Characterizing Event Constraints with General Disjunctive Programming

Joshua L. Pulsipher [a,1], Daniel Ovalle [a], Hector D. Perez [a], Carl D. Laird [a] and Ignacio E. Grossmann [a]

[a] Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

In this paper, we present a generalized disjunctive programming (GDP) representation of event constraints and leverage GDP solution strategies to solve optimization problems involving this challenging class of constraints. Event constraints are a new modeling paradigm that generalizes joint-chance constraints from stochastic optimization to enforce a constraint on the probability of satisfying a set of constraints aggregated via application-specific logic (constituting an event). This new constraint class offers significant modeling flexibility in posing measured infinite-dimensional optimization constraints (e.g., chance constraints), but can be challenging to reformulate/solve due to difficulties in representing arbitrary logical conditions and specifying a probabilistic measure on a collection of constraints. To address these challenges, we derive a GDP representation of event constrained optimization problems, which readily enables us to pose logical event conditions in a standard form and allows us to draw from a suite of GDP solution strategies to tackle this problem class. We illustrate these findings with a stochastic power grid design case study.

Keywords

Infinite-dimensional optimization, Generalized disjunctive programming, Stochastic programming, Chance constraints.

Problem Definition and Setting

We consider stochastic optimization (SO) problems that are subject to event constraints:

$$\begin{aligned} \min_{z\in\mathcal{Z},q(\xi)\in\mathcal{Q}} \quad & M_\xi f(z,q(\xi),\xi) \\ \text{s.t.} \quad & g_j(z,q(\xi),\xi)\leq 0,\ \ j\in\mathcal{J},\ \xi\in\mathcal{D}_\xi \\ & \mathbb{P}_\xi\left(\Omega(h(z,q(\xi),\xi)\leq 0)\right)\geq\alpha \end{aligned} \tag{1}$$

following the notation introduced in Pulsipher et al. (2022), where the authors established a unifying abstraction for infinite-dimensional optimization (InfiniteOpt) problems and introduced the notion of event constraints. Here, $\xi\in\mathcal{D}_\xi\subseteq\mathbb{R}^{n_\xi}$ are uncertain parameters, $z\in\mathcal{Z}\subseteq\mathbb{R}^{n_z}$ are first-stage variables, $q(\xi)\in\mathcal{Q}\subseteq\mathbb{R}^{n_q}$ are second stage variables, $f(\cdot)$ is the cost function measured by the measure operator M_ξ, and $g_j(\cdot),\ j\in\mathcal{J}$, are constraints strictly enforced over $\mathcal{D}_\xi$ (i.e., almost surely). The event constraint is enforced on a collection of constraints $h_i(\cdot)\leq 0,\ i\in I$, which are combined with the event logic encoded in $\Omega:\{\text{True},\text{False}\}^{|I|}\mapsto\{\text{True},\text{False}\}$; the probability of incurring this event is constrained by the minimum probability level $\alpha\in[0,1]$. We refer the reader to (Pulsipher et al., 2022) for an intuitive introduction on event constraints.

Event constraints are a new constraint class that generalizes classical joint-chance constraints; the later have been applied to a wide variety of problem classes in the literature which include optimal power flow (Baker and Toomey, 2017), model predictive control (Paulson et al., 2020), scheduling (Liu et al., 2020), process design/intensification (Wendt et al., 2002), flexibility/reliability analysis (Pulsipher and Zavala, 2019), and portfolio planning (Pagnoncelli et al., 2009). Joint-chance constraints constrain the event when the intersection (i.e., Ω uses the logical AND operator $\wedge$) of constraints $h(\cdot)\leq 0$ are held and are expressed:

$$\mathbb{P}_\xi\left(\bigwedge_{i\in I} h_i(z,q(\xi),\xi)\leq 0\right)\geq\alpha. \tag{2}$$

We note that the literature typically omits explicitly expressing the operator $\wedge$. This exacts that all the constraints $h(\cdot)\leq 0$ must be enforced jointly for a particular realization of ξ. Such a condition may be overly restrictive in a variety of problems where application-specific logic can be incorporated to enforce a less strict condition on the constraints. For instance, it might be sufficient to satisfy only a certain subset of customer demands in a distribution system. Event constraints provide the modeling flexibility to embed such logic into SO formulations. Following Pulsipher et al. (2022), event constraints can apply to general InfiniteOpt problems, but we restrict the scope of this work to SO for simplicity in presentation.

[1] Corresponding author. Email: pulsipher@cmu.edu.

Event constraints are complex modeling objects that can be challenging to formulate and solve. In the special case of joint-chance constraints, a variety of reformulation/solution techniques have been proposed in the literature. Due to the difficulty in determining the joint probability density function needed for exact analytical reformulations of (2), big-M constraint representations that use binary variables $y(\xi) \in \{0,1\}$ are often used to reformulate (2):

$$\begin{aligned} &h_i(z,q(\xi),\xi) \leq (1-y(\xi))M_i, \quad i \in I, \xi \in \mathcal{D}_\xi \\ &\mathbb{E}_\xi[y(\xi)] \geq \alpha \end{aligned} \tag{3}$$

where $M_i \in \mathbb{R}_+$ is a sufficiently large upper bound that allows relaxing the constraint when $y_i(\xi) = 0$. Moreover, (3) is typically transformed into a finite-dimensional optimization problem via sample average approximation (SAA) using random samples $\{\hat{\xi}_k : k \in \mathcal{K}\}$ (Pagnoncelli et al., 2009):

$$\begin{aligned} &h_i(z,q_k,\hat{\xi}_k) \leq (1-y_k)M_i, \quad i \in I,\ k \in \mathcal{K} \\ &\frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} y_k \geq \alpha. \end{aligned} \tag{4}$$

This SAA approach is simple to implement, but can become intractable for complex systems with a large number of samples. To alleviate this limitation, several iterative cutting-plane solution strategies have been proposed such as branch-and-cut decomposition (Luedtke, 2014) and combinatorial Benders' cuts (Codato and Fischetti, 2006). Moreover, a few extensions of classical disjunctive programming have been made to solve SAA representations of joint-chance constraints (Vielma et al., 2012). Alternative solution techniques include data-driven kernel smoothing, which seeks to estimate the density function of joint-chance constraints (Calfa et al., 2015), and differentiable SAA, which produces a representation that approximates the quantile function (Peña-Ordieres et al., 2020). However, connections to generalized disjunctive programming (GDP) have not yet been explored to our best knowledge.

General event constraints have been solved using big-M constraints and SAA (similar to (4)), where the event logic is encoded via manually derived auxiliary constraints with additional binary variables (Pulsipher et al., 2022). However, such big-M representations are prone to the scalability limitations that are often seen in problems involving joint-chance constraints. Furthermore, as the complexity in the event logic increases, deriving valid auxiliary constraints that encode this logic is non-trivial and error prone if done manually.

To address these issues, we propose a GDP representation of general event constraints. Generalized disjunctive programming provides an intuitive representation for event constraints that provides a straightforward methodology for enforcing arbitrary event logic. A variety of solution techniques have been developed to effectively solve GDP problems that leverage the special structure present in GDP formulations (Grossmann, 2021). Moreover, there are software tools such as `DisjunctiveProgramming.jl` and `Pyomo.GDP` to easily implement this problem class (Chen et al., 2021). The contributions of this work are:

- The expression of event constraints in SO via GDP.
- The application of GDP solution strategies to event/joint-chance constrained problems.
- The use of atleast$(\cdot)$ logic from constraint programming in characterizing events.
- The demonstration of the effect of varied event logic in shaping the Pareto frontier.

Below we provide relevant background on GDP, derive the GDP formulation for event-constrained SO problems, and we present an illustrative case study using a stochastic optimal power system design problem.

Generalized Disjunctive Programming

GDP provides a modeling abstraction that represents mathematical optimization problems via algebraic constraints, disjunctive constraints, and Boolean logic (Raman and Grossmann, 1994; Grossmann and Trespalacios, 2013). Although mathematical programs with discrete decisions are traditionally modelled via mixed-integer programming (MIP), using GDP has several benefits:

- Modeling systems directly via MIP can result in significantly different computational performances depending on the algebraic formulation chosen to represent the same set of constraints involving discrete decisions (Grossmann and Trespalacios, 2013). In contrast, GDP provides a level of abstraction that relies on the underlying logic behind the discrete decisions in the system. The result is a unifying mathematical model that can then be tackled via different solution methods. Rather than being tied to a fixed mathematical representation, as is the case with MIP, the algebraic formulation best suited to the problem can be selected and fine tuned.
- GDP enables more intuitive modeling since discrete decisions (and the relationships between them) are usually better understood in terms of disjunctions and Boolean logic propositions. Using this approach can often aid the modeler in avoiding common modeling mistakes that occur when mixed-integer constraints are used directly to represent logic-based decisions.

General Form

The general GDP formulation is:

$$\begin{aligned} \min_{z,Y} \quad & f(z) \\ \text{s.t.} \quad & g(z) \leq 0 \\ & \bigvee_{i\in I_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(z) \leq 0 \end{bmatrix}, \qquad k \in \mathcal{K} \\ & \Omega(Y) = True \\ & Y_{ik} \in \{True, False\}, \quad i \in I_k,\ k \in \mathcal{K} \\ & z \in \mathcal{Z} \end{aligned} \tag{5}$$

with global continuous constraints $g(z) \leq 0$, Boolean variables Y, and propositional logic $\Omega : \{\text{True},\text{False}\}^{|I_k|\times|\mathcal{K}|} \mapsto \{\text{True},\text{False}\}$ which can encode a variety of logical operators including the NOT ($\neg$), AND ($\wedge$), OR ($\vee$), implication ($\Rightarrow$), and equivalence ($\Longleftrightarrow$) operators. Each disjunction k

is assumed to be proper, meaning that only one disjunct $i \in I_k$ is enforced. The Boolean variable Y_{ik} indicates which set of constraints $h_{ik}(z) \leq 0$ is enforced in each disjunction.

GDP is a generalization of Disjunctive Programming (DP) (Balas, 2018; Hooker, 2009), where DP uses binary variables in place of the Boolean variables in the disjunctions, and 0-1 algebraic constraints in place of the Boolean logic propositions. DP is also restricted to linear constraints. GDP also uses logical propositions which allows intuitive modeling of the relationships between the discrete decisions.

Solution Strategies

The modeling abstraction of GDP makes a suite of solution methods available. One common approach is to perform the direct reformulation of the GDP model to an equivalent MIP model. Here, the logical propositions Ω are converted into Conjunctive Normal Form (CNF), and the resulting CNF propositions are translated into algebraic constraints, where the Boolean variables Y are replaced by binary variables y (see Grossmann and Trespalacios (2013)). In addition, the algebraic constraint $\sum_{i \in I_k} y_{ik} = 1,\ k \in \mathcal{K}$, is added to enforce exclusive nature of the disjunctions. The algebraic constraints inside the disjuncts can then be reformulated with one of the following methods.

- Big-M reformulation (Raman and Grossmann, 1994): Each constraint is enforced or relaxed via a binary variable y_{ik}, which relaxes the constraint by adding a sufficiently large $M \in \mathbb{R}_+$ when $y_{ik} = 0$. This is straightforward, but may result in loose continuous relaxations, which may degrade performance.
- Hull reformulation (Lee and Grossmann, 2000): The formulation is lifted into a higher-dimensional space via variable disaggregation. Its continuous relaxation is obtained by intersecting the convex-hulls of the disjuncts belonging to each disjunction. This approach increases the number of variables and constraints, but can have significantly tighter continuous relaxations relative to big-M.

These reformulation approaches can also be used in conjunction with the following solution methods:

- Hybrid Cutting Plane Methods: The big-M reformulation approach is strengthened via convex-hull cuts (Sawaya and Grossmann, 2005) or basic steps (Trespalacios and Grossmann, 2016).
- Disjunctive Branch-and-bound (DBB) (Lee and Grossmann, 2000): Branching is done on the disjunct whose reformulated binary variable is closest to 1 (the remaining disjuncts in that disjunction are assigned to the complementary branch).
- Logic-based Outer-approximation (LOA) (Türkay and Grossmann, 1996): If the GDP is nonlinear, the problem is decomposed by sequentially solving a reduced NLP and a relaxed MILP problem.
- Relaxation with Integer Cuts (RIC) (Grossmann, 2002): LOA is enhanced with integer cuts to ensure candidate binary solutions y^* are not revisited.

GDP Event Constraint Formulations

In this section, we derive the GDP formulation for general event-constrained SO problems following the form of (1).

Infinite-Dimensional Formulation

To derive the formulation in its natural infinite-dimensional form, we begin by defining Boolean variables $Y_i(\xi) \in \{\text{True}, \text{False}\},\ i \in I$, for each constraint $h_i(\cdot) \leq 0,\ i \in I$ in (1) that indicate when each constraint is satisfied (i.e., if $Y_i(\xi) = \text{True}$, then $h_i(\cdot) \leq 0$). With these we can impose a disjunction at each constraint:

$$\begin{bmatrix} Y_i(\xi) \\ h_i(\xi) \leq 0 \end{bmatrix} \bigvee \begin{bmatrix} \neg Y_i(\xi) \\ h_i(\xi) > 0 \end{bmatrix},\ i \in I,\ \xi \in \mathcal{D}_\xi \tag{6}$$

where we write $h_i(\xi) := h_i(z, q(\xi), \xi)$ for compactness in presentation. Using the property that $\mathbb{P}_\xi(\cdot) = \mathbb{E}[\mathbb{1}(\cdot)]$ (where $\mathbb{1}(\cdot)$ is the indicator function), we can represent the event constraint in (1) using the Boolean variables Y_i:

$$\Omega(Y(\xi)) \iff W(\xi),\ \xi \in \mathcal{D}_\xi \tag{7a}$$

$$\mathbb{E}_\xi[W(\xi)] \geq \alpha \tag{7b}$$

where $W(\xi) \in \{\text{True}, \text{False}\}$ indicates whether an event occurs satisfying the event logic function $\Omega(\cdot)$. Here, the logical constraints encoded in (7a) can be systematically converted into linear inequalities by converting them to CNF following the methodology described in (Raman and Grossmann, 1991).

Substituting (6) and (7) into (1), we obtain the the full GDP formulation for event-constrained SO problems:

$$\begin{aligned} \min\quad & M_\xi f(z, q(\xi), \xi) \\ \text{s.t.}\quad & g_j(z, q(\xi), \xi) \leq 0,\ j \in \mathcal{J},\ \xi \in \mathcal{D}_\xi \\ & \begin{bmatrix} Y_i(\xi) \\ h_i(\xi) \leq 0 \end{bmatrix} \bigvee \begin{bmatrix} \neg Y_i(\xi) \\ h_i(\xi) > 0 \end{bmatrix},\ i \in I,\ \xi \in \mathcal{D}_\xi \\ & \Omega(Y(\xi)) \iff W(\xi),\ \xi \in \mathcal{D}_\xi \\ & \mathbb{E}_\xi[W(\xi)] \geq \alpha \\ & q(\xi) \in \mathcal{Q},\ \xi \in \mathcal{D}_\xi \\ & Y_i(\xi), W(\xi) \in \{\text{True}, \text{False}\},\ i \in I,\ \xi \in \mathcal{D}_\xi \\ & z \in \mathcal{Z}. \end{aligned} \tag{8}$$

In the following section, we describe how (8) can be made finite-dimensional via SAA; however, we note that the GDP reformulations described in this section can be applied directly to (8) before SAA or any other finite transformation technique is applied.

Sample Average Approximation

Commonly, SO problems are transformed into a finite-dimensional form (making it compatible with conventional optimization solvers). We accomplish this via SAA by selecting samples $\{\hat{\xi}_k : k \in \mathcal{K}\}$ that are typically drawn randomly from the distribution of ξ, although more advanced

sampling schemes such as importance sampling can be used (Nemirovski and Shapiro, 2006). Applying SAA to (8):

$$\begin{aligned}
\min \quad & \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} f(z,q_k,\hat{\xi}_k) \\
\text{s.t.} \quad & g_j(z,q_k,\hat{\xi}_k)\le 0,\ \ j\in\mathcal{J},\ k\in\mathcal{K} \\
& \begin{bmatrix} Y_{ik} \\ h_i(\hat{\xi}_k)\le 0 \end{bmatrix} \veebar \begin{bmatrix} \neg Y_{ik} \\ h_i(\hat{\xi}_k) > 0 \end{bmatrix},\ \ i\in I,\ k\in\mathcal{K} \\
& \Omega(Y_k) \iff W_k,\ \ k\in\mathcal{K} \\
& \text{atleast}(\text{ceil}(\alpha|\mathcal{K}|), W) \\
& q_k\in\mathcal{Q},\ \ k\in\mathcal{K} \\
& z\in\mathcal{Z}
\end{aligned} \tag{9}$$

where Y_k is the variable collection $\{Y_{ik}, i\in I\}$. Taking inspiration from constraint programming, we express the expectation using the atleast$(\cdot)$ condition which enforces that at least ceil$(\alpha|\mathcal{K}|)$ of the indicator variables $W_k = \text{True}$. Here, ceil$(\cdot)$ rounds up to the nearest integer. We let $M_\xi = \mathbb{E}_\xi$ for simplicity in presentation, but any risk measure M_ξ can be used following SAA transformations common to SO (Ruszczyński and Shapiro, 2006).

We also note that alternative transformations (e.g., polynomial chaos expansion) can be applied to (8) to obtain a tractable finite-dimensional formulation as discussed in (Pulsipher et al., 2022).

Case Study

In this section, we study the design of the IEEE-14 power distribution network with stochastic demand proposed by (Dabbagchi, 1962) by adapting the formulation given in (Pulsipher and Zavala, 2019). The goal is to minimize the cost of adding generator/line capacity to the grid subject to an event constraint over the capacity constraints (i.e., the event describes how the capacity limits are respected in response to random demand). The objective is posed with the added generator capacities $z_{g,i}\in\mathcal{Z}_g$, $i\in I$, and line capacities $z_{l,\ell}\in\mathcal{Z}_l$, $\ell\in\mathcal{L}$,:

$$\min \sum_{i\in I} c_{g,i} z_{g,i} + \sum_{\ell\in\mathcal{L}} c_{l,\ell} z_{l,\ell} \tag{10}$$

with unit costs $c_{g,i}\in\mathbb{R}_+$ and $c_{l,\ell}\in\mathbb{R}_+$. This is subject to a flow balance at each node $n\in\mathcal{N}$ with line flows $q_l(\xi)$, generation $q_g(\xi)$, and uncertain demand $\xi\sim\mathcal{N}(\mu,\Sigma)$ (using the same values of μ and Σ presented in (Pulsipher and Zavala, 2019)):

$$\sum_{\ell\in\mathcal{L}_n^{\text{in}}} q_{l,\ell}(\xi) - \sum_{\ell\in\mathcal{L}_n^{\text{out}}} q_{l,\ell}(\xi) + \sum_{i\in I_n} q_{g,i}(\xi) - \sum_{r\in\mathcal{R}_n} \xi_r = 0 \tag{11}$$

where I_n is the set of generators at node n, $\mathcal{R}_n$ is the set of demands at node n, and $\mathcal{L}_n^{\text{in}}$,$\mathcal{L}_n^{\text{out}}$ denote the set of lines that flow toward and away from a node n, respectively. We express the disjunction that arises from satisfying the capacity limit at each generator $i\in I$ for a given demand ξ:

$$\begin{bmatrix} Y_{g,i}(\xi) \\ q_{g,i}(\xi)\le \bar{q}_{g,i} + z_{g,i} \end{bmatrix} \veebar \begin{bmatrix} \neg Y_{g,i}(\xi) \\ q_{g,i}(\xi) > \bar{q}_{g,i} + z_{g,i} \end{bmatrix} \tag{12}$$

where $\bar{q}_{g,i}$ are the existing generator capacities. Similarly, we enforce a disjunction at each line $\ell\in\mathcal{L}$ under a specific demand ξ:

$$\begin{bmatrix} Y^L_{l,\ell}(\xi) \\ -\bar{q}_{l,\ell} - z_{l,\ell} > q_{l,\ell}(\xi) \end{bmatrix} \veebar \begin{bmatrix} Y_{f,\ell}(\xi) \\ -\bar{q}_{l,\ell} - z_{l,\ell} \le q_{l,\ell}(\xi) \\ q_{l,\ell}(\xi) \le \bar{q}_{l,\ell} + z_{l,\ell} \end{bmatrix} \veebar \begin{bmatrix} Y^U_{l,\ell}(\xi) \\ q_{l,\ell}(\xi) > \bar{q}_{l,\ell} + z_{l,\ell} \end{bmatrix} \tag{13}$$

where $\bar{q}_{l,\ell}$ are the existing line capacities. Here, either the capacity limit of the line is met $Y_{l,\ell}$ or one of the lower/upper limits is violated (indicated by $Y^L_{l,\ell}$ and $Y^U_{l,\ell}$, respectively). Finally, we enforce an event constraint logic over the collection of capacity constraints, using a Boolean variable $W(\xi)$ that accounts for the result of the event encoded in $\Omega(\cdot)$ (which will be varied following Eqns. (15) and (16) below):

$$\begin{aligned}
& W(\xi) \iff \Omega(Y_g(\xi), Y_l(\xi)),\ \ \xi\in\mathcal{D}_\xi \\
& \mathbb{E}_\xi[W(\xi)] \ge \alpha,\ \ \xi\in\mathcal{D}_\xi.
\end{aligned} \tag{14}$$

Equations (10)-(14) are combined to yield an infinite-dimensional event-constrained formulation following (8). We apply SAA following our previous discussion using 1,000 Monte Carlo samples of ξ where each element of each realization is truncated at 0 such that no negative demands are incurred. The source files are available at `https://github.com/infiniteopt/source-code/tree/main/gdp_event_constrs`.

Pareto Frontier under varied Event Logic

We explore how different event logic $\Omega(\cdot)$ impacts the Pareto frontier of this design problem. In particular, we consider intersection logic (i.e., a joint-chance constraint as shown in (2)):

$$\Omega_\wedge(Y_g, Y_l) := \left(\bigwedge_{i\in I} Y_{g,i}\right) \wedge \left(\bigwedge_{\ell\in\mathcal{L}} Y_{l,\ell}\right) \tag{15}$$

in juxtaposition to using atleast$(\cdot)$ logic to relax the number line/generator limits that have to be respected:

$$\begin{aligned}
\Omega_{\text{atleast}}(Y_g, Y_l; Y_{g,\min}, Y_{l,\min}) :=& \text{atleast}(Y_{g,\min}, Y_g) \wedge \\
& \text{atleast}(Y_{l,\min}, Y_l)
\end{aligned} \tag{16}$$

where $Y_{g,\min}\in\{1,\dots,5\}$, $Y_{l,\min}\in\{1,\dots,20\}$ are the minimum number generator and line capacity limits enforced for a particular value of ξ, respectively. We solve the problem using a range of α values and several choices of $Y_{g,\min}$ and $Y_{l,\min}$. In each case, the corresponding GDP model is solved using the big-M reformulation in `Pyomo.GDP` with `Gurobi v9.5.1` as the solver on a Linux machine with 8 Intel® Xeon® Gold 6234 CPUs running at 3.30 GHz with 128 total hardware threads and 1 TB of RAM running Ubuntu.

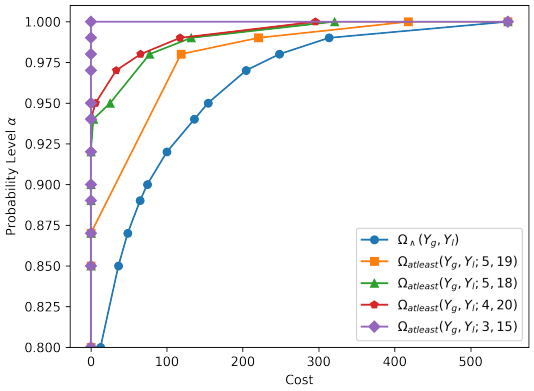

Figure 1: Pareto frontiers obtained using $\Omega_\wedge(\cdot)$ and $\Omega_{atleast}(\cdot)$ using varied values of $Y_{g,min}$ and $Y_{l,min}$.

Figure 1 shows the optimal Pareto frontiers corresponding to each choice of $\Omega(\cdot)$. As we would expect, the intersection logic $\Omega_\wedge(\cdot)$ incurs the greatest costs since it strictly enforces that every capacity constraint be satisfied for each instance of ξ. Hence, we have to add more capacity to the design relative to the frontiers derived from $\Omega_{\text{atleast}}(\cdot)$ for a fixed probability level α. We observe how decreasing the values of $Y_{g,\text{min}}$ and $Y_{l,\text{min}}$ decreases the costs in engineering sufficient capacity for feasible operation. For this application, this means that if only a subset of line/generator capacity constraints need to be respected for a particular demand profile $\hat{\xi}$, then we can embed that logic into our problem formulation and obtain a lower cost design relative to using traditional joint-chance constraint formulations. Hence, we can use application specific logic to avoid over-engineering in design problems.

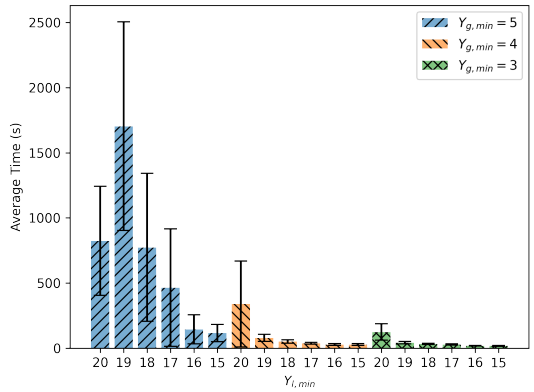

Figure 2: The mean computational time ($\pm 1\sigma$) required to solve each Pareto pair for varied $Y_{g,min}$ and $Y_{l,min}$.

Figure 2 shows how the different event logic affects the solution time of the problem. In particular, we average over the time required to solve each Pareto pair when computing the Pareto frontier with a specified $Y_{g,\text{min}}$ and $Y_{l,\text{min}}$ using the big-M reformulation method. Note that the event with $Y_{g,\text{min}} = 5$ and $Y_{l,\text{min}} = 20$ denotes traditional intersection logic $\Omega_\wedge(\cdot)$ that arises from joint-chance constraints. Interestingly, all but one instance (i.e., $Y_{g,\text{min}} = 5$ and $Y_{l,\text{min}} = 19$) exhibit reduced computational times relative to the joint-chance baseline. This highlights that events with more complex logic aggregation do not necessarily incur increased computational cost, and in fact, can lead to reduced computational cost in certain cases.

Solution Method Comparison

Finally, we compare the performance of the different GDP transformation methods in this case study. In particular, we use the big-M, cutting-plane, and hull transformation methods and use classical joint-chance constraint logic (i.e., $\Omega_\wedge(\cdot)$). Figure 3 shows the performance plot indicating the fraction of Pareto pair instances that each method solved as a function of wall-time. Generally, the cutting plane method proposed in (Trespalacios and Grossmann, 2016) required the least amount of time to solve every Pareto pair. This can likely be attributed to the cutting plane formulation having a tighter relaxation relative to the big-M approach. The hull reformulation exhibits the worst performance, likely due to the computational burden resulting from the increased formulation size, despite being the tightest formulation. We leave the comparison of other GDP solution techniques to future work.

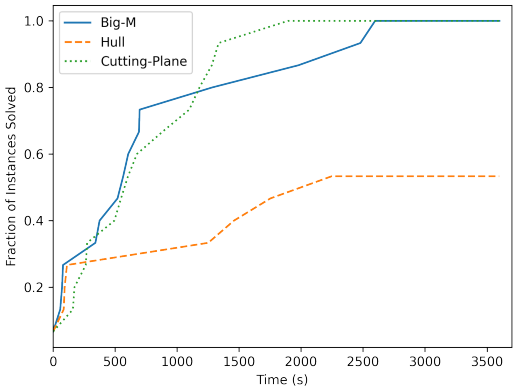

Figure 3: Performance of various GDP transformation methods in solving Pareto pairs with $\Omega_\wedge(\cdot)$.

Conclusions and Future Work

In this work, we present a GDP formulation for event-constrained SO problems, which capture conventional joint-chance constraints as a special case. GDP enables introducing arbitrary event logic, which can be systematically reformulated into linear algebraic constraints using CNF (automated in software such as `DisjunctiveProgramming.jl` and `Pyomo.GDP`). Using GDP also opens up a suite of solution methods, of which the cutting plane method was found to provide some computational advantages in the case study shown relative to classical big-M approaches.

Future work will include extending the GDP formulation to general InfiniteOpt problems (e.g., dynamic optimization). Extending some of the solution strategies developed in the joint-chance constraint literature (e.g., branch-and-cut decomposition and data-driven kernel smoothing) to event constrained problems is also interesting direction of future research. Finally, this new class of GDP problems inspires the development of new solution strategies that are better suited for problems with a large number of disjunctions.

Acknowledgements

We thank the Center for Advanced Process Decision-making for supporting this work.

References

Baker, K. and B. Toomey (2017). Efficient relaxations for joint chance constrained ac optimal power flow. *Electric Power Systems Research 148*, 230–236.

Balas, E. (2018). *Disjunctive programming*. Springer.

Calfa, B. A., I. E. Grossmann, A. Agarwal, S. J. Bury, and J. M. Wassick (2015). Data-driven individual and joint chance-constrained optimization via kernel smoothing. *Computers & Chemical Engineering 78*, 51–69.

Chen, Q., E. S. Johnson, D. E. Bernal, R. Valentin, S. Kale, J. Bates, J. D. Siirola, and I. E. Grossmann (2021). Pyomo.gdp: an ecosystem for logic based modeling and optimization development. *Optimization and Engineering*, 1–36.

Codato, G. and M. Fischetti (2006). Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research 54*(4), 756–766.

Dabbagchi, I. (1962). Ieee 14 bus power flow test case. *American Electric Power System, Golden CO*.

Grossmann, I. E. (2002). Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and engineering 3*(3), 227–252.

Grossmann, I. E. (2021). *Advanced optimization for process systems engineering*. Cambridge University Press.

Grossmann, I. E. and F. Trespalacios (2013). Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming. *AIChE Journal 59*(9), 3276–3295.

Hooker, J. N. (2009). A principled approach to mixed integer/linear problem formulation. In *Operations research and cyber-infrastructure*, pp. 79–100. Springer.

Lee, S. and I. E. Grossmann (2000). New algorithms for nonlinear generalized disjunctive programming. *Computers & Chemical Engineering 24*(9-10), 2125–2141.

Liu, B., Q. Zhang, X. Ge, and Z. Yuan (2020). Cvar-based approximations of wasserstein distributionally robust chance constraints with application to process scheduling. *Industrial & Engineering Chemistry Research 59*(20), 9562–9574.

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

Nemirovski, A. and A. Shapiro (2006). Scenario approximations of chance constraints. In *Probabilistic and randomized methods for design under uncertainty*, pp. 3–47. Springer.

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

Paulson, J. A., E. A. Buehler, R. D. Braatz, and A. Mesbah (2020). Stochastic model predictive control with joint chance constraints. *International Journal of Control 93*(1), 126–139.

Peña-Ordieres, A., J. R. Luedtke, and A. Wachter (2020). Solving chance-constrained problems via a smooth sample-based nonlinear approximation. *SIAM Journal on Optimization 30*(3), 2221–2250.

Pulsipher, J. L. and V. M. Zavala (2019). A scalable stochastic programming approach for the design of flexible systems. *Computers & Chemical Engineering 128*, 69–76.

Pulsipher, J. L., W. Zhang, T. J. Hongisto, and V. M. Zavala (2022). A unifying modeling abstraction for infinite-dimensional optimization. *Computers & Chemical Engineering 156*, 107567.

Raman, R. and I. E. Grossmann (1991). Relation between milp modelling and logical inference for chemical process synthesis. *Computers & Chemical Engineering 15*(2), 73–84.

Raman, R. and I. E. Grossmann (1994). Modelling and computational techniques for logic based integer programming. *Computers & Chemical Engineering 18*(7), 563–578.

Ruszczyński, A. and A. Shapiro (2006). Optimization of risk measures. In *Probabilistic and randomized methods for design under uncertainty*, pp. 119–157. Springer.

Sawaya, N. W. and I. E. Grossmann (2005). A cutting plane method for solving linear generalized disjunctive programming problems. *Computers & chemical engineering 29*(9), 1891–1913.

Trespalacios, F. and I. E. Grossmann (2016). Cutting plane algorithm for convex generalized disjunctive programs. *INFORMS Journal on Computing 28*(2), 209–222.

Türkay, M. and I. E. Grossmann (1996). Logic-based minlp algorithms for the optimal synthesis of process networks. *Computers & Chemical Engineering 20*(8), 959–978.

Vielma, J. P., S. Ahmed, and G. L. Nemhauser (2012). Mixed integer linear programming formulations for probabilistic constraints. *Operations Research Letters 40*(3), 153–158.

Wendt, M., P. Li, and G. Wozny (2002). Nonlinear chance-constrained process optimization under uncertainty. *Industrial & engineering chemistry research 41*(15), 3621–3629.