develop exact solution approaches for optimizing QGP in Model (6) to maximize the quantile bound. In Section 3.1, we consider chance-constrained programs with linear decision variables and constraints, for which we reformulate QGP as an MILP that can be directly optimized by off-the-shelf solvers. In Section 3.2, we propose a branch-and-cut algorithm for optimizing QGP for general chance-constrained programs given in (1).

### 3.1 MILP reformulation of QGP for chance-constrained linear programs

We first examine special chance-constrained programs whose corresponding scenario grouping models can be directly solved by off-the-shelf solvers. In particular, we consider chance-constrained linear programs in the form of (1), in which set \(\mathcal{X} = \mathbb{R}^d_+\) and

\[
F_k = \left\{ x : A_k x \geq r_k \right\},
\]

where \(A_k \in \mathbb{R}^{m_k \times d}\) and \(r_k \in \mathbb{R}^{m_k}\) for \(k = 1, \ldots, K\). Then, for each \(n = 1, \ldots, N\), we can reformulate (6b) as follows:

\[
\phi_n \leq \min \left\{ c^\top x : A_k x \geq r_k - M_k (1 - y_{kn}), \forall k = 1, \ldots, K, x \in \mathbb{R}^d_+ \right\}, \tag{7}
\]

where \(M_k\) is an \(m_k\)-dimensional big-M coefficient vector ensuring that when \(y_{kn} = 0\), we can relax the constraint \(A_k x \geq r_k\) in the \(n\)th group subproblem. The computation of valid \(M_k\) can follow
the procedures of strengthening big-M coefficients for general joint chance-constrained programs discussed in detail in, e.g., Qiu et al. (2014) and Song et al. (2014).

The right-hand side of (7) for each \( n = 1, \ldots, N \) is a linear program in variable \( x \). Let \( \lambda_{kn} \in \mathbb{R}^{m_k}_+ \) be the dual vector associated with the \( k \)th set of constraints in the minimization problem in (7). The dual problem is formulated as

\[
D_n(y) := \max \sum_{k=1}^{K} \left( r_k^T \lambda_{kn} - M_k^T \lambda_{kn} (1 - y_{kn}) \right) \tag{8a}
\]

s.t. \( \sum_{k=1}^{K} A_k^T \lambda_{kn} \leq c \tag{8b} \)

\( \lambda_{kn} \in \mathbb{R}^{m_k}_+ \). \tag{8c}

We further define auxiliary decision vectors \( w_{kn} \in \mathbb{R}^{m_k}_+ \) such that

\[
w_{kn} \equiv \lambda_{kn} y_{kn}, \forall k = 1, \ldots, K, n = 1, \ldots, N. \tag{9a}\]

Using the McCormick inequalities (McCormick, 1976), we can linearize the above dual and derive an equivalent MILP to replace the right-hand side of (7) for each \( n = 1, \ldots, N \):

\[
\max_{y, \lambda, w} \sum_{k=1}^{K} \left( r_k^T \lambda_{kn} - M_k^T \lambda_{kn} + M_k^T w_{kn} \right) \tag{9a}
\]

s.t. \( w_{kn} \leq \lambda_{kn}, w_{kn} \leq \overline{\lambda}_{kn} y_{kn}, \forall k = 1, \ldots, K \tag{9b} \)

\( w_{kn} \geq \lambda_{kn} - \overline{\lambda}_{kn} (1 - y_{kn}), \forall k = 1, \ldots, K \tag{9c} \)

\( \lambda_{kn}, w_{kn} \in \mathbb{R}^{m_k}_+, y_{kn} \in \{0, 1\}, \forall k = 1, \ldots, K. \tag{9d} \)

Here \( \overline{\lambda}_{kn} \in \mathbb{R}^{m_k}_+ \) represents a valid upper bound of vector \( \lambda_{kn} \). For example, when all the entries in matrix \( A_k^T \) are positive, we can set

\[
\overline{\lambda}_{knj} = \min_{i=1, \ldots, d} \left\{ \frac{c_i}{(A_k^T)_{ij}} \right\}
\]

as the \( j \)th element of the vector \( \overline{\lambda}_{kn} \), \( j = 1, \ldots, m_k \), where \( (A_k^T)_{ij} \) denotes the entry in row \( i \), column \( j \) of matrix \( A_k^T \), for each \( i = 1, \ldots, d \) and \( j = 1, \ldots, m_k \). Also if all the entries in vector \( r_k \) are nonnegative, due to the maximization nature in the objective function (9a), we can relax the lower bound constraints (9c) since they will be automatically satisfied at optimum.

Overall, for chance-constrained linear programs, we can directly solve QGP as the following MILP with continuous variables \( \phi, \lambda, w, \) and binary variables \( y \):

\[
\max_{\phi, \lambda, w, y} \left\{ \phi_{K + 1} : \phi_n \leq \sum_{k=1}^{K} \left( r_k^T \lambda_{kn} - M_k^T \lambda_{kn} + M_k^T w_{kn} \right), n = 1, \ldots, N, \ (6c)-(6e), (8b), (9b), (9d) \right\}. \tag{10}\]

We note that the MILP model solved by Ryan et al. (2016) for grouping scenarios only accounts for a subset of grouping choices and thus is an approximation. Here in model (10) we optimize over all the solutions and present an exact formulation for optimal scenario grouping.
3.2 A branch-and-cut approach for optimizing QGP

QGP is a bilevel program different from the single-level model solved in Ryan et al. (2016). We optimize it by applying a cutting-plane algorithm with separation procedures given as follows. We consider a master problem of QGP as:

$$\max \{ \phi_{K+1} : (6c) - (6e), (y, \phi) \in A, \phi_n \in \mathbb{R}, y_{kn} \in \{0, 1\}, \forall k = 1, \ldots, K, n = 1, \ldots, N \},$$  \hspace{1cm} (11)

where we rewrite constraint (6b) as \((y, \phi) \in A\) and enforce it by iteratively adding linear inequalities of \((y, \phi)\) – they are cutting planes we obtain by solving group subproblems. We integrate the cutting-plane procedures into a branch-and-cut framework.

Specifically, for any \((\hat{y}, \hat{\phi})\) (where \(\hat{y}\) could be fractional) obtained from solving a relaxation of (11) (which is referred to as relaxed master problem containing a subset of inequalities defining the set \(A\)), we consider and define a group set \(G_n^* := \{ k \in \{1, \ldots, K \} : \hat{y}_{kn} > 0 \} \) for each \(n = 1, \ldots, N \). Then we compute the value of

$$\phi^*_n = \min \left\{ c^\top x : x \in \bigcap_{k \in G_n^*} F_k, x \in \mathcal{X} \right\}. $$  \hspace{1cm} (12)

If \(\phi^*_n < \hat{\phi}_n\), following Laporte and Louveaux (1993) we add a cut

$$\phi_n \leq (U - \phi_n^*) \left( \sum_{k: \hat{y}_{kn} = 0} y_{kn} - 1 \right) + U $$  \hspace{1cm} (13)

to the relaxed master problem, where \(U\) is a finite value that needs to satisfy

$$U \geq \max\{ \phi_n : (y, \phi) \text{ in the feasible region of (11)} \}. $$

Assuming the set \(\mathcal{X}\) being bounded, we set \(U = \max\{c^\top x : x \in \mathcal{X}\}\) in our later computation.

**Proposition 1** (13) is valid for QGP.

**Proof:** Consider any feasible solution \((y, \phi)\) of QGP. For any \(n \in \{1, \ldots, N\}\), if \(\sum_{k: \hat{y}_{kn} = 0} y_{kn} \geq 1\), the inequality (13) is trivially satisfied. Otherwise, we have \(\{k : \hat{y}_{kn} = 0\} \subseteq \{k : y_{kn} = 0\}\) due to that \(\sum_{k: \hat{y}_{kn} = 0} y_{kn} = 0\). It then follows that \(\{k : y_{kn} = 1\} \subseteq \{k : \hat{y}_{kn} = 1\}\). Therefore,

$$\phi_n \leq \min \left\{ c^\top x : x \in \bigcap_{k : \hat{y}_{kn} = 1} F_k, x \in \mathcal{X} \right\} \leq \min \left\{ c^\top x : x \in \bigcap_{k : \hat{y}_{kn} = 1} F_k, x \in \mathcal{X} \right\} = \phi^*_n.$$  \hspace{1cm}

This verifies that the inequality (13) is satisfied and completes the proof. \(\square\)
We incorporate the valid inequality (13) in a branch-and-cut framework for optimizing QGP, detailed in Algorithm 1. At each branching node \( \nu \), we compute the following relaxation problem:

\[
\text{NRP}(\nu) : \max \phi_{K'+1} \quad (14a)
\]
\[
\text{s.t. } (6d), (6e), (6c)
\]
\[
(y, \phi) \in \tilde{A} \subset A, \quad (14b)
\]
\[
0 \leq y_{kn} \leq 1, \forall k = 1, \ldots, K, \ n = 1, \ldots, N \quad (14c)
\]
\[
y_{kn} = 0, \forall (k, n) \in F_0(\nu), \quad y_{kn} = 1, \forall (k, n) \in F_1(\nu), \quad (14d)
\]

where set \( \tilde{A} \) in (14b) collects all the cuts (13) that were added up to the branching node \( \nu \) up to the current iteration. Constraints (14d) present the variables being fixed at 0 and 1, indicated by indices \( (k, n) \) in set \( F_0(\nu) \) and set \( F_1(\nu) \), respectively. (The sets \( F_0(\nu) \) and \( F_1(\nu) \) are disjoint subsets of \( \{1, \ldots, K\} \times \{1, \ldots, N\} \).) Throughout Algorithm 1, \( \overline{\text{obj}} \) keeps an incumbent upper bound.

**Algorithm 1** A branch-and-cut algorithm for optimizing QGP and bounding CCP

1: Initialize \( \overline{\text{obj}} \leftarrow \infty \).
2: Initialize \( \text{NodeList} \leftarrow \{\nu_0\} \), where \( F_0(\nu_0) = \emptyset \), \( F_1(\nu_0) = \emptyset \).
3: Choose a branching node \( \nu \in \text{NodeList} \).
4: Solve \( \text{NRP}(\nu) \) and obtain the solution \((\hat{y}, \hat{\phi})\).
5: Compute \( \phi^*_1, \ldots, \phi^*_N \), and sort them in a nonincreasing order.
6: if \( \phi^*_{K'+1} < \overline{\text{obj}} \) then
7: \quad if for any \( n, \phi^*_n < \hat{\phi}_n \) then
8: \quad \quad \quad Add a cut (13) into \((y, \phi) \in \tilde{A}\).
9: \quad \quad \quad Go to Step 4.
10: \quad end if
11: \quad if \( \hat{y} \) is integral then
12: \quad \quad \overline{\text{obj}} \leftarrow \phi^*_{K'+1}.
13: \quad else
14: \quad \quad Branch on a binary variable having a fractional solution value and add two new branching nodes to \( \text{NodeList} \).
15: \quad end if
16: end if
17: \text{NodeList} \leftarrow \text{NodeList}\backslash\{\nu\}.
18: if \( \text{NodeList} \) is empty then
19: \quad Report \( \overline{\text{obj}} \) as the optimal objective value.
20: else
21: \quad Go to Step 3.
22: end if
4 Heuristic-Based Scenario Grouping

In this section, we investigate heuristic approaches to group scenarios instead of solving QGP. In Section 4.1, we develop a method to guarantee that the quantile bound of the group subproblems is at least as tight as the quantile bound of the original CCP. In Section 4.2, we group similar scenarios by using a greedy algorithm and a K-means clustering algorithm from machine learning. In Section 4.3, we develop a greedy heuristic to group dissimilar scenarios. In this paper, similarity is measured by the $L^1$-norm distance between vectors characterizing individual scenarios, which we describe in detail later.

4.1 Anchored grouping

We first solve scenario subproblems in (2) to obtain $\psi_1, \ldots, \psi_K$, and sort their objective values such that $\psi_1 \geq \cdots \geq \psi_K$. Clearly, due to the preliminaries in Section 2.1, $\psi_{K'+1}$ is a valid quantile bound for CCP. We construct $N$ ($N \geq K/P$ and $N \geq K'+1$) non-empty groups such that scenario $k_n$ is in $G_n$ for $n = 1, \ldots, N$. Then we distribute the remaining scenarios into different groups and meanwhile make sure that all the group sizes do not exceed $P$. Following such grouping procedures, the resulting SGM has a quantile bound that is at least $\psi_{K'+1}$, which we show in Proposition 2.

**Proposition 2** Given that $k_n \in G_n$ for $n = 1, \ldots, N$ where $(k_1, \ldots, k_K)$ is a permutation of $1, \ldots, K$ such that $\psi_1 \geq \cdots \geq \psi_K$, the quantile bound of SGM is at least as large as the quantile bound of CCP.

**Proof:** For all $n = 1, \ldots, N$,

$$\phi_n = \min \left\{ c^\top x : x \in \bigcap_{k \in G_n} F_k, x \in \mathcal{X} \right\} \geq \min \left\{ c^\top x : x \in F_{k_n} \cap \mathcal{X} \right\} = \psi_{k_n}.$$

Then it follows that the quantile bound of SGM, which is the $(K'+1)$th largest of $\{\phi_1, \ldots, \phi_N\}$, is at least as large as the quantile bound of CCP, which is the $(K'+1)$th largest of $\{\psi_1, \ldots, \psi_K\}$. □

Next, we group scenarios based on their similarity or dissimilarity. We let $v^1, \ldots, v^K$ be the vectors characterizing features of scenarios $1, \ldots, K$, and define the distance between any two scenarios $k$ and $k'$ using the $L^1$-norm of the difference between their corresponding characterizing vectors, i.e.,

$$d(k, k') := \|v^k - v^{k'}\|.$$

We compare three grouping approaches in Sections 4.2.1, 4.2.2, 4.3, of which the first two aim to group more similar scenarios, while the last one groups dissimilar scenarios.
4.2 Grouping similar scenarios

4.2.1 A greedy approach

We assume that the computational effort for solving each scenario-based subproblem is similar. We therefore attempt to form evenly-sized groups, with the size of each $G_n$ given by

$$s_n := \begin{cases} 
\lfloor K/N \rfloor + 1 & \text{for } n = 1, \ldots, K_{\text{mod}}N \\
\lfloor K/N \rfloor & \text{for } n = K_{\text{mod}}N + 1, \ldots, N 
\end{cases} \quad (15)$$

This can intuitively be explained as spreading $K$ scenarios across $N$ groups in a round robin manner, as a result of which the sizes of the maximal and the minimal groups differ at most by 1.

To populate each group, we start with randomly picking an ungrouped scenario to start a group, and then repeatedly include a scenario closest to the center of the incumbent group until we reach the size limit of that group. Here the center of any group $G \subseteq \{1, \ldots, K\}$, denoted $\bar{\tau}$, is defined as the arithmetic mean of the characterizing vectors of the contained scenarios, i.e.,

$$\bar{\tau} = \frac{1}{|G|} \sum_{k \in G} v^k.$$ 

Let $S$ denote the set of all the ungrouped scenarios. Then, the nearest ungrouped scenario to some “center” vector $\bar{\tau}$ is given by

$$\arg \min_{k \in S} \|v^k - \bar{\tau}\|.$$ 

Algorithm 2 outlines the procedures of the greedy algorithm for grouping similar scenarios.

**Algorithm 2** A greedy approach for grouping similar scenarios.

1: Initialize $S \leftarrow \{1, \ldots, K\}$.
2: for $n = 1, \ldots, N$ do
3: \hspace{1em} Update $\hat{k} \leftarrow$ a randomly chosen scenario from $S$.
4: \hspace{1em} Update $G_n \leftarrow \{\hat{k}\}$, and $S \leftarrow S \setminus \{\hat{k}\}$. Compute $s_n$ according to (15).
5: \hspace{1em} for $s_n - 1$ times do
6: \hspace{2em} Compute $\tau \leftarrow$ the center of $G_n$.
7: \hspace{2em} Choose $k^* \leftarrow$ the nearest ungrouped scenario to $\tau$.
8: \hspace{2em} Update $G_n \leftarrow G_n \cup \{k^*\}$, and $S \leftarrow S \setminus \{k^*\}$.
9: \hspace{1em} end for
10: end for

4.2.2 A $K$-means clustering approach for grouping similar scenarios

$K$-means clustering, first used by MacQueen et al. (1967), is a very popular method of vector quantization for cluster analysis in machine learning. The method clusters data points to minimize the sum of squared distances from each point being clustered to its cluster center. However, finding an exact solution to this problem is NP-hard (see, e.g., Aloise et al., 2009), and we apply a
standard approach called Lloyd’s algorithm (Lloyd, 1982) to find an approximate solution to the exact $K$-means clustering. Roughly speaking, the algorithm alternates between the following two steps: (i) assigning each point to a cluster whose center is the closest, and (ii) recomputing the center of each cluster.

Algorithm 3 A $K$-means clustering approach for grouping similar scenarios.

1: Initialize $S \leftarrow \{1, \ldots, K\}$.
2: Initialize $\hat{k} \leftarrow$ a random scenario chosen randomly from $S$.
3: Initialize $\pi_1 \leftarrow v^{\hat{k}}$, and $S \leftarrow S \setminus \{\hat{k}\}$.
4: for $n = 2, \ldots, N$ do
5:   for $k \in S$ do
6:     Update $d(k) \leftarrow \min_{n' \in \{1, \ldots, n-1\}} \|v^k - \pi_{n'}\|
7:   end for
8:   Update $k^* \leftarrow$ a randomly chosen scenario from $S$ with probability proportional to $d(k)^2$.
9:   Update $\pi_n \leftarrow v^{k^*}$, and $S \leftarrow S \setminus \{k^*\}$.
10: end for
11: repeat
12:   for $k = 1, \ldots, K$ do
13:     Update $n^* \leftarrow \arg\min_{n \in \{1, \ldots, N\}} \|v^k - \pi_n\|$
14:     Update $G_{n^*} \leftarrow G_{n^*} \cup \{k\}$.
15:   end for
16:   for $n = 1, \ldots, N$ do
17:     Update $\pi_n \leftarrow$ the center of $G_n$.
18:   end for
19: until no change from the previous iteration

We detail our $K$-means clustering algorithm for grouping similar scenarios in Algorithm 3, in which Steps 1–10 are referred to as the $K$-means++ algorithm, and Steps 11–19 are Lloyd’s algorithm. Note that the $K$-means++ approach appoints scenarios in CCP (viewed as "data points") as initial cluster centers. The first center is randomly chosen from all the scenarios to be clustered. The subsequent center is picked from the remaining scenarios, with a certain probability proportional to its distance to the nearest center that has already been chosen.

4.3 Grouping dissimilar scenarios

We develop a simple extension of the greedy algorithm for grouping similar scenarios to group dissimilar scenarios. The motivation is that, if dissimilar scenarios are grouped together, each group subproblem may become harder to solve, but can potentially produce tighter quantile bounds since the solution of each subproblem needs to satisfy constraints across dissimilar scenarios.

We present the detailed procedures as follows. We first group similar scenarios to form $\Omega$
groups as $G'_1, \ldots, G'_{\Omega}$. Then we collect one scenario from each group $G'_\omega$ ($\omega \in \{1, \ldots, \Omega\}$) to form a new group $G'_n$, which then consists of $\Omega$ “dissimilar scenarios”, each from a different group obtained from the previous similar scenario grouping. We repeat the above process until all the scenarios are grouped.

To make this method comparable to the previous heuristics, we fix the total number of groups at $N$. As a result, it becomes important to size the first set of groups so that the second step can produce $N$ reasonably-sized groups. Here for simplicity, we use the greedy approach for grouping similar scenarios because it allows us to specify the size of each group. In particular, we let $\Omega := \lceil K/N \rceil$, and the size of each group $G'_\omega$, $\omega = 1, \ldots, \Omega$ as

$$s_1, \ldots, s_\Omega := \begin{cases} N & \text{if } K \mod N = 0 \\ N, s_\Omega := K \mod N & \text{otherwise.} \end{cases}$$

The detailed algorithmic steps are presented in Algorithm 4.

**Algorithm 4** A greedy approach for grouping dissimilar scenarios.

1: Evaluate $s_1, \ldots, s_\Omega$ according to (16).
2: Run Algorithm 2 with $s_1, \ldots, s_\Omega$ to form groups $G'_1, \ldots, G'_{\Omega}$.
3: for $n = 1, \ldots, N$ do
4:   for $\omega = 1, \ldots, \Omega$ do
5:     if $G'_\omega \neq \emptyset$ then
6:       Update $\hat{k} \leftarrow$ a scenario chosen randomly from $G'_\omega$.
7:       Update $G'_\omega \leftarrow G'_\omega \setminus \{\hat{k}\}$, and $G_n \leftarrow G_n \cup \{\hat{k}\}$
8:     end if
9:   end for
10: end for

5 Numerical Studies

In this section we evaluate the proposed bounds and algorithms enabled by scenario grouping on two classes of chance-constrained programming problems: (i) chance-constrained portfolio optimization, and (ii) chance-constrained multi-dimensional 0-1 knapsack. Problem (i) contains only linear decision variables and constraints, and thus its optimal grouping model QGP can be solved directly as an MILP specified from Model (10). Problem (ii) contains binary packing variables, and we need to implement the branch-and-cut algorithm in Section 3.2 to optimize the scenario grouping decisions.

For both Problems (i) and (ii), we test standard instances from the literature and compare the quantile bounds obtained by optimal grouping and by other grouping strategies discussed before. We also discuss how to incorporate scenario grouping to derive a finitely convergent scenario decomposition algorithm for solving chance-constrained 0-1 programs, and test the approach on Problem (ii) instances that contain binary variables.
Section 5.1 describes detailed experimental setups including sources of the test instances and computational settings. Section 5.2 compares the bounds of different grouping strategies for solving instances of Problem (i) and Problem (ii). In Section 5.3, we describe a scenario decomposition algorithm for chance-constrained 0-1 programs using group subproblems rather than scenario subproblems. We then demonstrate the computational results using Problem (ii) instances.

5.1 Experimental setup and test instances

**Problem (i):** The first class of instances are based on the chance-constrained portfolio optimization problem described below, studied by Qiu et al. (2014) and posted on the Stochastic Integer Programming Test Problem Library (SIPLIB, Ahmed et al. (2015)). We consider

\[
\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad \mathbb{P}\left\{ (a(\xi))^T x \geq r \right\} \geq 1 - \epsilon \\
& \quad x \in \mathcal{X} = \left\{ x \in \mathbb{R}^d_+ : e^T x = 1 \right\},
\end{align*}
\]

where the decision vector \(x\) represents investments on \(d\) assets with random return rates. We minimize the total cost of investment, with \(c \in \mathbb{R}^d\) being the cost vector, and require that the total return, as the product of the investment decision \(x\) and random return rate vector \(a(\xi)\), is no less than a threshold value \(r\), with at least \(1 - \epsilon\) probability. The constraint in set \(\mathcal{X}\) scales the nonnegative investment levels to a unit budget.

The above model is a chance-constrained linear program studied in Section 3.1, and we denote \(a_k\) as the realization of the random vector \(a(\xi)\) in scenario \(k\). Specifically, \(m_k = 1, \forall k = 1, \ldots, K\) and we have \(A_k = a_k^T, r_k = r\), leading to a single-row linear constraint \(a_k^T x \geq r\) in each \(F_k\). As a result, we can eliminate the big-M coefficients and further simplify Model (7) as

\[
\phi_n \leq \min \left\{ c^T x : a_k^T x \geq ry_k, \forall k = 1, \ldots, K, e^T x = 1, x \in \mathbb{R}^d_+ \right\}.
\]

We then follow similar procedures in Section 3.1 to dualize the right-hand side minimization problem, linearize the bilinear dual model, and optimize scenario grouping by solving an MILP.

We use the same parameter setups in Qiu et al. (2014) as follows. We consider \(d = 20\) assets and the number of scenarios \(K = 200\). We generate each element of the vector \(a_k\) for each scenario \(k\) independently from a uniform distribution between 0.8 and 1.5, representing a range between a 20% loss and a 50% gain of the investment. We set \(r = 1.1\), and generate elements in \(c\) as integer values uniformly distributed between 1 and 100 for each asset. In Qiu et al. (2014), the authors allow violating 15 out of 200 scenarios, which corresponds to \(\epsilon = 0.075\).

**Problem (ii):** The second class of instances are based on the chance-constrained multi-dimensional 0-1 knapsack problem, in which we assign items with random weights to knapsacks to maximize the total value and also guarantee that the joint probability of exceeding multiple knapsacks’ capacities is sufficiently small. We use the two sets of instances \(mk-20-10\) and \(mk-39-5\) tested in
Ahmed et al. (2016), which are originally from Song et al. (2014). The instances with the name \( mk-n-m \) have \( n \) items and \( m \) knapsack constraints. We test scenario size \( K \in \{100, 500, 1000\} \) and run five replications for each scenario size. We follow Song et al. (2014) to randomly generate item weights in every scenario. Since the results are similar among different replications, we report the averages. We vary the risk parameter \( \epsilon = 0.1, 0.2 \) for each type of instances and scenario size.

All computations are performed on a Linux workstation with four 3.4 GHz processors and 16 GB memory. All involved MILP and LP models are implemented in C++ and solved by CPLEX 12.6 via ILOG Concert Technology. We set the number of threads to one, and use the default optimality gap tolerance as 0.01% in CPLEX for optimizing MILPs. The computational time limit for solving each instance is 3600 seconds.

5.2 Results of group-based bounds

We first compare the bounds of CCP models in Problem (i) and Problem (ii) produced by different grouping methods in Sections 3 and 4. For each instance we run the following procedures, in which we use equally sized groups and vary the values of \( P \) between 2 and 20 as the number of scenarios in each group. (Correspondingly, we have \( N = K/P \) groups.)

- We call the CPLEX solver to directly optimize the extended MILP reformulation of CCP based on the \( K \) scenarios, and report the results under Column \( \nu^* \);
- We solve the quantile bound of CCP following (3) and report the results under Column \( \nu^Q \);
- For Problem (i) we directly use CPLEX to optimize the MILP in (10), and for Problem (ii) we implement the branch-and-cut algorithm in Section 3.2, both to optimize QGP and obtain optimization-based grouping solutions. We obtain the quantile bound \( \nu^Q_{SGM} \) as the optimal objective value of QGP, and report related results under Column \( OG \);
- We construct \( N \) groups by distributing scenarios to each group in a round-robin manner, i.e.,

\[
G_n = \{ k : (k - 1) \mod N = (n - 1) \}, \ \forall n = 1, \ldots, N.
\]

We then compute the quantile bound \( \nu^Q_{SGM} \) of the corresponding SGM following (4) and report the related results under Column \( RG \) (round-robin grouping);
- We construct \( N \) groups by applying the anchored grouping method in Section 4.1 such that

\[
G_n = \{ k_i : (i - 1) \mod N = (n - 1) \}, \ \forall n = 1, \ldots, N,
\]

and by following the heuristics in Sections 4.2.1, 4.2.2, 4.3 to group similar or dissimilar scenarios. We then compute \( \nu^Q_{SGM} \) in (4) of the corresponding SGM for each heuristic and report their results under Columns \( AG \) (anchored grouping), \( SG \) (similar scenario grouping), \( KG \) (\( K \)-means clustering grouping), and \( DG \) (dissimilar scenario grouping), respectively.
We first compare the bounds obtained by the different procedures above in Table 1. We solve the ten instances tested in Qiu et al. (2014), with their optimal objective values and generic quantile bounds reported in Columns $v^*$ and $v^Q$. Except Instance No.10, all are solved to optimum within the one-hour time limit. Recall that $K = 200$ for all Problem (i) instances. We test $N = 100, 20, 10$ groups with sizes $P = 2, 10, 20$, respectively, and present their corresponding group bounds in the six columns under $v^Q_{SGM}$.

In Table 1, the generic quantile bounds $v^Q$ are quite loose for most instances and the group bounds $v^Q_{SGM}$ obtained by various heuristic approaches are not significantly improved, especially when the group size is small (i.e., $P = 2$). The bounds $v^Q_{SGM}$ based on optimal grouping (i.e., columns OG) are better than the ones obtained by heuristics and are especially tight when we increase the group size to $P = 20$. The bound improvements resulting from optimal scenario grouping are much more significant as compared to the improvements given by heuristic grouping, as we increase the group size $P$.

In Table 2, we report the average CPU time of solving the optimal objective value $v^*$ for the ten instances and the average CPU time of solving various types of bounds. We observe that CPLEX takes on average 3103.72 seconds to solve the MILP reformulations of the chance-constrained programs of Instances No.1–No.9. (Instance No.10 cannot be optimized within the time limit.) Computing the quantile bound $v^Q$ is very efficient, which only requires solving each scenario subproblem as a linear program and sorting their optimal objective values. When applying heuristics to determine grouped scenarios, it takes slightly longer to find slightly better quantile bounds $v^Q_{SGM}$ by RG, AG, SG, KG, and DG when the group size $P = 2$. It becomes harder to optimize the group subproblems when the group size $P$ increases, and thus the average CPU time increases, almost linearly as $P$ increases and $N$ decreases. To determine optimal grouping decisions, one needs to solve an MILP as discussed in Section 5.1 for Problem (i) instances, and thus computing $v^Q_{SGM}$ by OG takes much longer time than the other grouping methods, as expected. The average CPU time of implementing optimization-based scenario grouping increases more drastically than the time of computing bounds by other grouping heuristics as $P$ increases in Table 2.

We further test the quantile bound $v^Q$ given by feasible (not necessarily optimal) grouping solutions calculated CPLEX when we only solve QGP for certain percentage of the overall optimization time reported in Table 2 under Column OG. In particular, we set the computational time limit as 10%, 20%, 30%, 40%, and 50% of the total time for optimizing the grouping problem and report the values of the quantile bound $v^Q_{SGM}$ using the best feasible grouping solutions given by CPLEX in Table 3 under Columns OG-10%, OG-20%, OG-30%, OG-40%, OG-50%, respectively.

In Table 3, the feasible grouping solutions we obtain by running CPLEX for 20%–50% of the total optimization time are able to produce sufficiently good quantile bounds, especially when the group sizes are large (i.e., cases of $P = 10$ and $P = 20$). When we only run CPLEX for 50% of its total time of optimizing QGP, we can obtain optimal grouping results that produce the same quantile bounds in 7 out of ten instances when $P = 20$, and the rest also have very small gaps that are within 10% of the quantile bounds based on optimal scenario grouping.

To evaluate the strengths of the above quantile bounds, we report in Table 4 how much the
Table 1: Bounds given by optimal and heuristic grouping strategies for Problem (i) instances

<table>
<thead>
<tr>
<th>Inst.</th>
<th>$v^*$</th>
<th>$v^Q$</th>
<th>$v^Q_{OG}$</th>
<th>$v^Q_{RG}$</th>
<th>$v^Q_{AG}$</th>
<th>$v^Q_{SG}$</th>
<th>$v^Q_{KG}$</th>
<th>$v^Q_{DG}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40.7</td>
<td>9.31</td>
<td>15.73</td>
<td>9.74</td>
<td>9.40</td>
<td>9.40</td>
<td>8.89</td>
<td>9.82</td>
</tr>
<tr>
<td>2</td>
<td>15.36</td>
<td>8.42</td>
<td>9.75</td>
<td>8.52</td>
<td>8.47</td>
<td>8.49</td>
<td>8.28</td>
<td>8.59</td>
</tr>
<tr>
<td>3</td>
<td>29.13</td>
<td>20.56</td>
<td>22.80</td>
<td>20.57</td>
<td>20.61</td>
<td>20.57</td>
<td>20.57</td>
<td>20.80</td>
</tr>
<tr>
<td>4</td>
<td>36.91</td>
<td>4.37</td>
<td>9.34</td>
<td>4.56</td>
<td>4.51</td>
<td>4.48</td>
<td>4.47</td>
<td>4.66</td>
</tr>
<tr>
<td>5</td>
<td>23.91</td>
<td>7.86</td>
<td>11.45</td>
<td>8.55</td>
<td>8.16</td>
<td>7.86</td>
<td>7.89</td>
<td>8.20</td>
</tr>
<tr>
<td>6</td>
<td>35.18</td>
<td>9.78</td>
<td>13.32</td>
<td>9.96</td>
<td>9.96</td>
<td>9.85</td>
<td>9.81</td>
<td>10.02</td>
</tr>
<tr>
<td>7</td>
<td>41.59</td>
<td>12.06</td>
<td>17.36</td>
<td>12.20</td>
<td>12.25</td>
<td>12.20</td>
<td>12.24</td>
<td>12.66</td>
</tr>
<tr>
<td>8</td>
<td>22.52</td>
<td>7.34</td>
<td>10.63</td>
<td>7.47</td>
<td>7.68</td>
<td>7.49</td>
<td>7.53</td>
<td>7.66</td>
</tr>
<tr>
<td>9</td>
<td>43.98</td>
<td>13.94</td>
<td>21.67</td>
<td>14.68</td>
<td>14.11</td>
<td>14.06</td>
<td>14.56</td>
<td>15.09</td>
</tr>
<tr>
<td>10</td>
<td>33.50-35.73†</td>
<td>10.48</td>
<td>15.25</td>
<td>10.48</td>
<td>10.70</td>
<td>10.70</td>
<td>10.83</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inst.</th>
<th>$v^*$</th>
<th>$v^Q$</th>
<th>$v^Q_{OG}$</th>
<th>$v^Q_{RG}$</th>
<th>$v^Q_{AG}$</th>
<th>$v^Q_{SG}$</th>
<th>$v^Q_{KG}$</th>
<th>$v^Q_{DG}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40.7</td>
<td>9.31</td>
<td>24.03</td>
<td>9.38</td>
<td>11.81</td>
<td>12.51</td>
<td>12.50</td>
<td>13.15</td>
</tr>
<tr>
<td>2</td>
<td>15.36</td>
<td>8.42</td>
<td>11.55</td>
<td>8.83</td>
<td>9.09</td>
<td>9.38</td>
<td>9.18</td>
<td>9.05</td>
</tr>
<tr>
<td>4</td>
<td>36.91</td>
<td>4.37</td>
<td>16.88</td>
<td>5.86</td>
<td>5.88</td>
<td>5.99</td>
<td>6.71</td>
<td>6.55</td>
</tr>
<tr>
<td>5</td>
<td>23.91</td>
<td>7.86</td>
<td>17.49</td>
<td>8.59</td>
<td>8.47</td>
<td>8.30</td>
<td>9.31</td>
<td>9.66</td>
</tr>
<tr>
<td>6</td>
<td>35.18</td>
<td>9.78</td>
<td>21.86</td>
<td>11.67</td>
<td>12.30</td>
<td>11.44</td>
<td>11.08</td>
<td>11.85</td>
</tr>
<tr>
<td>7</td>
<td>41.59</td>
<td>12.06</td>
<td>24.81</td>
<td>14.83</td>
<td>15.17</td>
<td>14.41</td>
<td>15.26</td>
<td>15.78</td>
</tr>
<tr>
<td>8</td>
<td>22.52</td>
<td>7.34</td>
<td>15.85</td>
<td>8.20</td>
<td>9.80</td>
<td>8.46</td>
<td>8.45</td>
<td>9.22</td>
</tr>
<tr>
<td>9</td>
<td>43.98</td>
<td>13.94</td>
<td>27.79</td>
<td>16.92</td>
<td>17.76</td>
<td>17.47</td>
<td>17.12</td>
<td>17.96</td>
</tr>
<tr>
<td>10</td>
<td>33.50-35.73†</td>
<td>10.48</td>
<td>19.27</td>
<td>11.18</td>
<td>13.05</td>
<td>11.85</td>
<td>12.56</td>
<td>12.76</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inst.</th>
<th>$v^*$</th>
<th>$v^Q$</th>
<th>$v^Q_{OG}$</th>
<th>$v^Q_{RG}$</th>
<th>$v^Q_{AG}$</th>
<th>$v^Q_{SG}$</th>
<th>$v^Q_{KG}$</th>
<th>$v^Q_{DG}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40.7</td>
<td>9.31</td>
<td>36.73</td>
<td>11.04</td>
<td>14.84</td>
<td>16.65</td>
<td>17.57</td>
<td>17.60</td>
</tr>
<tr>
<td>2</td>
<td>15.36</td>
<td>8.42</td>
<td>13.69</td>
<td>9.15</td>
<td>9.75</td>
<td>10.36</td>
<td>10.17</td>
<td>9.54</td>
</tr>
<tr>
<td>3</td>
<td>29.13</td>
<td>20.56</td>
<td>28.65</td>
<td>20.56</td>
<td>22.07</td>
<td>21.30</td>
<td>20.67</td>
<td>21.60</td>
</tr>
<tr>
<td>4</td>
<td>36.91</td>
<td>4.37</td>
<td>30.48</td>
<td>7.53</td>
<td>7.67</td>
<td>8.01</td>
<td>10.06</td>
<td>9.21</td>
</tr>
<tr>
<td>5</td>
<td>23.91</td>
<td>7.86</td>
<td>22.72</td>
<td>8.63</td>
<td>8.80</td>
<td>8.76</td>
<td>10.99</td>
<td>11.38</td>
</tr>
<tr>
<td>6</td>
<td>35.18</td>
<td>9.78</td>
<td>33.88</td>
<td>13.68</td>
<td>15.19</td>
<td>13.30</td>
<td>12.52</td>
<td>14.02</td>
</tr>
<tr>
<td>7</td>
<td>41.59</td>
<td>12.06</td>
<td>35.47</td>
<td>18.04</td>
<td>18.80</td>
<td>17.03</td>
<td>19.02</td>
<td>19.66</td>
</tr>
<tr>
<td>8</td>
<td>22.52</td>
<td>7.34</td>
<td>21.64</td>
<td>9.01</td>
<td>12.52</td>
<td>9.57</td>
<td>9.47</td>
<td>11.09</td>
</tr>
<tr>
<td>9</td>
<td>43.98</td>
<td>13.94</td>
<td>35.63</td>
<td>19.50</td>
<td>22.36</td>
<td>21.70</td>
<td>20.12</td>
<td>21.38</td>
</tr>
<tr>
<td>10</td>
<td>33.50-35.73†</td>
<td>10.48</td>
<td>27.37</td>
<td>11.93</td>
<td>15.91</td>
<td>13.23</td>
<td>14.73</td>
<td>15.05</td>
</tr>
</tbody>
</table>

†: When the CCP cannot be optimized within the time limit, we report the upper and lower bounds achieved.

The root node gap is closed (in percentage) for each Problem (i) instance after separately adding $v^Q$ and each of the $v^Q_{SGM}$-bounds to the LP relaxation of the MILP reformulation of each chance-
Table 2: Average computational time (in second) for computing bounds for Problem (i) instances

<table>
<thead>
<tr>
<th>$v^*$</th>
<th>$v^Q_{SGM}$</th>
<th>Group Size</th>
<th>OG</th>
<th>RG</th>
<th>AG</th>
<th>SG</th>
<th>KG</th>
<th>DG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$N = 100, P = 2$</td>
<td>36.83</td>
<td>11.69</td>
<td>11.57</td>
<td>12.02</td>
<td>15.92</td>
<td>12.17</td>
</tr>
<tr>
<td>3103.72</td>
<td>10.44</td>
<td>$N = 20, P = 10$</td>
<td>249.27</td>
<td>86.03</td>
<td>59.81</td>
<td>66.46</td>
<td>72.01</td>
<td>107.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$N = 10, P = 20$</td>
<td>909.48</td>
<td>182.61</td>
<td>117.73</td>
<td>160.02</td>
<td>131.88</td>
<td>292.41</td>
</tr>
</tbody>
</table>

constrained program. In Table 4 neither $v^Q$ nor $v^Q_{SGM}$-bounds obtained from heuristic approaches can close much gap at the root of the branch-and-bound tree for optimizing the MILP extended reformulation. When $N = 100$ and $P = 2$, bounds $v^Q_{SGM}$ given by optimal grouping close around 10% optimality gap in all the instances. As the OG bounds become much tighter when we increase $P = 10, 20$ (as shown in Table 1), they significantly improve the root node solution of each MILP reformulation, demonstrated in the last two columns in Table 4. (For presentation brevity, we omit the gaps closed by weaker $v^Q_{SGM}$-bounds given by the heuristic approaches for $N = 20$ and $N = 10$.)

In the following, we repeat the aforementioned procedures on Problem (ii) instances formulated as chance-constrained 0-1 programs with pure binary packing variables. We first compare non-grouping bound $v^Q$ with grouping bounds $v^Q_{SGM}$ by reporting their gaps from the optimal objective value $v^*$, measured by $(v^* - LB)/|v^*|$ where LB takes the value of either $v^Q$ or $v^Q_{SGM}$ in Table 5. We vary the sample size $K = 100, 500, 1000$, and choose group size $P = 10, 20$ for all the grouping methods. Note that different from Problem (i) instances, all the bounds including $v^Q$ are very tight in Table 5. The heuristic-based grouping bounds $v^Q_{SGM}$ are slightly tighter than $v^Q$ for some instances. The optimization grouping bound $v^Q_{SGM}$ is still much tighter than the others, and can be strengthened if we increase $P$ and decrease the number of groups. When we only run the optimal grouping model for 50% of the total time, the quantile bounds are again very close or equal to the ones based on optimal scenario grouping.

We compare the average solution time (in second) of computing each type of bound for $N = 0.1K$ and $P = 10$ in Table 6. The time for computing $v^Q_{SGM}$ bounds by using different heuristic approaches are much shorter than the time of computing $v^Q$ and optimization grouping based $v^Q_{SGM}$ bound. Each method’s solution time almost linearly increases in the number $K$ of scenarios.

5.3 Results of scenario decomposition with grouping

In addition to the quantile bounds, we investigate the effectiveness of scenario grouping used in a finite scenario decomposition algorithm for solving chance-constrained 0-1 programs. The derivation of the algorithm follows the related procedures in Ahmed (2013) for optimizing expectation-based stochastic 0-1 programs in finite steps. Here, we use the quantile bound $v^Q_{SGM}$ as a lower bound in the scenario decomposition approach, while shrinking the feasible region $\mathcal{X}$ after evaluating candidate solutions from solving group subproblems, to guarantee the convergence of the algorithm.
Table 3: Bounds given by different time limits of solving QGP for Problem (i) instances

<table>
<thead>
<tr>
<th>Inst.</th>
<th>OG</th>
<th>OG-10%</th>
<th>OG-20%</th>
<th>OG-30%</th>
<th>OG-40%</th>
<th>OG-50%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group Size: N = 100, P = 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9.75</td>
<td>8.42</td>
<td>8.75</td>
<td>8.75</td>
<td>8.94</td>
<td>8.94</td>
</tr>
<tr>
<td>3</td>
<td>22.80</td>
<td>20.80</td>
<td>21.23</td>
<td>21.23</td>
<td>21.23</td>
<td>22.80</td>
</tr>
<tr>
<td>4</td>
<td>9.34</td>
<td>4.47</td>
<td>5.28</td>
<td>5.28</td>
<td>5.28</td>
<td>6.50</td>
</tr>
<tr>
<td>5</td>
<td>11.45</td>
<td>7.93</td>
<td>7.93</td>
<td>9.32</td>
<td>9.32</td>
<td>10.21</td>
</tr>
<tr>
<td>6</td>
<td>13.32</td>
<td>9.96</td>
<td>10.36</td>
<td>10.36</td>
<td>10.36</td>
<td>11.50</td>
</tr>
<tr>
<td>7</td>
<td>17.36</td>
<td>12.81</td>
<td>13.52</td>
<td>13.52</td>
<td>13.52</td>
<td>14.36</td>
</tr>
<tr>
<td>8</td>
<td>10.63</td>
<td>7.89</td>
<td>8.13</td>
<td>8.13</td>
<td>8.86</td>
<td>8.86</td>
</tr>
<tr>
<td>9</td>
<td>21.67</td>
<td>15.49</td>
<td>15.49</td>
<td>16.90</td>
<td>16.90</td>
<td>16.90</td>
</tr>
<tr>
<td>10</td>
<td>15.25</td>
<td>10.70</td>
<td>11.01</td>
<td>11.01</td>
<td>12.43</td>
<td>12.43</td>
</tr>
</tbody>
</table>

|       |    |        |        |        |        |        |
|       |    |        |        |        |        |        |
| Group Size: N = 20, P = 10 |
| 1     | 24.03 | 11.23  | 15.46  | 18.23  | 18.23  | 19.41  |
| 2     | 11.55 | 9.18   | 10.61  | 10.61  | 10.61  | 11.02  |
| 3     | 25.56 | 21.09  | 23.90  | 23.90  | 23.90  | 25.56  |
| 5     | 17.49 | 9.80   | 14.71  | 14.71  | 16.92  | 17.49  |
| 6     | 21.86 | 13.16  | 18.47  | 18.47  | 19.27  | 19.27  |
| 7     | 24.81 | 15.89  | 19.75  | 21.72  | 20.39  | 22.18  |
| 8     | 15.85 | 9.33   | 12.44  | 12.67  | 13.01  | 13.01  |
| 9     | 27.79 | 18.34  | 23.63  | 23.63  | 23.63  | 23.63  |
| 10    | 19.27 | 12.25  | 15.33  | 15.96  | 15.96  | 16.27  |

|       |    |        |        |        |        |        |
|       |    |        |        |        |        |        |
| Group Size: N = 10, P = 20 |
| 1     | 36.73 | 13.69  | 27.82  | 27.82  | 28.36  | 33.44  |
| 2     | 13.69 | 9.83   | 12.73  | 12.73  | 13.69  | 13.69  |
| 3     | 28.65 | 21.39  | 26.05  | 28.65  | 28.65  | 28.65  |
| 4     | 30.48 | 10.49  | 24.40  | 24.40  | 28.60  | 30.48  |
| 5     | 22.72 | 10.71  | 19.87  | 19.87  | 20.93  | 22.72  |
| 6     | 33.88 | 16.34  | 26.16  | 27.64  | 33.88  | 33.88  |
| 7     | 35.47 | 20.57  | 28.62  | 33.45  | 33.45  | 33.45  |
| 8     | 21.64 | 10.76  | 19.36  | 19.36  | 21.17  | 21.64  |
| 9     | 35.63 | 21.88  | 31.63  | 32.23  | 34.74  | 35.63  |
| 10    | 27.37 | 2.74   | 23.73  | 24.87  | 24.87  | 27.31  |

We denote $u$ and $\ell$ as the upper and lower bounds of the optimal objective value of a given
Table 4: Root gap closed (in percentage) after adding bounds for Problem (i) instances

<table>
<thead>
<tr>
<th>Inst.</th>
<th>(v^Q)</th>
<th>(N = 100)</th>
<th>(N = 20)</th>
<th>(N = 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OG</td>
<td>RG</td>
<td>AG</td>
<td>SG</td>
</tr>
<tr>
<td>1</td>
<td>0%</td>
<td>12%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>3%</td>
<td>9%</td>
<td>3%</td>
<td>3%</td>
</tr>
<tr>
<td>3</td>
<td>4%</td>
<td>8%</td>
<td>4%</td>
<td>4%</td>
</tr>
<tr>
<td>4</td>
<td>0%</td>
<td>11%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>5</td>
<td>0%</td>
<td>9%</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>6</td>
<td>0%</td>
<td>7%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>7</td>
<td>1%</td>
<td>7%</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>8</td>
<td>0%</td>
<td>8%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>9</td>
<td>0%</td>
<td>10%</td>
<td>1%</td>
<td>0%</td>
</tr>
<tr>
<td>10</td>
<td>1%</td>
<td>8%</td>
<td>1%</td>
<td>1%</td>
</tr>
</tbody>
</table>

We use \(P = 10\) and \(N = 0.1K\) for grouping scenarios and using their bounds in scenario decomposition for optimizing Problem (ii) instances. In addition to \(mk-20-10\) instances, we also test \(mk-39-5\) instances that are much harder to optimize also according to Ahmed et al. (2016). We report the computational results of different procedures in Table 7, where Columns \textbf{Non-G}, \textbf{KG (ReG)}, \textbf{KG (w/o ReG)}, \textbf{OG (ReG)}, \textbf{OG (w/o ReG)} depict the average solution time (in second) for optimizing each set of five replications by using the scenario decomposition without grouping, \(K\)-means clustering grouping with or without re-grouping each iteration, optimization grouping with or without iteratively re-grouping, respectively. (Given the previous results, we only run the optimal grouping model for 50% of the total optimization time and use the produced bounds in the scenario decomposition algorithm.) For instances that are not solved within the time limit, we report the average optimality gaps (in percentage) followed by the number of solved instances in the parenthesis under Column \textbf{Gap}. Note that the scenario decomposition with optimization grouping but no-re-grouping can optimize all the instances within the time limit, and thus we omit the column \textbf{Gap} for the last method, which will have “0.0% (5)” for all the rows if being
### Table 5: Bound comparison for Problem (ii) \( mk-20-10 \) instances

<table>
<thead>
<tr>
<th>Inst.</th>
<th>( \epsilon )</th>
<th>( K )</th>
<th>( v^Q )</th>
<th>OG</th>
<th>OG-20%</th>
<th>OG-50%</th>
<th>( v^Q_{\text{SGM}} )</th>
<th>RG</th>
<th>AG</th>
<th>SG</th>
<th>KG</th>
<th>DG</th>
</tr>
</thead>
<tbody>
<tr>
<td>mk-20-10</td>
<td>0.1</td>
<td>100</td>
<td>1.6%</td>
<td>0.5%</td>
<td>0.9%</td>
<td>0.5%</td>
<td>1.1%</td>
<td>1.0%</td>
<td>1.0%</td>
<td>1.0%</td>
<td>0.9%</td>
<td>1.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>1.8%</td>
<td>0.6%</td>
<td>1.3%</td>
<td>0.7%</td>
<td>1.5%</td>
<td>1.4%</td>
<td>1.4%</td>
<td>1.2%</td>
<td>1.4%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>2.0%</td>
<td>0.8%</td>
<td>1.7%</td>
<td>0.8%</td>
<td>1.8%</td>
<td>1.7%</td>
<td>1.6%</td>
<td>1.6%</td>
<td>1.8%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>100</td>
<td>2.3%</td>
<td>0.7%</td>
<td>2.1%</td>
<td>0.8%</td>
<td>2.3%</td>
<td>2.3%</td>
<td>2.2%</td>
<td>2.1%</td>
<td>2.3%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>1.5%</td>
<td>0.3%</td>
<td>1.3%</td>
<td>0.4%</td>
<td>1.5%</td>
<td>1.5%</td>
<td>1.5%</td>
<td>1.4%</td>
<td>1.5%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>2.2%</td>
<td>0.6%</td>
<td>1.9%</td>
<td>0.8%</td>
<td>2.2%</td>
<td>2.2%</td>
<td>2.1%</td>
<td>2.0%</td>
<td>2.2%</td>
<td></td>
</tr>
</tbody>
</table>

| Group Size: \( N = 0.1K, \ P = 10 \) |

<table>
<thead>
<tr>
<th>Inst.</th>
<th>( \epsilon )</th>
<th>( K )</th>
<th>( v^Q )</th>
<th>OG</th>
<th>OG-20%</th>
<th>OG-50%</th>
<th>( v^Q_{\text{SGM}} )</th>
<th>RG</th>
<th>AG</th>
<th>SG</th>
<th>KG</th>
<th>DG</th>
</tr>
</thead>
<tbody>
<tr>
<td>mk-20-10</td>
<td>0.1</td>
<td>100</td>
<td>1.6%</td>
<td>0.3%</td>
<td>0.8%</td>
<td>0.3%</td>
<td>1.1%</td>
<td>1.1%</td>
<td>1.0%</td>
<td>0.8%</td>
<td>1.0%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>1.8%</td>
<td>0.4%</td>
<td>1.2%</td>
<td>0.4%</td>
<td>1.4%</td>
<td>1.4%</td>
<td>1.3%</td>
<td>0.9%</td>
<td>1.3%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>2.0%</td>
<td>0.5%</td>
<td>1.3%</td>
<td>0.5%</td>
<td>1.7%</td>
<td>1.7%</td>
<td>1.6%</td>
<td>1.3%</td>
<td>1.5%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>100</td>
<td>2.3%</td>
<td>0.3%</td>
<td>1.7%</td>
<td>0.4%</td>
<td>2.1%</td>
<td>2.2%</td>
<td>2.2%</td>
<td>1.8%</td>
<td>2.1%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>1.5%</td>
<td>0.2%</td>
<td>1.0%</td>
<td>0.3%</td>
<td>1.5%</td>
<td>1.4%</td>
<td>1.5%</td>
<td>1.1%</td>
<td>1.4%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>2.2%</td>
<td>0.3%</td>
<td>1.7%</td>
<td>0.3%</td>
<td>2.2%</td>
<td>2.0%</td>
<td>2.0%</td>
<td>1.6%</td>
<td>2.0%</td>
<td></td>
</tr>
</tbody>
</table>

| Group Size: \( N = 0.05K, \ P = 20 \) |

### Table 6: Average time (in second) for computing bounds for \( mk-20-10 \) instances with \( N = 0.1K \)

<table>
<thead>
<tr>
<th>Inst.</th>
<th>( \epsilon )</th>
<th>( K )</th>
<th>( v^Q )</th>
<th>OG</th>
<th>RG</th>
<th>AG</th>
<th>SG</th>
<th>KG</th>
<th>DG</th>
</tr>
</thead>
<tbody>
<tr>
<td>mk-20-10</td>
<td>0.1</td>
<td>100</td>
<td>24.09</td>
<td>29.67</td>
<td>6.38</td>
<td>6.51</td>
<td>6.64</td>
<td>7.72</td>
<td>6.49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>143.27</td>
<td>121.77</td>
<td>34.53</td>
<td>34.18</td>
<td>33.49</td>
<td>49.38</td>
<td>33.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>272.99</td>
<td>238.73</td>
<td>63.69</td>
<td>68.15</td>
<td>64.96</td>
<td>100.63</td>
<td>63.05</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>100</td>
<td>24.01</td>
<td>29.56</td>
<td>6.77</td>
<td>6.91</td>
<td>7.04</td>
<td>8.80</td>
<td>6.57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>146.16</td>
<td>211.70</td>
<td>36.28</td>
<td>34.83</td>
<td>36.64</td>
<td>47.16</td>
<td>37.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>277.48</td>
<td>349.51</td>
<td>63.70</td>
<td>65.61</td>
<td>63.06</td>
<td>97.46</td>
<td>64.34</td>
</tr>
</tbody>
</table>
Algorithm 5 A scenario decomposition algorithm based on scenario groups $G_1, \ldots, G_N$.

1: repeat
2: Group scenarios in $\{1, \ldots, K\}$ into $G_1, \ldots, G_N$. Set $S \leftarrow \emptyset$.
3: for $n = 1, \ldots, N$ do
4:    Solve the $G_n$-based subproblem to obtain the optimal objective value $\phi_n$ and an optimal solution $\hat{x}^n$.
5:    Update $S \leftarrow S \cup \{\hat{x}^n\}$.
6: end for
7: Update $\ell \leftarrow$ the $(K' + 1)$th largest of $\phi_1, \ldots, \phi_N$.
8: for $\hat{x} \in S$ do
9:    if $\hat{x}$ satisfies constraint (1b) then
10:       Set $u \leftarrow \min\{u, c^\top x\}$.
11: end if
12: end for
13: Update $X \leftarrow X \setminus S$ (by including no-good cuts into set $X$ in each group subproblem).
14: until $u - \ell \leq \epsilon$

Table 7: Performance of scenario decomposition for Problem (ii) instances with $N = 0.1K$ ($P = 10$)

<table>
<thead>
<tr>
<th>Inst.</th>
<th>$\epsilon$</th>
<th>$K$</th>
<th>Non-G</th>
<th>KG (ReG)</th>
<th>KG (w/o ReG)</th>
<th>OG (ReG)</th>
<th>OG (w/o ReG)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Time (s)</td>
<td>Gap (s)</td>
<td>Time (s)</td>
<td>Gap (s)</td>
<td>Time (s)</td>
</tr>
<tr>
<td>mk-20-10</td>
<td>0.1</td>
<td>100</td>
<td>113.78</td>
<td>0.0%</td>
<td>35.18</td>
<td>0.0%</td>
<td>31.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>377.22</td>
<td>0.0%</td>
<td>134.64</td>
<td>0.0%</td>
<td>120.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>1012.73</td>
<td>0.0%</td>
<td>164.26</td>
<td>0.0%</td>
<td>146.01</td>
</tr>
<tr>
<td>mk-39-5</td>
<td>0.1</td>
<td>100</td>
<td>LIMIT</td>
<td>3.6%</td>
<td>LIMIT</td>
<td>1.2%</td>
<td>LIMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>LIMIT</td>
<td>3.9%</td>
<td>LIMIT</td>
<td>1.5%</td>
<td>LIMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>LIMIT</td>
<td>4.0%</td>
<td>LIMIT</td>
<td>1.9%</td>
<td>LIMIT</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>100</td>
<td>LIMIT</td>
<td>3.4%</td>
<td>LIMIT</td>
<td>1.7%</td>
<td>LIMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>LIMIT</td>
<td>3.2%</td>
<td>LIMIT</td>
<td>1.9%</td>
<td>LIMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>LIMIT</td>
<td>3.8%</td>
<td>LIMIT</td>
<td>1.8%</td>
<td>LIMIT</td>
</tr>
</tbody>
</table>

In Table 7, we use $K$-means clustering to represent other heuristic based grouping methods due to that (i) their solution time are similar in Table 6 and (ii) $K$-means clustering in general leads to tighter $v_Q^{SGM}$ in Table 5. It takes much shorter time on average as compared to using scenario decomposition with no grouping and with optimization based grouping. It is also slightly faster for optimizing $mk-20-10$ instances if we do not re-group scenarios in each iteration. However, when solving $mk-39-5$ instances, the re-grouping procedures can improve the optimality gaps if we cannot optimize the instances within the time limit. When using scenario decomposition with optimization-based grouping, we recommend not re-grouping scenarios in each iteration, since
each re-grouping step requires solving the optimization-based scenario grouping problem by the branch-and-cut algorithm, and will slow down the computation significantly.

6 Conclusion

We investigated optimization driven scenario grouping for strengthening quantile bounds of general chance-constrained programs, and incorporated the method to a finitely convergent algorithm of scenario decomposition for optimizing chance-constrained 0-1 programs. To solve the optimal scenario grouping model, we developed an exact branch-and-cut approach, and also designed heuristic approaches by utilizing data mining and clustering. We also considered a chance-constrained linear programming case, for which the scenario grouping problem can be directly optimized as an MILP. Extensive computational studies were conducted to verify the strengths of objective bounds given by different grouping strategies. The optimal grouping bounds are much tighter than the quantile bounds without grouping and heuristic grouping bounds, also yielding smaller root node gaps and improved implementation of the scenario decomposition algorithm. The optimal grouping bounds become tighter if we increase the group size but take longer time to solve. In fact, the time of computing optimal grouping bounds is much longer than the one of solving non-grouping quantile bounds and heuristic grouping bounds. For most instances, we showed that the bounds, given by feasible solutions obtained after running the solver for 20%-50% of the total solution time, are sufficiently close to the quantile bounds based on optimal scenario groups (with gaps being under 10% of the corresponding optimal grouping bounds).

Future research directions include (i) developing more efficient cutting-plane methods for optimizing the scenario grouping problem; (ii) implementing scenario grouping and decomposition algorithms in distributed computing frameworks; (iii) investigating optimal scenario grouping solution patterns to improve heuristic-based grouping; and (iv) investigating how to generalize the scenario grouping approaches for solving broader classes of risk-averse stochastic programs.

Acknowledgement

The authors thank Dr. Yongjia Song for sharing multi-dimensional knapsack instances used in this paper. The research has been supported in part by National Science Foundation (NSF) grant CMMI-1633196) (Ahmed), ONR N00014-14-0315 (Lee), and the NSF Grant CMMI-1433066 (Shen).

References


24


