Scenario grouping and decomposition algorithms for chance-constrained programs

Yan Deng[*] Huiwen Jia[†] Shabbir Ahmed[‡] Jon Lee[§] Siqian Shen[¶]

February 3, 2020

Abstract

A lower bound for a finite-scenario-based chance-constrained program is 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 solving larger subproblems. The quality of the bound depends on how the scenarios are grouped. In this paper, 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, a mixed-integer linear-programming reformulation is derived. We also propose several heuristics for grouping similar or dissimilar scenarios. Our computational results demonstrate that optimal-grouping bounds are much tighter than heuristic bounds, resulting in smaller root-node gaps and better performance of scenario decomposition for solving chance-constrained 0-1 programs. Also, the optimal grouping bounds can be greatly 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;
[†]Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor;
[‡]School of Industrial and Systems Engineering, Georgia Institute of Technology;
[§]Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor;
[¶]Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor; email: siqian@umich.edu

1 Introduction

We consider general chance-constrained programs:

$$\min \quad c^\top x \tag{1a}$$

$$\text{s.t.} \quad \mathbb{P}\{x \in \mathcal{F}(\xi)\} \geq 1 - \epsilon \tag{1b}$$

$$x \in \mathcal{X} \subseteq \mathbb{R}^d, \tag{1c}$$

where x represents a d-dimensional decision vector, ξ is a multivariate random vector, and $c \in \mathbb{R}^d$ is a given cost parameter. We denote $\mathcal{F}(\xi) \subseteq \mathbb{R}^d$ as some region parameterized by ξ, 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 that can be either continuous or discrete.

The chance-constrained programs (1) are often used for guaranteeing the quality of service and bounding the risk of occurrence of undesirable outcomes in application areas such as energy [Zhang et al., 2017b,a], transportation [Lu et al., 2018, Shen and Chen, 2013], project management and production planning [Shen et al., 2010, Jiang et al., 2017], and healthcare operations management [Deng and Shen, 2016, Deng et al., 2019, Zhang et al., 2018]. However, they are generally nonconvex, except under some special distributions for the random parameter ξ [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 ξ. They analyze the objective bounds and solution feasibility, depending 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. We note that the SAA approach can easily adapt to any types of random vector ξ that follows either a discrete or a continuous distribution.

Assuming finite support of the random vector ξ, 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 that these copies take the same value. Then, a Lagrangian relaxation can be formulated by relaxing the nonanticipativity constraints and can be fully decomposed into scenario-independent subproblems. In addition to relaxing the nonanticipativity constraints, Watson et al. [2010] also relax the knapsack constraint that couples binary indicator 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 work by Ahmed [2013], Carøe and Schultz [1999], Dentcheva and Römisch [2004], Miller and Ruszczyński [2011], Collado et al. [2012], Deng et al. [2017], with different variable settings and risk-preference setups.

Ahmed et al. [2017] provide a comprehensive review of the literature on modeling and solving chance-constrained programs with finite support. They propose new Lagrangian-dual formulations to derive lower bounds for the optimal objective value of the chance-constrained programs (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., 2017], and we 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., 2017, Gade et al., 2016], we formulate a mathematical-optimization model to guide the 0-1 decisions of grouping scenarios, so as to obtain the tightest quantile bounds. To our 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]. Rather, it is 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.

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 we 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 conduct computational studies on diverse sets of chance-constrained problem instances by applying different grouping methods in scenario decomposition. Finally, in Section 6, we make some brief conclusions and outline future research.

2 Preliminaries and Literature on Scenario Grouping

Recall the chance-constrained program (1). In what follows, we assume that (i) the random vector ξ has finite support $\Xi = \{\xi^1, \ldots, \xi^K\}$, where each $k \in \{1, \ldots, K\}$ refers to a *scenario*,

(ii) each scenario is realized with equal probability, $1/K$, and (iii) the feasible region is nonempty. Note that Assumption (ii) is made without loss of generality, because if the scenarios have different probabilities of being realized, we can make sufficient 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 $f(x) : \mathbb{R}^d \to \mathbb{R}$, 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 $\mathbb{I}(\cdot)$ be the indicator function, we rewrite (1) in an equivalent form:

$$\begin{aligned} \mathsf{CCP}: \quad v^* := \min \quad & c^\top x \\ \text{s.t.} \quad & \sum_{k=1}^{K} \mathbb{I}(x \notin \mathcal{F}_k) \leq K' \\ & x \in \mathcal{X}, \end{aligned}$$

where $\mathcal{F}_k := \mathcal{F}(\xi^k)$ 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., 2017]. As previously mentioned 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., 2017]. It was proposed based on the idea that in order to meet the probabilistic requirement, the decision vector x must fall in sufficiently many subregions. The optimal objective values of the K scenario subproblems are

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

These values are ordered to obtain a permutation σ 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 $(K'+1)^{\text{th}}$ quantile value, $\psi_{\sigma_{K'+1}}$, is a valid lower bound for CCP, because the decision vector x will fall in at least one $\mathcal{F}_k$ with $k \in \{\sigma_1, \ldots, \sigma_{K'+1}\}$. The problem of finding such a quantile bound is given by

$$v^{\mathsf{Q}} := \max \left\{ \rho : \sum_{k=1}^{K} \frac{1}{K} \mathbb{I}(\rho \leq \psi_k) > \epsilon \right\} = \max \left\{ \rho : \sum_{k=1}^{K} \mathbb{I}(\rho \leq \psi_k) \geq K' + 1 \right\}. \tag{3}$$

Another way to derive a valid lower bound for CCP is to use a relaxed formulation obtained by using scenario grouping as follows [see Ahmed et al., 2017]. 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:

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

SGM is a relaxation of CCP, because the number of violated groups is at least as many as the number of 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 the 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:

$$v^{\mathsf{Q}}_{\mathsf{SGM}} := \max \left\{ \rho : \sum_{n=1}^{N} \mathbb{I}\left(\rho \leq \phi_n\right) \geq K' + 1 \right\}, \tag{4}$$

where

$$\phi_n := \min \left\{ c^\top x : x \in \bigcap_{k \in G_n} \mathcal{F}_k,\ x \in \mathcal{X} \right\},\ n = 1, \ldots, N. \tag{5}$$

Model (5) for each n is called a *group subproblem* in this paper. The grouping-based quantile bound $v^{\mathsf{Q}}_{\mathsf{SGM}}$ is a valid lower bound for CCP (i.e., $v^{\mathsf{Q}}_{\mathsf{SGM}} \leq v_{\mathsf{SGM}} \leq v^*$).

2.2 Literature on scenario grouping

In scenario grouping, how to group scenarios is a crucial 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] demonstrate the advantages of using input parameters to characterize a scenario (as compared to using an optimal solution), and the advantages of using groups that form a cover of scenarios (as compared to those that form a partition).

Ryan et al. [2016] is the first to 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 number of subproblems to bundle and demonstrate 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. [2017] 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 MILP solved in Ryan et al. [2016] when using optimal grouping for expectation-based stochastic programs. To implement the scenario grouping more efficiently, we develop both exact algorithms using branch-and-cut and heuristic approaches. 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.

3 Optimization-based Scenario Grouping

Recall that K is the number of scenarios, and N is the number of groups of disjoint subsets of scenarios of $\{1, \ldots, K\}$. We let $K' = \lfloor \epsilon K \rfloor$, where ϵ is the risk tolerance level in the chance-constrained program (1). We define binary 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 calculate 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 $v^{\mathsf{Q}}_{\mathsf{SGM}}$, which is $\phi_{K'+1}$. The optimization model is

$$\begin{aligned}
\mathsf{QGP}: \quad \max \quad & \phi_{K'+1} && \text{(6a)}\\
\text{s.t.} \quad & \phi_n \le \min\{c^\top x : x \in \mathcal{X};\ \mathbb{I}(x \in \mathcal{F}_k) \ge y_{kn},\ \forall k = 1, \ldots, K\} \quad \forall n = 1, \ldots, N && \text{(6b)}\\
& \phi_n - \phi_{n+1} \ge 0 \quad \forall n = 1, \ldots, N-1 && \text{(6c)}\\
& \sum_{n=1}^{N} y_{kn} = 1 \quad \forall k = 1, \ldots, K && \text{(6d)}\\
& 1 \le \sum_{k=1}^{K} y_{kn} \le P \quad \forall n = 1, \ldots, N && \text{(6e)}\\
& y_{kn} \in \{0, 1\} \quad \forall n = 1, \ldots, N,\ k = 1, \ldots, K. && \text{(6f)}
\end{aligned}$$

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 $\mathbb{I}(x \in \mathcal{F}_k) = 1$ and thus $x \in \mathcal{F}_k$. Otherwise, the constraints $x \in \mathcal{F}_k$ are relaxed in the corresponding group subproblems to allow larger ϕ_n-values and thus larger quantile bound $\phi_{K'+1}$ maximized in the objective function (6a). The purpose of (6c) is to exclude symmetric grouping solutions, where we order the optimal objective values of group subproblems non-increasingly. Constraints (6d) ensure that we assign every scenario to a group. In (6e), we restrict each group size by an integer parameter $P \ge K/N$. Without this restriction, 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]. Additionally, (6e) also avoids empty groups with no scenarios.

In Section 3.1, we provide complexity results for QGP, depending on the value of P (i.e., the maximum number of scenarios allowed in each group). In Section 3.2, 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 3.3, we propose a branch-and-cut algorithm for optimizing QGP for general chance-constrained programs of the form (1).

3.1 Complexity of QGP in (6)

In what follows, we assume that each scenario subproblem (2) is polynomial-time solvable. Note that the group subproblem (5) with $|G_n| = 1$ is exactly the scenario subproblem (2), and therefore (5) is at least as hard as (2). Next we demonstrate that optimizing QGP with

$P = 2$ can be done in polynomial time (see Proposition 1), but QGP with $P \geq 3$ is $\mathcal{NP}$-hard (see Theorem 1).

Proposition 1 *If the group subproblems* (5) *can be solved in polynomial time, then QGP with* $P = 2$ *is polynomial-time solvable.*

Proof: The proof consists of four parts. We first provide an efficient transformation from any given QGP instance with $P = 2$ into a matching problem, and we describe algorithmic steps for finding an optimal matching on the graph. We demonstrate that the output of the algorithm corresponds to an optimal solution to the original QGP, and then we derive the complexity of the algorithm, which is polynomial in the problem size.

(i) Graph construction. Given any QGP instance with $P = 2$, for every pair of scenarios $k_1, k_2 \in \{1, \ldots, K\}$ and $k_1 \neq k_2$, we compute the effect of grouping them as $\varepsilon((k_1, k_2)) = \min\{c^\top x : x \in \mathcal{F}_{k_1} \cap \mathcal{F}_{k_2},\ x \in \mathcal{X}\}$. We then define a complete graph $G'(V', E')$, where vertices correspond to all scenarios, and the weight of each edge $(k_1, k_2) \in E'$ is computed by $\varepsilon((k_1, k_2))$. Moreover, for each scenario node k in V', we construct a duplicated node k' that is only connected to node k by edge $(k, k') \in E''$. We define the set of all duplicated nodes as V'', and we compute $\varepsilon((k, k')) = \min\{c^\top x : x \in \mathcal{F}_k,\ x \in \mathcal{X}\}$ as the weight of edge $(k, k') \in E''$. As a result, $|E'| = \frac{K(K-1)}{2}$ and $|E''| = K$. The overall graph $G(V, E)$ is given by $V = V' \cup V''$ and $E = E' \cup E''$. If the group subproblem in (5) can be solved via polynomial subroutines, the graph construction steps can be completed in polynomial time. Then the given problem QGP with $P = 2$ is equivalent to finding a matching in G such that the $(K' + 1)^{\text{th}}$ largest weight of edges in the matching is maximized.

(ii) Algorithm design. We develop an algorithm for solving the matching problem where we maximize the $(K' + 1)^{\text{th}}$ largest weight among its edges. First, we sort all the edges in E following a non-increasing order of their weights: $\varepsilon(e_1) \geq \varepsilon(e_2) \geq \ldots \geq \varepsilon(e_{|E|})$. We pick $e_1, \ldots, e_{K'+1}$ and their endpoints to form a subgraph $\bar{G}(\bar{V}, \bar{E})$. Then, we find a matching in $\bar{G}$ of maximum cardinality. If the size of the matching, i.e., the number of the edges in the matching, is equal to $K' + 1$, we terminate the algorithm and the smallest weight among all the matched edges is the optimal objective value of QGP in (6); otherwise, we add an edge not in $\bar{E}$ with the largest weight to $\bar{E}$, updating $\bar{G}$. We repeat the above steps until the maximum-cardinality matching of $\bar{G}$ contains exactly $K' + 1$ edges. As a result, we find an optimal matching of G, where the $(K' + 1)^{\text{th}}$ largest weight in the matching is maximized. We provide algorithmic details in Algorithm 1.

(iii) Algorithm validation. When the algorithm terminates, the matching edges form groups of scenarios, and we can directly let the remaining nodes in V' be match to their

Algorithm 1 Procedures of solving QGP when $P = 2$.

Input: $\mathcal{F}_k, \forall k = 1, \ldots, K,\ \mathcal{X},\ c,\ K'$, subroutines for solving (5).
Construct graph $G(V, E)$, V' and V'', compute $\varepsilon(e),\ \forall e \in E$.
Sort edges in E by their weights such that $\varepsilon(e_1) \geq \varepsilon(e_2) \geq \ldots \geq \varepsilon(e_{|E|})$.
Define function $\tilde{V}(\cdot)$ to return endpoints of input edges.
Initialize $\bar{E} \leftarrow \{e_1, \ldots, e_{K'+1}\}$, $\bar{V} \leftarrow \tilde{V}(\bar{E})$, $l = K' + 1$.
Find a maximum-cardinality matching M^* in graph $\bar{G}(\bar{V}, \bar{E})$.
if $|M^*| < K' + 1$ **then**
    $\bar{E} \leftarrow \bar{E} \cup \{e_{l+1}\}$, $\bar{V} \leftarrow \tilde{V}(\bar{E})$, $l \leftarrow l + 1$, go to Step 6.
end if
return Matching M^* and the optimal objective value $\varepsilon(e_l)$.

duplicated nodes in V''. Therefore, we build a matching M^* that matches all scenario nodes in V'. Each edge in M^* corresponds to a group with scenarios corresponding to the two endpoints of the edge. Without loss of optimality, N is the cardinality of M^*, because we do not have groups with no scenarios. After sorting the edges in M^* by weights such that $\varepsilon(e_1^*) \geq \cdots \geq \varepsilon(e_N^*)$, then for $n = 1, \ldots, N$, (i) ϕ_n^* in QGP equals to $\varepsilon(e_n^*)$; (ii) if edge e_n^* is (k_1, k_2), then $y_{k_1 n}^* = y_{k_2 n}^* = 1$; (iii) if edge e_n^* is (k, k'), then $y_{kn}^* = 1$; and (iv) all other y^* variables equal to zero.

We first demonstrate that such an assignment of solution values is feasible to QGP in (6). Constraints (6b) are naturally satisfied according to the definition of edge weights. Constraints (6c) are satisfied because we sort all the edges by their weights. Moreover, constraints (6d) are satisfied because every node (scenario) in V' is connected to another node (scenario) by an edge, and in M^* each node can be an endpoint of at most one edge. Finally, we satisfy (6e) because in each group we can only have one or two scenarios (i.e., $P = 2$).

Now we demonstrate the optimality of the solution (ϕ^*, y^*) by contradiction. Suppose that there exists another feasible solution (ϕ', y') having a larger objective value than (ϕ^*, y^*). Without loss of generality, we suppose that constraints (6b) are tight at (ϕ', y'), and we can use the values of y' for $n = 1, \ldots, K' + 1$ to recover a matching M' of size $K' + 1$ in $G(V, E)$. Because $\phi'_{K'+1} > \phi^*_{K'+1}$, M' only consists of edges with larger weights than $\phi^*_{K'+1}$. Because (ϕ^*, y^*) is obtained by Algorithm 1, the algorithm termination criterion indicates that we cannot form a matching of size $K' + 1$ when $\bar{E}$ only contains the edges with larger weights than $\phi^*_{K'+1}$. This contradicts the fact that there exists a valid matching M' of size $K' + 1$, which implies that the objective value of (ϕ', y') is larger than the one of (ϕ^*, y^*). Therefore, we conclude that any solution produced by Algorithm 1 is optimal to QGP.

(iv) Algorithm complexity. We first analyze the algorithmic complexity for the matching problem. The sorting of edges can be implemented with heap sort in time $O(|E|\log(|E|))$ [see Knuth, 1998], here $|E| = \frac{K(K-1)}{2} + K$. For the number of iterations in Algorithm 1, i.e., where we update $\bar{G}$ and solve a maximum-cardinality matching problem, the best case happens when the first $K'+1$ edges form a matching, and the worst case happens when $\bar{E}$ includes as many as possible edges before we can match $K'+1$ edges. Given the special shape of the constructed graph $G(V,E)$, we claim that the worst case happens when the algorithm terminates with $|\bar{E}_{\text{worst}}| = KK' - \frac{K'(K'-1)}{2} + 1$ (proof below). The largest number of iterations is $|\bar{E}_{\text{worst}}| - (K'+1) + 1$, where $K'+1$ is the initial size of $\bar{E}$. Within each iteration, we solve a maximum-cardinality matching problem in worst-case time proportional to $|\bar{V}||\bar{E}|$ using Edmonds' blossom algorithm [see Edmonds, 1965]. According to the proof of the worst case below, we have $|\bar{V}_{\text{worst}}| = K + K' + 1$, and the time in each iteration is at most $O(|\bar{V}_{\text{worst}}||\bar{E}_{\text{worst}}|)$. To conclude, the time for solving the matching problem is proportional to $(\frac{K(K-1)}{2} + K)\log(\frac{K(K-1)}{2} + K) + (KK' - \frac{K'(K'-1)}{2} + 1 - K')^2(K + K' + 1)$, which is $O(K'^2K^3)$.

Worst-case complexity analysis: First, we demonstrate that any subgraph of G which can match at most K' edges, has at most $KK' - \frac{K'(K'-1)}{2}$ edges. The K'-cardinality matching contains $2K'$ nodes and K' edges. If the subgraph contains nodes outside the matched ones, then they can only be connected to the same endpoint of a matched edge. In addition, the remaining endpoints cannot be connected to each other. Otherwise, we can find an alternating path and increase the number of matched edges by at least 1. Recall that in G, subgraph $G'(V', E')$ is a complete graph and nodes in V'' are duplicated nodes of V', only connecting with the corresponding origin nodes. Consequently, the largest number of edges to which one endpoint of a matched edge can be connected without increasing the cardinality of the matching is $K-1$. This indicates that in total we can have $K'(K-1)$ more edges. However, every two matched edges have one double-counted edge, and thus in total $\frac{K'(K'-1)}{2}$ edges are double counted. To conclude, we can have K' matched edges and at most $K'(K-1)$ edges that connect to one endpoint of each matched edge including $\frac{K'(K'-1)}{2}$ double-counted edges, which equals to $K' + K'(K-1) - \frac{K'(K'-1)}{2} = K'K - \frac{K'(K'-1)}{2}$ edges.

Second, we demonstrate that we can find a subgraph $\bar{G}$ with exactly $KK' - \frac{K'(K'-1)}{2}$ edges in G so that we can match at most K' edges. Our procedures of constructing $\bar{E}$ and $\bar{V}$ are as follows. (i) We add arbitrary K' nodes from V' to $\bar{V}$, which are denoted as $\bar{V}_1$, and add all edges connecting the nodes in $\bar{V}_1$ into $\bar{E}$. (ii) We add the duplicated nodes in V'' of $\bar{V}_1$ to $\bar{V}$ and add the corresponding K' edges that connect those duplicated nodes to nodes in $\bar{V}_1$ to $\bar{E}$. (iii) We add the remaining $K - K'$ nodes in V' to $\bar{V}$ and add the edges that connect those newly added nodes with the nodes in $\bar{V}_1$ to $\bar{E}$. As a result, we have $\frac{K'(K'-1)}{2} + K' + K'(K-K') = K'K - \frac{K'(K'-1)}{2}$ edges in $\bar{G}(\bar{V}, \bar{E})$. The maximum-cardinality matching of $\bar{G}$ contains K' edges that connect the nodes in $\bar{V}_1$ and

their duplicated nodes. The new edge in the next iteration can be either in E' with two end points in $V' \backslash \bar{V}_1$ or in E''. In both two cases, we can directly add this edge to the current matching and increase the cardinality by 1. Therefore, when the algorithm terminates we have $|\bar{E}_{\text{worst}}| = KK' - \frac{K'(K'-1)}{2} + 1$ edges. Moreover, the largest size of $\bar{V}$ is realized when the new added edge is in E'', where $|\bar{V}_{\text{worst}}| = K + K' + 1$.

We denote the complexity of solving the group subproblem (5) by $O(\tau(d))$, where $\tau(d)$ is a polynomial function of the problem size d that is proportional to the number of variables and constraints in (5). The graph transformation, where $|E| = \frac{K(K-1)}{2} + K$, can be completed in time proportional to $(\frac{K(K-1)}{2} + K)\tau(d)$. By sequentially constructing the graph and applying Algorithm 1 for solving the matching problem, we conclude that QGP is solvable in time $O((\frac{K(K-1)}{2} + K)\tau(d) + K'^2K^3)$ when $P = 2$. □

In online supplement Appendix A, we provide an example to illustrate the graph transformation and procedures of Algorithm 1 for QGP with $P = 2$.

To demonstrate that QGP is $\mathcal{NP}$-hard when $P \geq 3$, we first state the decision version of QGP with $P \geq 3$ as follows.

Definition 1 `QGP-P`*: Given K scenarios, an integer K', and a scalar B, is there a scenario-grouping, where the size of each group is at most P, such that the $(K'+1)^{th}$ largest optimal objective value of* (5) *is at least as large as B?*

We demonstrate that `QGP-P` is $\mathcal{NP}$-complete by reducing an arbitrary instance of `Partition into Isomorphic Subgraphs` (see Definition 2) in polynomial time to a special case of `QGP-P`.

Definition 2 `Partition into Isomorphic Subgraphs`*: Given a graph $G(V, E)$ and $H(V', E')$, with $|V| = q \times |V'|$, for some integer q. Can the vertices of G be partitioned into q disjoint sets $V_1, \ldots, V_q$, such that, for $1 \leq i \leq q$, the subgraph of G induced by V_i is isomorphic to H?*

Theorem 1 *QGP in* (6) *is $\mathcal{NP}$-hard when $P \geq 3$.*

Proof: We show that any `Partition into Isomorphic Subgraphs` instance can be polynomially reduced to a special case of `QGP-P`. For any `Partition into Isomorphic Subgraphs` instance, we construct an instance of `QGP-P` by letting $K = |V|$, $K' + 1 = q$, $P = |V'|$, and $B = 1$. For any group of P different scenarios $(k_i : k_i = 1, \ldots, K,\ i = 1, \ldots, P)$, we define the optimal objective value of the group subproblem as follows.

- $\varepsilon(k_i : k_i = 1, \ldots, K,\ i = 1, \ldots, P) = 1$ if the subgraph induced by nodes $\{k_i : k_i = 1, \ldots, K,\ i = 1, \ldots, P\}$ is isomorphic to H.

- $\varepsilon(k_i : k_i = 1, \ldots, K,\ i = 1, \ldots, P) = 0$ otherwise.

For any other groups with fewer than P scenarios, we define the optimal objective value of group subproblems as zero. Whether the $(K' + 1)^{\text{th}}$ largest objective value of group subproblem is as large as 1 is equivalent to whether there exists a `Partition into Isomorphic Subgraph.` If a partition exists, it forms a grouping decision in which every P scenarios that are partitioned in one set form a group, and the answer to `QGP-P` is yes; if a partition does not exist, then we cannot find $K' + 1$ groups with grouping objectives as large as one and thus the answer to `QGP-P` is no. Garey and Johnson [1979] demonstrate that `Partition into Isomorphic Subgraphs` is $\mathcal{NP}$-complete for any fixed H that contains at least three vertices. Therefore, `QGP-P` with $P \geq 3$ is at least as hard as `Partition into Isomorphic Subgraphs`, and is $\mathcal{NP}$-hard. This completes the proof. □

In online supplement Appendix B, we provide an example to illustrate the graph transformation and the problem reducing procedure of `Partition into Isomorphic Subgraphs` for QGP with $P = 3$.

3.2 Reformulating QGP as an MILP 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

$$\mathcal{F}_k = \{x : A_k x \geq r_k\},$$

where $A_k \in \mathbb{R}^{m_k \times d}$ and $r_k \in \mathbb{R}^{m_k}$ for $k = 1, \ldots, K$. 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

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

$$\text{s.t.} \quad \sum_{k=1}^{K} A_k^\mathsf{T} \lambda_{kn} \leq c \tag{8b}$$

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

We further define auxiliary decision vectors $w_{kn} \in \mathbb{R}_+^{m_k}$ such that $w_{kn} \equiv \lambda_{kn} y_{kn}$, $\forall k = 1, \ldots, K$, $n = 1, \ldots, N$. Using the McCormick inequalities [McCormick, 1976], we can linearize dual problem (8) and derive an equivalent MILP to replace the right-hand side of (7) for each $n = 1, \ldots, N$:

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

$$\text{s.t.} \quad (8b)$$

$$w_{kn} \leq \lambda_{kn}, \; w_{kn} \leq \overline{\lambda}_{kn} y_{kn}, \quad \forall k = 1, \ldots, K \tag{9b}$$

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

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

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

$$\overline{\lambda}_{knj} = \min_{i=1,\ldots,d} \left\{ c_i / (A_k^\mathsf{T})_{ij} \right\}$$

as the j^{th} element of the vector $\overline{\lambda}_{kn}$, $j = 1, \ldots, m_k$, where $(A_k^\mathsf{T})_{ij}$ denotes the entry in row i, column j of matrix A_k^T, for each $i = 1, \ldots, d$ and $j = 1, \ldots, m_k$. Note that the objective function (9a) maximizes the values of the w-variables, and therefore constraints (9c) are redundant, and we delete them in our later computational studies.

Overall, for chance-constrained linear programs, we can directly solve QGP as the following MILP with continuous variables ϕ, λ, w, and binary variables y:

$$\max_{\phi, \lambda, w, y} \left\{ \phi_{K'+1} : \phi_n \leq \sum_{k=1}^{K} \left(r_k^\mathsf{T} \lambda_{kn} - M_k^\mathsf{T} \lambda_{kn} + M_k^\mathsf{T} w_{kn} \right), \; n = 1, \ldots, N, \; \text{(6c)–(6e)}, \; \text{(8b)}, \; \text{(9b)}, \; \text{(9d)} \right\}. \tag{10}$$

We note that the MILP 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.

3.3 A branch-and-cut approach for optimizing general 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:

$$\max\{\phi_{K'+1} : (6c)\text{–}(6e),\ (y,\phi) \in \mathcal{A},\ y_{kn} \in \{0,1\},\ \forall k = 1,\dots,K,\ n = 1,\dots,N\}, \tag{11}$$

where we rewrite constraint (6b) as $(y,\phi) \in \mathcal{A}$ and enforce it by iteratively adding linear inequalities of (y,ϕ), which are cutting planes obtained from 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 $\mathcal{A}$), we consider and define a group set $G_n^* := \{k \in \{1,\dots,K\} : \hat{y}_{kn} > 0\}$ for each $n = 1,\dots,N$. Then we compute the value of

$$\phi_n^* = \min\left\{c^\top x : x \in \bigcap_{k \in G_n^*} \mathcal{F}_k,\ x \in \mathcal{X}\right\}. \tag{12}$$

If $\phi_n^* < \hat{\phi}_n$, following Laporte and Louveaux [1993], we add a cut

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

to the relaxed master problem, where U is a finite value chosen to satisfy

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

By the assumption that the set $\mathcal{X}$ is bounded, we set $U = \max\{c^\top x : x \in \mathcal{X}\}$ in our later computation.

Proposition 2 *Cut* (13) *is valid for* QGP.

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

$$\begin{aligned}
\phi_n &\le \min\left\{c^\top x : x \in \bigcap_{k:y_{kn}=1} \mathcal{F}_k,\ x \in \mathcal{X}\right\} \\
&\le \min\left\{c^\top x : x \in \bigcap_{k:\hat{y}_{kn}=1} \mathcal{F}_k,\ x \in \mathcal{X}\right\} \\
&= \phi_n^*.
\end{aligned}$$

Algorithm 2 A branch-and-cut algorithm for optimizing QGP and bounding CCP

Initialize $\overline{\texttt{obj}} \leftarrow +\infty$.
Initialize NodeList $\leftarrow \{\nu_0\}$, where $F_0(\nu_0) = \emptyset,\ F_1(\nu_0) = \emptyset$.
Choose a branching node $\nu \in$ NodeList.
Solve $\mathsf{NRP}(\nu)$ and obtain a solution $(\hat{y}, \hat{\phi})$.
Compute $\phi^*_1, \ldots, \phi^*_N$, and sort them in a non-increasing order.
if $\phi^*_{K'+1} < \overline{\texttt{obj}}$ **then**
  if $\hat{y}$ is integral **then**
    $\overline{\texttt{obj}} \leftarrow \phi^*_{K'+1}$.
  else
    Branch on an arbitrarily chosen binary variable having a fractional solution value, and add two new branching nodes to NodeList.
    if for any n, $\phi^*_n < \hat{\phi}_n$ **then**
      Add Cut (13) into $(y, \phi) \in \widetilde{\mathcal{A}}$.
      Go to Step 4.
    end if
  end if
end if
NodeList $\leftarrow$ NodeList$\backslash\{\nu\}$.
if NodeList is empty **then**
  Report $\overline{\texttt{obj}}$ as the optimal objective value.
else
  Go to Step 3.
end if

This verifies that the inequality (13) is satisfied and completes the proof. □

We incorporate the valid inequality (13) in a branch-and-cut framework for optimizing QGP, detailed in Algorithm 2, where $\overline{\texttt{obj}}$ holds an incumbent upper bound. At each branching node ν, we compute the following relaxation problem:

$$
\begin{aligned}
\mathsf{NRP}(\nu): \quad \max \quad & \phi_{K'+1} && \text{(14a)}\\
\text{s.t.} \quad & \text{(6d), (6e), (6c)} \\
& (y, \phi) \in \widetilde{\mathcal{A}} \subset \mathcal{A} && \text{(14b)}\\
& 0 \le y_{kn} \le 1,\ \forall k = 1, \ldots, K,\ n = 1, \ldots, N && \text{(14c)}\\
& y_{kn} = 0, \forall (k,n) \in F_0(\nu), \quad y_{kn} = 1, \forall (k,n) \in F_1(\nu), && \text{(14d)}
\end{aligned}
$$

where set $\widetilde{\mathcal{A}}$ in constraints (14b) collects all the cuts (13) that were added to the branching node ν up to the current iteration. Constraints (14d) indicate all the y-variables being fixed

at 0 and 1, given 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 $\{1, \ldots, K\} \times \{1, \ldots, N\}$.)

4 Heuristic-based Scenario Grouping

We describe a few heuristic approaches to group scenarios, as alternatives to solving QGP, and later we compare their results to demonstrate the advantages of optimal grouping over the heuristic methods via extensive computational studies. 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 group dissimilar scenarios following a greedy heuristic. In what follows, similarity/dissimilarity is measured by the L^1-norm distance between vectors describing 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 $\psi_{k_1} \geq \cdots \geq \psi_{k_K}$. Clearly, because of the preliminaries in Section 2.1, $\psi_{k_{K'+1}}$ 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 ensure that no group size exceeds P. Following such grouping procedures, the resulting SGM has a quantile bound that is at least $\psi_{k_{K'+1}}$, which we demonstrate in Proposition 3.

Proposition 3 *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 $\psi_{k_1} \geq \cdots \geq \psi_{k_K}$, the quantile bound of SGM is at least as large as the quantile bound of CCP.*

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

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

Then it follows that the quantile bound of SGM, which is the $(K'+1)^{\text{th}}$ largest of $\{\phi_1, \ldots, \phi_N\}$, is at least as large as the quantile bound of CCP, which is the $(K'+1)^{\text{th}}$ largest of $\{\psi_{k_1}, \ldots, \psi_{k_N}\}$. □

Next, we group scenarios based on their similarity or dissimilarity. Let $v^1, \ldots, v^K$ be the vectors characterizing scenario features. For example, in the chance-constrained programs,

the differences between scenarios can be characterized by realizations of the random vector ξ, and therefore we can set $(v^1, \ldots, v^K) = (\xi^1, \ldots, \xi^K)$. We 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, and 4.3, of which the first two aim to group 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 (5) is similar. We therefore attempt to form evenly-sized groups, with the size of each G_n given by

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

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 by at most by 1. To populate each group, we start by randomly picking an ungrouped scenario to start a group, and then we 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 a group $G \subseteq \{1, \ldots, K\}$, denoted by $\overline{v}$, is defined as the arithmetic mean of the characterizing vectors of the contained scenarios, i.e.,

$$\overline{v} = (1/|G|) \sum_{k \in G} v^k.$$

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

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

Algorithm 3 outlines the procedures of the greedy algorithm for grouping similar scenarios.

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.

Algorithm 3 A greedy approach for grouping similar scenarios.

Initialize $S \leftarrow \{1, \ldots, K\}$.
for $n = 1, \ldots, N$ **do**
  Update $\hat{k} \leftarrow$ a randomly chosen scenario from S.
  Update $G_n \leftarrow \{\hat{k}\}$, and $S \leftarrow S \setminus \{\hat{k}\}$. Compute s_n according to (15).
  for $s_n - 1$ times **do**
    Compute $\overline{v} \leftarrow$ the center of G_n.
    Choose $k^* \leftarrow$ the nearest ungrouped scenario to $\overline{v}$.
    Update $G_n \leftarrow G_n \cup \{k^*\}$, and $S \leftarrow S \setminus \{k^*\}$.
  end for
end for

However, finding an exact solution to this problem is $\mathcal{NP}$-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.

We detail our K-means clustering algorithm for grouping similar scenarios in Algorithm 4, 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++ algorithm assigns scenarios in CCP (viewed as "input 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 because 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 Ω groups as $G'_1, \ldots, G'_\Omega$. Then we collect one scenario from each group G'_ω ($\omega \in \{1, \ldots, \Omega\}$) to form a new group G_n, which then consists of Ω "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.

Algorithm 4 A K-means clustering approach for grouping similar scenarios.

Initialize $S \leftarrow \{1, \ldots, K\}$.
Initialize $\hat{k} \leftarrow$ a random scenario chosen from S.
Initialize $\overline{v}_1 \leftarrow v^{\hat{k}}$, and $S \leftarrow S \setminus \{\hat{k}\}$.
for $n = 2, \ldots, N$ **do**
  for $k \in S$ **do**
    Update $d(k) \leftarrow \min_{n' \in \{1,\ldots,n-1\}} \|v^k - \overline{v}_{n'}\|$.
  end for
  Update $k^* \leftarrow$ a randomly chosen scenario from S with probability proportional to $d(k)^2$.
  Update $\overline{v}_n \leftarrow v^{k^*}$, and $S \leftarrow S \setminus \{k^*\}$.
end for
repeat
  for $k = 1, \ldots, K$ **do**
    Update $n^* \leftarrow \arg\min_{n \in \{1,\ldots,N\}} \|v^k - \overline{v}_n\|$.
    Update $G_{n^*} \leftarrow G_{n^*} \cup \{k\}$.
  end for
  for $n = 1, \ldots, N$ **do**
    Update $\overline{v}_n \leftarrow$ the center of G_n.
  end for
until no change from the previous iteration.

To make this method comparable with the three heuristics we introduced in Section 4.1 and Section 4.2, we fix the total number of groups at N. As a result, it becomes important to restrict the size of 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 dissimilar 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

$$\begin{array}{ll} s_1, \ldots, s_\Omega := N & \text{if } K_{\text{mod}} N = 0 \\ s_1, \ldots, s_{\Omega-1} := N,\ s_\Omega := K_{\text{mod}} N & \text{otherwise.} \end{array} \tag{16}$$

The detailed algorithmic steps are presented in Algorithm 5.

5 Numerical Studies

Now we evaluate the proposed bounds and algorithms enabled by scenario grouping on two classes of chance-constrained programming problems: (i) chance-constrained portfolio opti-

Algorithm 5 A greedy approach for grouping dissimilar scenarios.

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

mization, 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 3.3 to optimize the scenario-grouping decisions.

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 Problems (i) and (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 in the Stochastic Integer Programming Test Problem Library (SIPLIB, Ahmed et al. [2016]). We consider

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & \mathbb{P}\left\{(a(\xi))^\top x \geq r\right\} \geq 1 - \epsilon \\ & x \in \mathcal{X} = \left\{x \in \mathbb{R}^d_+ : e^\mathsf{T} x = 1\right\}, \end{aligned} \tag{17}$$

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.

Model (17) is a chance-constrained linear program studied in Section 3.2, 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 $A_k = a_k^\intercal$, $r_k = r$, leading to a single-row linear constraint $a_k^\intercal x \geq r$ in each $\mathcal{F}_k$. We set $M_k = r_k$ for all k and obtain simplified version of Model (7) as follows.

$$\phi_n \leq \min \left\{ c^\top x : a_k^\intercal x \geq r y_{kn},\ \forall k = 1, \ldots, K,\ e^\intercal x = 1,\ x \in \mathbb{R}_+^d \right\}.$$

We then follow similar procedures in Section 3.2 to dualize the right-hand side minimization problem, linearize the bilinear dual model, and optimize scenario grouping by solving an MILP.

We follow the parameter settings in Qiu et al. [2014] detailed 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, maximizing the total value, and also guaranteeing 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. [2017], 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. Because the results are similar among different replications, we report the averages. We vary the risk parameter $\epsilon \in \{0.1, 0.2\}$ for each type of instances and scenario size.

Source code and examples of the Problem (i) and Problem (ii) instances can be downloaded from `http://www-personal.umich.edu/~siqian/docs/scen_decom_data_source_code.zip`. 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 Problems (i) and (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 we report the results under Column v^*;
- We solve the quantile bound of CCP following (3), and we report the results under Column v^{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 3.3, both to optimize QGP and obtain optimization-based grouping solutions. We obtain the quantile bound $v^{\mathsf{Q}}_{\mathsf{SGM}}$ as the optimal objective value of QGP, and we report related results under Column **OG**;
- We construct N groups by distributing scenarios to each group in a round-robin manner, i.e.,

  $$G_n = \{k : (k-1)_{\text{mod}} N = (n-1)\}\,,\ \forall n = 1, \ldots, N.$$

  We then compute the quantile bound $v^{\mathsf{Q}}_{\mathsf{SGM}}$ 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

  $$G_n = \{k_i : (i-1)_{\text{mod}} N = (n-1)\}\,,\ \forall n = 1, \ldots, N,$$

  and by following the heuristics in Sections 4.2.1, 4.2.2, 4.3 to group similar or dissimilar scenarios. We then compute $v^{\mathsf{Q}}_{\mathsf{SGM}}$ 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 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 for Instance No.10, all are solved to optimality within the one-hour time limit. Recall that $K = 200$ for all Problem (i)

Table 1: Bounds given by optimal and heuristic-grouping strategies for Problem (i) instances

Inst.	v^*	v^{Q}	$v^{\mathsf{Q}}_{\mathsf{SGM}}$ OG	RG	AG	SG	KG	DG
			Group Size: $N = 100, P = 2$					
1	40.7	9.31	15.73	9.74	9.40	9.40	8.89	9.82
2	15.36	8.42	9.75	8.52	8.47	8.49	8.28	8.59
3	29.13	20.56	22.80	20.57	20.61	20.57	20.57	20.80
4	36.91	4.37	9.34	4.56	4.51	4.48	4.47	4.66
5	23.91	7.86	11.45	8.55	8.16	7.86	7.89	8.20
6	35.18	9.78	13.32	9.96	9.96	9.85	9.81	10.02
7	41.59	12.06	17.36	12.20	12.25	12.20	12.24	12.66
8	22.52	7.34	10.63	7.47	7.68	7.49	7.53	7.66
9	43.98	13.94	21.67	14.68	14.11	14.06	14.56	15.09
10	33.50-35.73†	10.48	15.25	10.48	10.70	10.61	10.70	10.83
			Group Size: $N = 20, P = 10$					
1	40.7	9.31	24.03	9.38	11.81	12.51	12.50	13.15
2	15.36	8.42	11.55	8.83	9.09	9.38	9.18	9.05
3	29.13	20.56	25.56	20.56	21.33	20.93	20.62	21.20
4	36.91	4.37	16.88	5.86	5.88	5.99	6.71	6.55
5	23.91	7.86	17.49	8.59	8.47	8.30	9.31	9.66
6	35.18	9.78	21.86	11.67	12.30	11.44	11.08	11.85
7	41.59	12.06	24.81	14.83	15.17	14.41	15.26	15.78
8	22.52	7.34	15.85	8.20	9.80	8.46	8.45	9.22
9	43.98	13.94	27.79	16.92	17.76	17.47	17.12	17.96
10	33.50-35.73†	10.48	19.27	11.18	13.05	11.85	12.56	12.76
			Group Size: $N = 10, P = 20$					
1	40.7	9.31	36.73	11.04	14.84	16.65	17.57	17.60
2	15.36	8.42	13.69	9.15	9.75	10.36	10.17	9.54
3	29.13	20.56	28.65	20.56	22.07	21.30	20.67	21.60
4	36.91	4.37	30.48	7.53	7.67	8.01	10.06	9.21
5	23.91	7.86	22.72	8.63	8.80	8.76	10.99	11.38
6	35.18	9.78	33.88	13.68	15.19	13.30	12.52	14.02
7	41.59	12.06	35.47	18.04	18.80	17.03	19.02	19.66
8	22.52	7.34	21.64	9.01	12.52	9.57	9.47	11.09
9	43.98	13.94	35.63	19.50	22.36	21.70	20.12	21.38
10	33.50-35.73†	10.48	27.37	11.93	15.91	13.23	14.73	15.05

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

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 $v^{\mathsf{Q}}_{\mathsf{SGM}}$.

In Table 1, the generic quantile bounds v^{Q} are quite loose for most instances, and the group bounds $v^{\mathsf{Q}}_{\mathsf{SGM}}$ obtained by various heuristic approaches are not significantly improved, especially when the group size is small (i.e., $P = 2$). The bounds $v^{\mathsf{Q}}_{\mathsf{SGM}}$ 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.

Table 2: Average time (in seconds) for computing bounds for Problem (i) instances

v^*	v^{Q}	Group Size	$v^{\mathsf{Q}}_{\mathsf{SGM}}$					
			OG	RG	AG	SG	KG	DG
		$N = 100, P = 2$	34.28	11.69	11.57	12.02	15.92	12.17
3103.72	10.44	$N = 20, P = 10$	229.37	86.03	59.81	66.46	72.01	107.19
		$N = 10, P = 20$	801.56	182.61	117.73	160.02	131.88	292.41

In Table 2, we report the average CPU time for calculating the optimal objective value v^* for the ten instances and the average CPU time for calculating 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, because it 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 $v^{\mathsf{Q}}_{\mathsf{SGM}}$ 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, we need to solve an MILP as discussed in Section 5.1 for Problem (i) instances, and thus computing $v^{\mathsf{Q}}_{\mathsf{SGM}}$ by OG takes much longer time than the other grouping methods, as expected. The average CPU time for optimization-based scenario grouping increases more drastically than the time for computing bounds by other grouping heuristics as P increases; see Table 2.

Next, we consider only solving QGP for a fraction of the total time needed for obtaining an optimal solution (which is reported in Table 2 under Column OG). Then using feasible (not necessarily optimal) grouping solutions obtained at the end of each computational threshold, we test the corresponding quantile bound v^{Q}. Specifically, we set the computational time limit as 10%, 20%, 30%, 40%, and 50% of the total time for optimizing the group subproblem and report the values of the quantile bound $v^{\mathsf{Q}}_{\mathsf{SGM}}$ using the best feasible grouping solutions given by CPLEX in Table 3 under Columns **OG-10%**, **OG-20%**, **OG-30%**, **OG-40%**, and

OG-50%, respectively.

Table 3: Bounds given by different time limits of solving QGP for Problem (i) instances

Inst.	OG	OG-10%	OG-20%	OG-30%	OG-40%	OG-50%
		Group Size: $N = 100, P = 2$				
1	15.73	10.46	10.46	11.99	11.99	11.99
2	9.75	8.42	8.75	8.75	8.94	8.94
3	22.80	20.80	21.23	21.23	21.23	22.80
4	9.34	4.47	5.28	5.28	5.28	6.50
5	11.45	7.93	7.93	9.32	9.32	10.21
6	13.32	9.96	10.36	10.36	10.36	11.50
7	17.36	12.81	13.52	13.52	13.52	14.36
8	10.63	7.89	8.13	8.13	8.86	8.86
9	21.67	15.49	15.49	16.90	16.90	16.90
10	15.25	10.70	11.01	11.01	12.43	12.43
		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
4	16.88	7.07	12.14	12.14	16.88	16.88
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

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% or less of the quantile bounds based on optimal scenario grouping.

Table 4: Root gap closed (in percentage) after adding bounds for Problem (i) instances

Inst.	v^{Q}	$N = 100, P = 2$						$N = 20, P = 10$	$N = 10, P = 20$
		OG	RG	AG	SG	KG	DG	OG	OG
1	0%	12%	0%	0%	0%	0%	1%	34%	89%
2	3%	9%	3%	3%	3%	2%	3%	25%	74%
3	4%	8%	4%	4%	4%	4%	4%	26%	82%
4	0%	11%	0%	0%	0%	0%	0%	21%	67%
5	0%	9%	1%	1%	0%	0%	1%	36%	88%
6	0%	7%	0%	0%	0%	0%	1%	33%	86%
7	1%	7%	1%	1%	1%	1%	1%	29%	66%
8	0%	8%	0%	0%	0%	0%	0%	31%	85%
9	0%	10%	1%	0%	1%	1%	1%	19%	74%
10	1%	8%	1%	1%	1%	1%	1%	17%	54%

To evaluate the strengths of bounds in Table 3, we report in Table 4 how much the root-node gap is closed (in percentage) for each Problem (i) instance after separately adding v^{Q} and each of the $v^{\mathsf{Q}}_{\mathsf{SGM}}$-bounds to the LP relaxation of the MILP reformulation of each chance-constrained program. In Table 4, neither v^{Q} nor $v^{\mathsf{Q}}_{\mathsf{SGM}}$-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 $v^{\mathsf{Q}}_{\mathsf{SGM}}$ given by the optimal grouping close approximately 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 $v^{\mathsf{Q}}_{\mathsf{SGM}}$-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 $v^{\mathsf{Q}}_{\mathsf{SGM}}$ by reporting their gaps from the optimal objective value v^*, measured by $(v^* - \text{LB})/|v^*|$ where LB takes the value of

either v^{Q} or $v^{\mathsf{Q}}_{\mathsf{SGM}}$ in Table 5. We vary the sample size $K = 100, 500, 1000$, and choose group

Table 5: Bound comparison for Problem (ii) *mk-20-10* instances

Inst.	ϵ	K	v^{Q}	$v^{\mathsf{Q}}_{\mathsf{SGM}}$							
				OG	OG-20%	OG-50%	RG	AG	SG	KG	DG
				Group Size: $N = 0.1K$, $P = 10$							
mk-20-10	0.1	100	1.6%	0.3%	0.9%	0.3%	1.1%	1.0%	1.0%	0.9%	1.0%
		500	1.8%	0.4%	1.3%	0.4%	1.5%	1.4%	1.4%	1.2%	1.4%
		1000	2.0%	0.4%	1.7%	0.4%	1.8%	1.8%	1.7%	1.6%	1.8%
	0.2	100	2.3%	0.3%	2.1%	0.4%	2.3%	2.3%	2.2%	2.1%	2.3%
		500	1.5%	0.2%	1.3%	0.2%	1.5%	1.5%	1.5%	1.4%	1.5%
		1000	2.2%	0.2%	1.9%	0.3%	2.2%	2.2%	2.1%	2.0%	2.2%
				Group Size: $N = 0.05K$, $P = 20$							
mk-20-10	0.1	100	1.6%	0.2%	0.8%	0.2%	1.1%	1.1%	1.0%	0.8%	1.0%
		500	1.8%	0.2%	1.2%	0.2%	1.4%	1.4%	1.3%	0.9%	1.3%
		1000	2.0%	0.2%	1.3%	0.3%	1.7%	1.7%	1.6%	1.3%	1.5%
	0.2	100	2.3%	0.2%	1.7%	0.2%	2.1%	2.2%	2.2%	1.8%	2.1%
		500	1.5%	0.1%	1.0%	0.1%	1.5%	1.4%	1.5%	1.1%	1.4%
		1000	2.2%	0.1%	1.7%	0.1%	2.2%	2.0%	2.0%	1.6%	2.0%

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 $v^{\mathsf{Q}}_{\mathsf{SGM}}$ are slightly tighter than v^{Q} for some instances. The optimization grouping bound $v^{\mathsf{Q}}_{\mathsf{SGM}}$ 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.

In Table 6, we compare the average solution time (in seconds) of computing each type of bound for $N = 0.1K$ and $P = 10$. The times for computing $v^{\mathsf{Q}}_{\mathsf{SGM}}$ bounds by using different heuristic approaches are much shorter than the times for computing v^{Q} and optimization grouping based $v^{\mathsf{Q}}_{\mathsf{SGM}}$ bound. Each method's solution time almost linearly increases in K.

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 $v^{\mathsf{Q}}_{\mathsf{SGM}}$ as a lower bound in the scenario decomposition approach, while shrinking the feasible

Table 6: Average time (in seconds) for computing bounds for *mk-20-10* instances with $N = 0.1K$

Inst.	ϵ	K	v^{Q}	$v^{\mathsf{Q}}_{\mathsf{SGM}}$					
				OG	RG	AG	SG	KG	DG
mk-20-10	0.1	100	24.09	25.93	6.38	6.51	6.64	7.72	6.49
		500	143.27	104.17	34.53	34.18	33.49	49.38	33.84
		1000	272.99	196.08	63.69	68.15	64.96	100.63	63.05
	0.2	100	24.01	21.25	6.77	6.91	7.04	8.80	6.57
		500	146.16	189.45	36.28	34.83	36.64	47.16	37.01
		1000	277.48	309.18	63.70	65.61	63.06	97.46	64.34

region $\mathcal{X}$ after evaluating candidate solutions from solving group subproblems, to guarantee the convergence of the algorithm.

We let u and ℓ be the upper and lower bounds of the optimal objective value for a given chance-constrained 0-1 program, which are updated in each iteration of the algorithm. Specifically, u is set as the value of $c^{\mathsf{T}}x$ based on some best found solution x, and ℓ is equal to the quantile bound of SGM, respectively. Until we close the gap between the upper bound u and the lower bound ℓ, we repeat the following procedures: (i) finding a set of scenario groups, (ii) optimizing group subproblems to identify temporary x-solutions and evaluate their corresponding bounds, and (iii) eliminating the 0-1 x-solutions that have already been evaluated via "no-good cuts" included in the feasible region $\mathcal{X}$ in each group subproblem. The algorithm converges in a finite number of steps because we only have a finite number of 0-1 x-solutions to exclude before finding an optimal solution.

We present the details in Algorithm 6. Note that (i) grouping scenarios can be done either by solving the optimization model or by heuristic approaches; (ii) we can perform scenario grouping as Step 2 outside the loop without repeating it in every iteration.

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 (according to Ahmed et al. [2017]). We report the computational results of different procedures in Table 7, where Columns **Non-G**, **KG (ReG)**, **KG (w/o ReG)**, **OG (ReG)**, **OG (w/o ReG)** provide the average solution time (in seconds) for optimizing each set of five replications by using the scenario decomposition without grouping, K-means clustering grouping with or without re-grouping in each iteration, optimization grouping with or without iteratively re-grouping, respectively. (Given the bound comparison results in Table 5, we only run the optimal-grouping model for 50% of the total optimization time and use the produced bounds in the

Algorithm 6 A scenario decomposition algorithm based on scenario groups $G_1, \ldots, G_N$.

repeat
  Group scenarios in $\{1, \ldots, K\}$ into $G_1, \ldots, G_N$. Initialize $\mathcal{X}_{\text{eval}} \leftarrow \emptyset$.
  for $n = 1, \ldots, N$ **do**
    Solve the group subproblem (5) to obtain the optimal objective value ϕ_n and an optimal solution $\hat{x}^n$.
    Update $\mathcal{X}_{\text{eval}} \leftarrow \mathcal{X}_{\text{eval}} \cup \{\hat{x}^n\}$.
  end for
  Update $\ell \leftarrow$ the $(K'+1)$th largest of $\phi_1, \ldots, \phi_N$.
  for $\hat{x} \in \mathcal{X}_{\text{eval}}$ **do**
    if $\hat{x}$ satisfies constraint (1b) **then**
      Set $u \leftarrow \min\{u, c^\top x\}$.
    end if
  end for
  Update $\mathcal{X} \leftarrow \mathcal{X} \backslash \mathcal{X}_{\text{eval}}$ (by including no-good cuts into set $\mathcal{X}$ in each group subproblem).
until $u - \ell \leq \epsilon$

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 parentheses, in Column **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 **Gap** for the last method, which would have "0.0% (5)" for all the rows.

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

Inst.	ϵ	K	Non-G		KG (ReG)		KG (w/o ReG)		OG (ReG)		OG (w/o ReG)
			Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)
mk-20-10	0.1	100	113.78	0.0% (5)	35.18	0.0% (5)	31.55	0.0% (5)	74.91	0.0% (5)	23.64
		500	377.22	0.0% (5)	134.64	0.0% (5)	120.97	0.0% (5)	398.26	0.0% (5)	75.08
		1000	1012.73	0.0% (5)	164.26	0.0% (5)	146.01	0.0% (5)	1006.81	0.0% (5)	62.55
	0.2	100	174.40	0.0% (5)	64.54	0.0% (5)	58.87	0.0% (5)	106.39	0.0% (5)	42.35
		500	367.88	0.0% (5)	173.29	0.0% (5)	158.59	0.0% (5)	307.09	0.0% (5)	78.22
		1000	1315.68	0.0% (5)	374.00	0.0% (5)	334.10	0.0% (5)	1208.77	0.0% (5)	221.61
mk-39-5	0.1	100	LIMIT	3.6% (0)	LIMIT	1.2% (0)	LIMIT	2.1% (0)	LIMIT	0.5% (0)	1793.72
		500	LIMIT	3.9% (0)	LIMIT	1.5% (0)	LIMIT	2.3% (0)	LIMIT	0.8% (0)	2019.24
		1000	LIMIT	4.0% (0)	LIMIT	1.9% (0)	LIMIT	2.2% (0)	LIMIT	0.8% (0)	2205.36
	0.2	100	LIMIT	3.4% (0)	LIMIT	1.7% (0)	LIMIT	2.6% (0)	LIMIT	0.5% (0)	2193.48
		500	LIMIT	3.2% (0)	LIMIT	1.9% (0)	LIMIT	2.7% (0)	LIMIT	0.7% (0)	2213.90
		1000	LIMIT	3.8% (0)	LIMIT	1.8% (0)	LIMIT	3.4% (0)	LIMIT	0.6% (0)	2899.06

In Table 7, we use K-means clustering to represent other heuristic-based grouping meth-

ods because (i) their solution times are similar in Table 6 and (ii) K-means clustering in general leads to tighter $v^{\mathsf{Q}}_{\mathsf{SGM}}$ in Table 5. It takes much less 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 when 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, because each re-grouping step requires solving the optimization-based scenario-grouping problem by the branch-and-cut algorithm, which will slow down the computation significantly.

6 Conclusion

We investigated optimization-driven scenario grouping for strengthening quantile bounds of general chance-constrained programs, and we incorporated the method in 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 we also designed heuristic-grouping approaches. We conducted extensive computational studies 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, which also yield smaller root node gaps. The optimal-grouping bounds become tighter if we increase the group size but take more time to solve. In fact, the time for computing optimal-grouping bounds is much larger than that of solving non-grouping quantile bounds and heuristic-grouping bounds. For most instances, we demonstrated 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.

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; and (iii) 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 grants N00014-14-0315 and N00014-17-1-2296 (Lee), NSF grant CMMI-1433066 and DOE grant DE-SC0018018 (Shen).

References

S. Ahmed. A scenario decomposition algorithm for 0-1 stochastic programs. *Operations Research Letters*, 41(6):565–569, 2013.

S. Ahmed, R. Garcia, N. Kong, L. Ntaimo, G. Parija, F. Qiu, and S. Sen. SIPLIB: a stochastic integer programming test library. `http://www2.isye.gatech.edu/~sahmed/siplib/`, 2016.

S. Ahmed, J. Luedtke, Y. Song, and W. Xie. Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs. *Mathematical Programming*, 162(1-2):51–81, 2017.

D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

C. C. Carøe and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24:37–45, 1999.

R. A. Collado, D. Papp, and A. Ruszczyński. Scenario decomposition of risk-averse multistage stochastic programming problems. *Annals of Operations Research*, 200(1):147–170, 2012.

T. G. Crainic, M. Hewitt, and W. Rei. Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Computers & Operations Research*, 43:90–99, 2014.

Y. Deng and S. Shen. Decomposition algorithms for optimizing multi-server appointment scheduling with chance constraints. *Mathematical Programming*, 157(1):245–276, 2016.

Y. Deng, S. Ahmed, and S. Shen. Parallel scenario decomposition of risk-averse 0-1 stochastic programs. *INFORMS Journal on Computing*, 30(1):90–105, 2017.

Y. Deng, S. Shen, and B. Denton. Chance-constrained surgery planning under conditions of limited and ambiguous data. *INFORMS Journal on Computing*, 31(3):559–575, 2019.

D. Dentcheva and W. Römisch. Duality gaps in nonconvex stochastic optimization. *Mathematical Programming*, 101(3):515–535, 2004.

J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

L. F. Escudero, M. A. Garín, G. Pérez, and A. Unzueta. Scenario cluster decomposition of the lagrangian dual in two-stage stochastic mixed 0–1 optimization. *Computers & Operations Research*, 40(1):362–377, 2013.

D. Gade, G. Hackebeil, S. M. Ryan, J.-P. Watson, R. J.-B. Wets, and D. L. Woodruff. Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. *Mathematical Programming*, 157(1):47–67, 2016.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, 1979.

Y. Jiang, J. Xu, S. Shen, and C. Shi. Production planning problems with joint service-level guarantee: a computational study. *International Journal of Production Research*, 55(1):38–58, 2017.

D. Knuth. Sorting by merging. *The Art of Computer Programming*, 3:158–168, 1998.

S. Küçükyavuz. On mixing sets arising in chance-constrained programming. *Mathematical Programming*, 132(1–2):31–56, 2012.

G. Laporte and F. V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993.

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982.

M. Lu, Z. Chen, and S. Shen. Optimizing the profitability and quality of service in carshare systems under demand uncertainty. *Manufacturing and Service Operations Management*, 20 (2):162–180, 2018.

J. Luedtke. A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Mathematical Programming*, 146(1-2):219–244, 2014.

J. Luedtke and S. Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19(2):674–699, 2008.

J. Luedtke, S. Ahmed, and G. Nemhauser. An integer programming approach for linear programs with probabilistic constraints. *Mathematical Programming*, 122(2):247–272, 2010.

J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I – convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.

N. Miller and A. Ruszczyński. Risk-averse two-stage stochastic linear programming: Modeling and decomposition. *Operations Research*, 59(1):125–132, 2011.

B. Pagnoncelli, S. Ahmed, and A. Shapiro. Sample average approximation method for chance constrained programming: Theory and applications. *Journal of Optimization Theory and Applications*, 142(2):399–416, 2009.

A. Prékopa. On probabilistic constrained programming. In *Proceedings of the Princeton Symposium on Mathematical Programming*, pages 113–138, 1970.

F. Qiu, S. Ahmed, S. S. Dey, and L. A. Wolsey. Covering linear programming with violations. *INFORMS Journal on Computing*, 26(3):531–546, 2014.

R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147, 1991.

K. Ryan, S. Ahmed, S. S. Dey, and D. Rajan. Optimization driven scenario grouping. Available at Optimization-Online `http://www.optimization-online.org/DB_FILE/2016/03/5366.pdf`, 2016.

S. Shen and Z. Chen. Optimization models for differentiating quality of service levels in probabilistic network capacity design problems. *Transportation Research Part B: Methodological*, 58(1):71–91, 2013.

S. Shen, J. C. Smith, and S. Ahmed. Expectation and chance-constrained models and algorithms for insuring critical paths. *Management Science*, 56(10):1794–1814, 2010.

Y. Song, J. R. Luedtke, and S. Küçükyavuz. Chance-constrained binary packing problems. *INFORMS Journal on Computing*, 26(4):735–747, 2014.

J.-P. Watson, R. J. Wets, and D. L. Woodruff. Scalable heuristics for a class of chance-constrained stochastic programs. *INFORMS Journal on Computing*, 22(4):543–554, 2010.

Y. Zhang, S. Shen, B. Li, and J. L. Mathieu. Distributionally robust multi-period optimal power flow with flexible loads. In *Proceedings of IEEE PES PowerTech Manchester 2017*, 2017a.

Y. Zhang, S. Shen, and J. L. Mathieu. Distributionally robust chance-constrained optimal power flow with uncertain renewables and uncertain reserves provided by loads. *IEEE Transactions on Power Systems*, 32(2):1378–1388, 2017b.

Y. Zhang, S. Shen, and S. A. Erdogan. Solving 0-1 semidefinite programs for distributionally robust allocation of surgery blocks. *Optimization Letters*, 12(7):1503–1521, 2018.

Online Supplement of the Paper
"Scenario grouping and decomposition algorithms
for chance-constrained programs"

Yan Deng[1], Huiwen Jia[1], Shabbir Ahmed[2], Jon Lee[1], Siqian Shen[1]

1: Department of Industrial and Operations Engineering,
University of Michigan, Ann Arbor, MI, USA

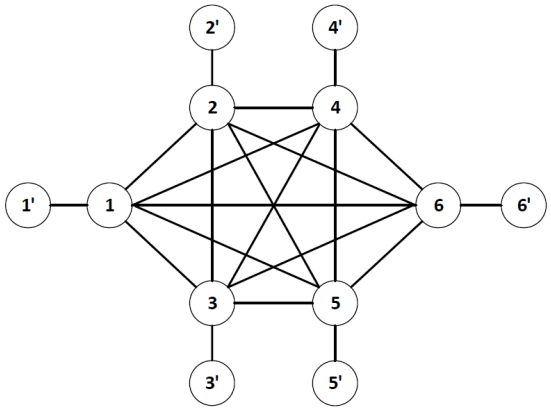
2: School of Industrial and Systems Engineering,
Georgia Institute of Technology, Atlanta, GA, USA

A An Example for the Proof of Proposition 1 for QGP with $P = 2$

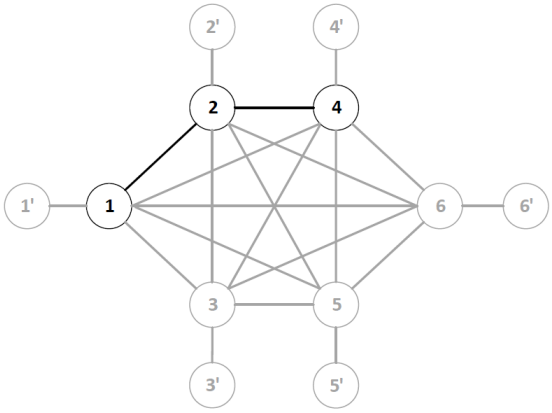
Example 1 *In Figure 1, we depict the graph $G(V, E)$ for a QGP problem instance with $K = 6,\ K' = 1$ and also demonstrate Algorithm 1 for $P = 2$.*

B An Example for the Proof of Theorem 1 for QGP with $P = 3$

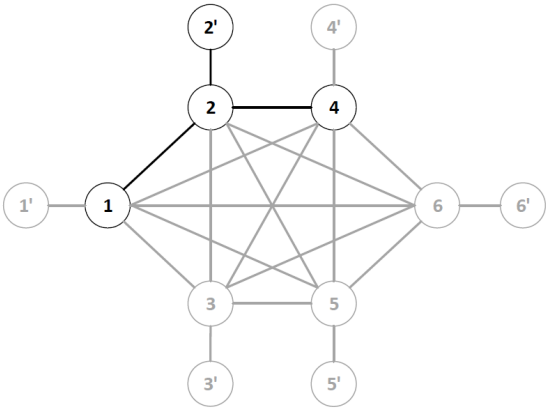
Example 2 *In Figure 2, we provide an example of the graph $G(V, E)$ for a QGP instance with $K = 6,\ K' = 1$ and $P = 3$. The* `Partition into Isomorphic Subgraphs` *with $|V'| = 3$ is also referred to as* `Partition into Triangles`*, which is also demonstrated to be $\mathcal{NP}$-complete in Garey and Johnson [1979].*



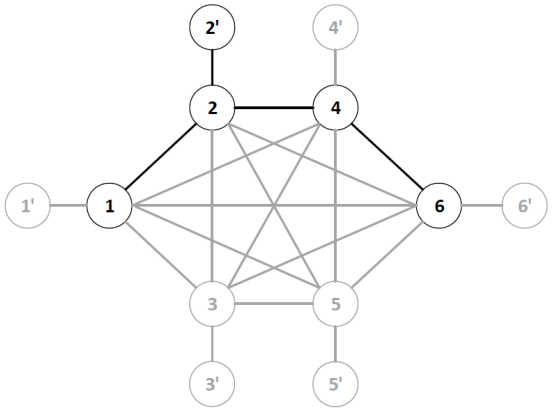
(a) Constructed graph $G(V = V' \cup V'', E)$.



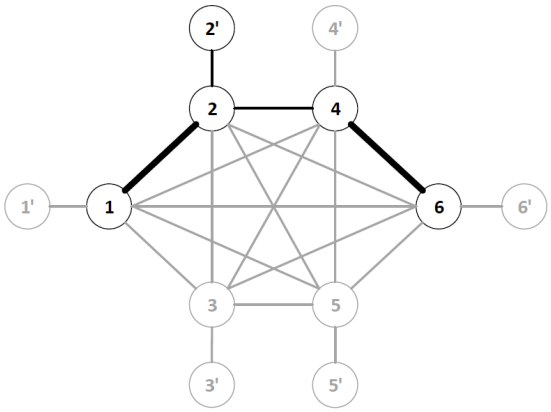
(b) Black part is $\bar{G}(\bar{V}, \bar{E})$ in 1^{st} iteration.



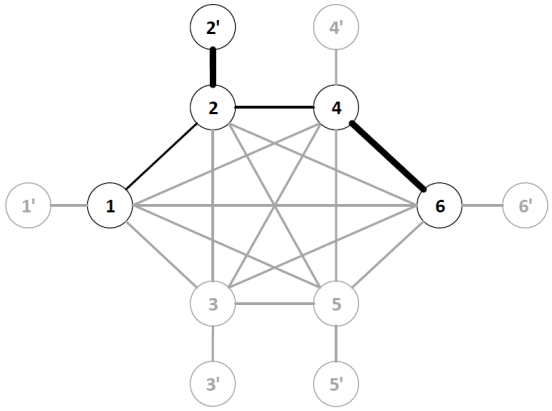
(c) Black part is $\bar{G}(\bar{V}, \bar{E})$ in 2^{nd} iteration.



(d) Black part is $\bar{G}(\bar{V}, \bar{E})$ in 3^{rd} iteration.

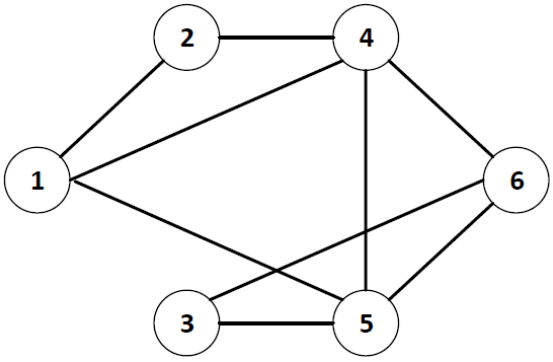


(e) One solution obtained after three iterations are bold lines. We group scenarios 1 and 2, scenarios 4 and 6, and let scenarios 3 and 5 be single-scenario groups, respectively. The optimal objective value equals to the weight of edge $(4, 6)$.

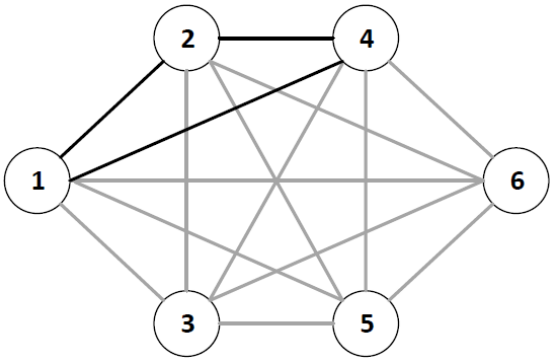


(f) Another solution obtained after three iterations are bold lines. We group scenarios 4 and 6, and let scenarios 1, 2, 3, 5 be single-scenario groups, respectively. The optimal objective value is still the weight of edge $(4, 6)$.

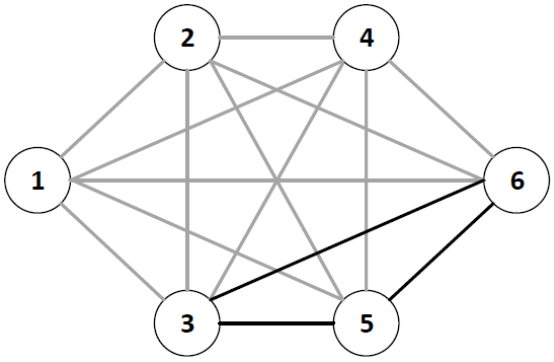
Figure 1: An illustrative example with $K = 6,\ K' = 1,\ P = 2$.



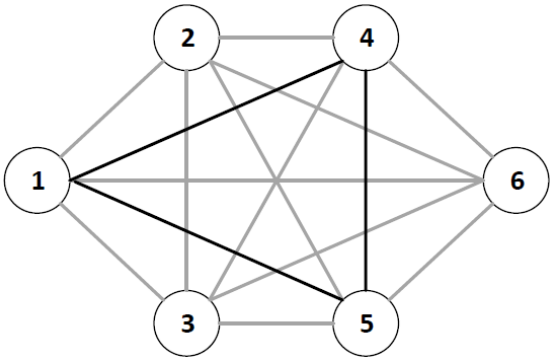
(a) An arbitrary graph $G(V, E)$.



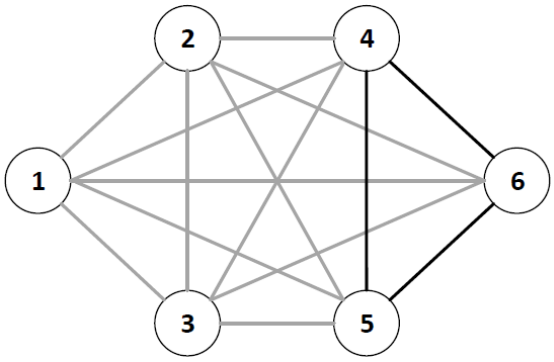
(b) 1^{st} group of three scenarios with value 1.



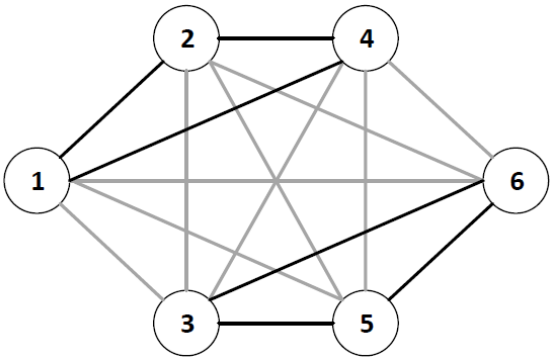
(c) 2^{nd} group of three scenarios with value 1.



(d) 3^{rd} group of three scenarios with value 1.



(e) 4^{th} group of three scenarios with value 1.



(f) Group decisions recovered from the partition into triangles decision. We form two groups and the optimal objective value of QGP is 1.

Figure 2: An illustrative example with $K = 6$, $K' = 1$, $P = 3$ and `Partition into Triangles` problem.