Chance Constrained Optimization: Bi-level reformulation

Master Research Project

Anoush Azar-Pey supervised by Saeed Masiha
INDY Lab, EPFL

June 24, 2025

Table of contents

Chance Constrained Optimization

Chance Constrained Optimization (CCO) [KHVR20]: A way of modeling uncertainty and controlling rare events in optimization.

Definition (Chance Constrained Optimization Problem)

$$\min_{x \in \mathcal{X}} f(x) \text{ s.t. } \mathbb{P}\{g(x, Z) \leq 0\} \geq 1 - \delta \tag{1}$$

where $\mathcal{X}$ is a closed convex subset of $\mathbb{R}^d$, $Z \sim P_Z$ is a scalar random variable, $f(x)$ is our *objective function*, $g(x, Z)$ our *chance function* and $1 - \delta$ our safety probability level, typically close to 1.

We assume f and g to be convex functions w.r.t. x [ROC70].

Objectives

In this project, we aim to design two simple algorithms for *chance constrained optimization* and compare them with prior work [LMA21; Zha+24].

Main Steps:

1. Reformulate the problem as a *bi-level* optimization by introducing a scalar auxiliary variable.
2. Relax the bi-level problem into a single-level problem using the *augmented Lagrangian method*.
3. Develop two methods: a *first-order* and a *zeroth-order* algorithm.
4. Conduct numerical experiments and compare with benchmarks from [LMA21; Zha+24].

Quantile definition

First, we reformulate the chance constraint using the notion of $(1-\delta)$-quantile.

Definition (Quantile)

Let Y be a scalar random variable, $F_Y(t)$ its Cumulative Distribution Function and $\delta \in [0,1]$ our level. Y's $(1-\delta)$-quantile [Hab96] denoted $Q_{1-\delta}(Y)$ is defined as:

$$Q_{1-\delta}(Y) = \inf\{t \in \mathbb{R} : F_Y(t) \geq 1-\delta\}. \tag{2}$$

Intuitively, it can be interpreted as the inverse of the CDF[1] of a random variable, i.e. $F_Y^{-1}(1-\delta)$.

[1] CDF stands for Cumulative Distribution Function, which describes the probability that a random variable takes a value less than or equal to a given point.

Quantile reformulation

We can rewrite our previously defined chance constraint using quantiles [LMA21]:

$$\mathbb{P}\{g(x,Z) \le 0\} \ge 1-\delta \iff Q_{1-\delta}(g(x,Z)) \le 0. \tag{3}$$

Intuitively, this means that the left $1-\delta$ proportion of the mass of $g(x,Z)$'s density should lie below 0. Using this equivalence, it follows that our main (Problem 1) can be rewritten as *Bi-level problem*:

$$\begin{cases} \min\limits_{x\in\mathcal{X}} & f(x) \\ \text{s.t.} & \mathbb{P}\{g(x,Z) \le 0\} \ge 1-\delta \end{cases} \longleftrightarrow \begin{cases} \min\limits_{x\in\mathcal{X}} & f(x) \\ \text{s.t.} & s \le 0 \\ & s = Q_{1-\delta}(g(x,Z)) \end{cases}. \tag{4}$$

Challenge $s = Q_{1-\delta}(g(x,Z))$ is a non-convex non-smooth constraint. To overcome it, we use the following notion.

Superquantile definition

Definition (Superquantile)

Let Y be a random variable, $\delta \in [0, 1]$ our level and $Q_{1-\delta}(Y)$ our quantile at the level $1 - \delta$. Y's $(1 - \delta)$-superquantile, denoted $SQ_{1-\delta}(Y)$ is defined as:

$$SQ_{1-\delta}(Y) = \mathbb{E}\{Y \mid Y \geq Q_{1-\delta}(Y)\} \tag{5}$$

which can be equivalently defined as (from [LMA21])

$$SQ_{1-\delta}(Y) = \frac{1}{\delta} \int_{1-\delta}^{1} Q_p(Y) dp. \tag{6}$$

Superquantiles can be intuitively interpreted as the average mass of high values of a density function.

Variational formulation for Superquantile

For a given random variable Y and a level δ, we will estimate these two quantities with the following convex minimization problem:

Proposition

[LMA21]

$$SQ_{1-\delta}(Y) = \min_{s\in\mathbb{R}} s + \frac{1}{\delta}\mathbb{E}[\max\{Y - s, 0\}]. \tag{7}$$

Moreover, the left endpoint of the solution set of (7) is $Q_{1-\delta}(Y)$.

Bi-level reformulation

Using (Proposition 1), we introduce the following *jointly convex* function $G_\delta(x, s)$, involving the auxiliary variable s:

$$G_\delta(x, s) = s + \frac{1}{\delta}\mathbb{E}_{P_Z}[\max\{g(x, Z) - s, 0\}] \tag{8}$$

and further rewrite our problem as:

$$\begin{cases} \min\limits_{x \in \mathcal{X}} & f(x) \\ \text{ s.t.} & s \leq 0 \\ & s = Q_{1-\delta}(g(x, Z)) \end{cases} \longleftrightarrow \begin{cases} \min\limits_{x \in \mathcal{X}} & f(x) \\ \text{ s.t.} & s \leq 0 \\ & s \in S(x) = \arg\min\limits_{s \in \mathbb{R}} G_\delta(x, s) \end{cases} . \tag{9}$$

Penalty method

We will assume that $S(x) = \arg\min_{s\in\mathbb{R}} G_\delta(x,s)$ is a singleton, i.e. it contains a unique solution, that we will denote $s^*(x)$, leading to the following reformulation of the CCO:

$$\min_{x\in\mathcal{X}} f(x) \text{ s.t. } s^*(x) \leq 0. \tag{10}$$

Penalty method: Reformulate (10) by its mini-max reformulation:

$$\min_{x\in\mathcal{X}} \max_{\lambda\geq 0} f(x) + \lambda s^*(x). \tag{11}$$

However, $\max_{\lambda\geq 0} f(x) + \lambda s^*(x)$ is non-smooth w.r.t. x.

Augmented Lagrangian method

To address this, we use the augmented Lagrangian:

$$\min_{x \in \mathcal{X}} \max_{\lambda \geq 0} P(x, \lambda) = \min_{x \in \mathcal{X}} \max_{\lambda \geq 0} \left[f(x) + \lambda s^*(x) - \frac{\mu}{2} \lambda^2 \right]. \tag{12}$$

Here, the added μ-regularization term $-\frac{\mu}{2}\lambda^2$ serves to soften the effect of the λ-penalized constraint $\lambda s^*(x)$.
By solving maximization over λ, rewrite (Problem 10) as the simpler:

$$\min_{x \in \mathcal{X}} F(x) = \min_{x \in \mathcal{X}} f(x) + \frac{s^*(x)}{2} \left[\frac{s^*(x)}{\mu} \right]_+ \tag{13}$$

with $F(x)$ being our *objective function*, $s^*(x)$ denoting $\arg\min_{s \in \mathbb{R}} G_\delta(x, s)$, $[\cdot]_+$ the $\max(\cdot, 0)$ or $\mathrm{ReLU}(\cdot)$ function.

Algorithms

To minimize $F(x) = f(x) + \frac{s^*(x)}{2}\left[\frac{s^*(x)}{\mu}\right]_+$, *first-order methods* are a natural choice.

Problem:

- $s^*(x)$ is not necessarily differentiable w.r.t. x.

Solutions:

1. Approximate $\max(x, 0)$ with a C^2 function $h_\theta(x)$ in the definition of $G_\delta(x, s)$,
   $\Rightarrow s^*_\theta(x) := \arg\min_s s + \delta^{-1}\mathbb{E}[h_\theta(g(x, Z) - s)]$ is differentiable.
   We approximate F by the C^1 function $\tilde{F}(x) := f(x) + \frac{s^*_\theta(x)}{2}\left[\frac{s^*_\theta(x)}{\mu}\right]_+$.
2. *Zeroth-order methods*, which do not require any derivations and computing values of $F(x) = f(x) + \frac{s^*(x)}{2}\left[\frac{s^*(x)}{\mu}\right]_+$ suffices.

Outline

Twice continuously differentiable $h_\theta(x)$ function

We introduce a C^2 function that will replace the $\max(x, 0)$ function inside $G_\delta(x, s)$:

$$h_\theta(x) = -\frac{x^4}{16\theta^3} + \frac{3x^2}{8\theta} + \frac{x}{2} + \frac{3\theta}{16}. \tag{14}$$

This function is almost identical to $\max(x, 0)$ except for the interval $x \in [-\theta, \theta]$ where it is smooth.
Our new $G_{\delta,\theta}(x, s)$ function is now defined as:

$$G_{\delta,\theta}(x, s) := s + \frac{1}{\delta}\mathbb{E}_{P_Z}[h_\theta(g(x, Z) - s)] \tag{15}$$

and is also twice differentiable w.r.t. to x.
Let's now introduce our first algorithm.

Algorithms: First-order method

To minimize $\tilde{F}$, we will first use the *gradient descent algorithm*.
Need to compute the partial derivatives $\frac{\partial}{\partial x}\tilde{F}(x)$ and $\frac{\partial}{\partial x}s_\theta^*(x)$.
We express and compute

- $\frac{\partial}{\partial x}\tilde{F}(x)$ as a combination of $\nabla f(x)$ and $\frac{\partial}{\partial x}s_\theta^*(x)$
- $\frac{\partial}{\partial x}s_\theta^*(x)$ as a combination of $\frac{\partial^2}{\partial x \partial s}G_{\delta,\theta}(x, s_\theta^*(x))$ and $\left[\frac{\partial^2}{\partial s^2}G_{\delta,\theta}(x, s_\theta^*(x))\right]^{-1}$

Algorithms: First-order method

We define our first algorithm, that we will refer to as `Algorithm 1`:

Algorithm Our first-order algorithm

$x \leftarrow x_0$
$i \leftarrow 0$
while $i <$ # iters **do**
  Sample from the Z distribution a fixed number of times
  Estimate $s_\theta^*(x)$ by minimizing $G_{\delta,\theta}(x, s)$ over s using GD with $\frac{\partial}{\partial s} G_{\delta,\theta}(x, s)$
  Update x using GD with $\frac{\partial}{\partial x}\tilde{F}(x)$, i.e. $x \leftarrow x - \gamma \frac{\partial}{\partial x}\tilde{F}(x)$
  $i \leftarrow i + 1$
end while

Outline

Algorithms: Zeroth-order method

Zeroth-order methods [NS17] estimate the first-order gradient of a function via the following procedure and subsequent gradient descent step:

$$
\begin{aligned}
&\text{Sample } u \text{ from } \{u : \|u\|_2 = 1\} \\
&\hat{\nabla}_k h(x) := \frac{h(x + uk) - h(x - uk)}{2k} \cdot u \qquad (16) \\
&x_{t+1} \leftarrow x_t - \eta \hat{\nabla}_k h(x_t).
\end{aligned}
$$

Algorithms: Zeroth-order method

Let's recall the definition of F:

$$F(x) = f(x) + s^*(x) \left[\frac{s^*(x)}{\mu} \right]_+ . \tag{17}$$

In our case we estimate the gradient of our function F:

$$\hat{\partial}_k F(x) := \frac{F(x+uk) - F(x-uk)}{2k} \cdot u. \tag{18}$$

We pick the *learning rate* η as our k parameter. We compute $s^*(x \pm up)$ the same way we compute $s^*(x)$, i.e. using *Subgradient Descent* (SD) with $\frac{\partial}{\partial s} G_\delta(x, s)$.

We define our second algorithm, that we will refer to as `Algorithm 2`:

Algorithm Our zeroth-order algorithm (2-point estimator)

$x \leftarrow x_0$
$i \leftarrow 0$
while $i < \#$ iters **do**
  Sample from the Z distribution a fixed number of times
  Estimate $s^*(x)$ by minimizing $G_\delta(x, s)$ over s using SD with $\frac{\partial}{\partial s} G_\delta(x, s)$
  Sample u, the direction of the perturbation for the zeroth-order method
  Estimate $s^*(x+uk)$ and $s^*(x-uk)$ by minimizing $G_\delta(x \pm uk, s)$ over s using SD with $\frac{\partial}{\partial s} G_\delta(x \pm uk, s)$
  Approximate the gradient of F with $\hat{\partial}_k F(x.)$
  Update x using GD with $\hat{\partial}_k F(x)$, i.e. $x \leftarrow x - \gamma \hat{\partial}_k F(x)$
  $i \leftarrow i + 1$
end while

Numerical Experiments

We will now introduce 4 examples to assess the performance of our two simple algorithms, Algorithm 1 (first-order) and Algorithm 2 (zeroth-order) in comparison with several known algorithms such as:

- The *Bundle algorithm* from [LMA21].
- 3 algorithms from [Zha+24]: Conformal Predictive Programming - Mixed Integer Programming (CPP-MIP), Conformal Predictive Programming - Karush Kuhn Tucker (CPP-KKT) and the Sample Average Approximation (SAA).

Numerical Experiments

We recall the definition of CCO problems:

$$\min_{x \in \mathcal{X}} f(x) \text{ s.t. } \mathbb{P}\{g(x, Z) \le 0\} \ge 1 - \delta. \tag{19}$$

For each of the following problem, we define:

- $f(x)$ our *objective function*
- $g(x, Z)$ our *chance function*
- the distribution P_Z of the scalar random variable Z
- $1 - \delta$ our safety probability level, typically close to 1

Example 1

We present Example 1, a simple uni-variate problem, We pick

$$f(x) = (x-2)^2 \tag{20}$$

as our convex objective function, and

$$g(x, Z) = xZ - 1, \quad Z \sim \mathcal{N}(1, 1). \tag{21}$$

We know that the global minimum of f is $x = 2$, but we pick our safety probability level $1 - \delta = 0.95$ so that this point is infeasible in CCO.

Example 1

To determine whether a solution is *optimal* we use the sub-optimality [LMA21] metric:

$$\frac{|f(x_k) - f(x^*)|}{|f(x^*)|}. \tag{22}$$

To determine whether a given point is feasible, we use the *Empirical coverage* metric [Zha+24]:

$$\mathrm{EC}(x) = \mathbb{E}_Z\left[\mathbf{1}\{g(x, Z) \le 0\}\right]. \tag{23}$$

Example 1

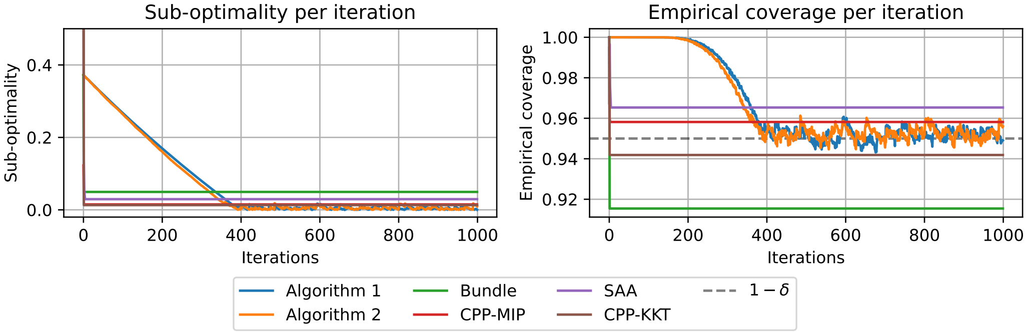

Figure: Convergence of all algorithms

Example 1

Algorithm	Sub-optimality	Empirical Coverage
Algorithm 1	0.0012	0.9494
Algorithm 2	0.0133	0.9575
Bundle	0.0495	0.9154
CPP-KKT	0.013	0.9428
CPP-MIP	0.0149	0.9590
SAA	0.0294	0.9657

Table: Algorithm performance: average of the last iterations

- Algorithms 2 and 1 are the best performing
- `Bundle algorithm`'s solution is infeasible, since its empirical coverage is below our $1-\delta$ safety probability treshold and the solutions of `CPP-MIP`, `SAA`, `CPP-KKT` have a higher sub-optimality

Example 2

We present Example 2, a more complex bi-variate problem from [LMA21]. The objective function

$$f(x) = \frac{1}{2}(x-a)^T Q(x-a) \text{ with } a = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, Q = \begin{pmatrix} 5.5 & 4.5 \\ 4.5 & 5.5 \end{pmatrix} \tag{24}$$

is quadratic with $Q \succeq 0$ and thus convex [ROC70]. The chance function is

$$\begin{aligned} g(x,Z) = Z^T W(x) Z + w^T Z \quad \text{with } W(x) &= \begin{pmatrix} x_1^2 + 0.5 & 0 \\ 0 & |x_2 - 1|^3 + 0.2 \end{pmatrix} \\ w &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ Z \sim \mathcal{N}(\mu, \Sigma) \quad \text{with } \mu &= \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \Sigma = \begin{pmatrix} 20 & 0 \\ 0 & 20 \end{pmatrix}. \end{aligned} \tag{25}$$

As in the paper, we choose an unusually low safety probability level $1 - \delta \approx 0.033$.

Example 2

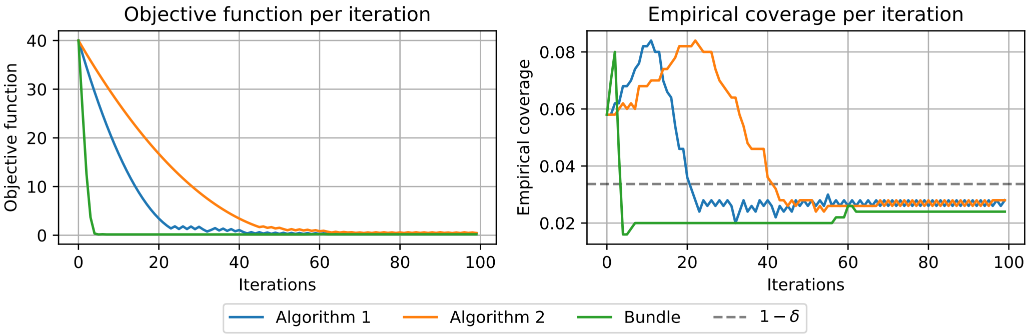

Figure: Convergence of Algorithms 1 and 2, and the `Bundle algorithm`

Example 2

Algorithm	**Objective function**	**Empirical coverage**
Algorithm 1	0.3273	0.0270
Algorithm 2	0.5245	0.0274
Bundle	0.1872	0.0240

Table: Algorithm performance: average of the last iterations

- The optimization trajectories and final points are similar for all three algorithms
- Algorithm 1 and the `Bundle algorithm` have close final points while Algorithms 1 and 2 have the same trajectory
- It is important to note that none of the solutions satisfy the feasibility conditions

Example 2.2

We slightly change the parameters of the original problem to further test and compare the different algorithms. The objective function is now

$$f(x) = \frac{1}{2}(x-a)^T Q(x-a) \text{ with } a = \begin{pmatrix} -2 \\ -3 \end{pmatrix}, Q = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix} \tag{26}$$

and is still quadratic with $Q \succeq 0$. The chance function is

$$\begin{aligned} g(x, Z) = Z^T W(x) Z + w^T Z \quad \text{with } W(x) &= \begin{pmatrix} (x_1-2)^2+1 & 0 \\ 0 & |x_2+1|^3 - 0.4 \end{pmatrix} \\ w &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ Z \sim \mathcal{N}(\mu, \Sigma) \quad \text{with } \mu &= \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \Sigma = \begin{pmatrix} 20 & 0 \\ 0 & 20 \end{pmatrix}. \end{aligned} \tag{27}$$

$1 - \delta \approx 0.033$ remains unchanged.

Example 2.2

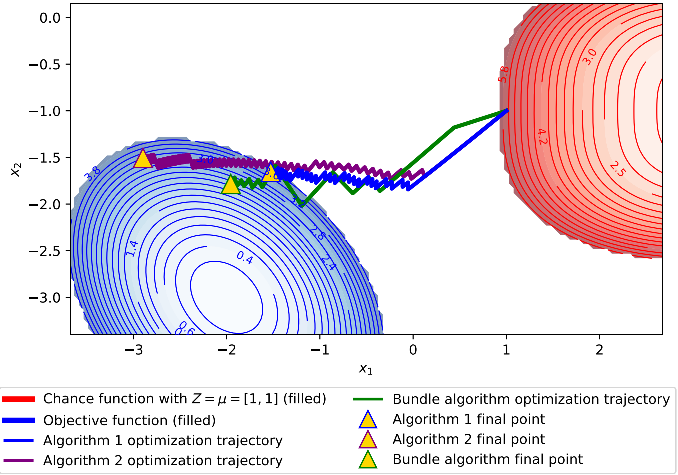

Figure: Optimization trajectories of Algorithms 1 and 2, and the `Bundle algorithm`, starting from $[1, -1]$

Example 2.2

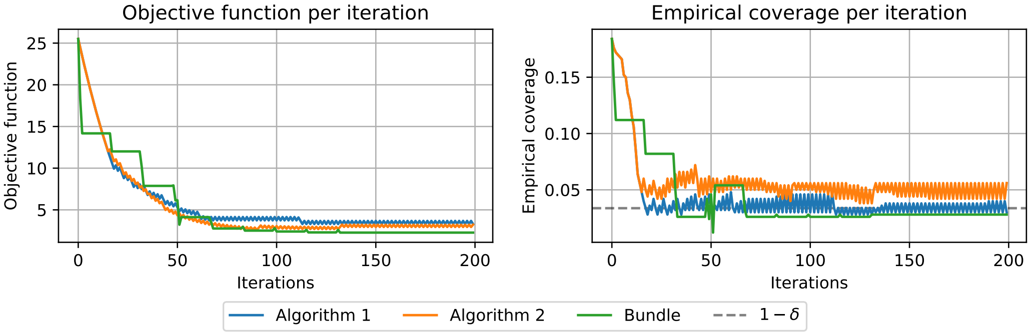

Figure: Convergence of Algorithms 1 and 2, and the `Bundle algorithm`

Example 2.2

Algorithm	Objective function	Empirical coverage
Algorithm 1	3.4888	0.0350
Algorithm 2	3.0806	0.0490
Bundle	2.2702	0.0280

Table: Algorithm performance: average of the last iterations

- We notice that the optimization trajectories are initially similar but differ in their final point
- The trajectory of Algorithm 2 stands out the most from the other two
- This time, Algorithms 1 and 2 yield feasible solutions whereas the *Bundle algorithm*'s solution lies just below the $1 - \delta$ threshold

Example 3

We present Example 3, a uni-variate problem from [Zha+24], with a high-variance scalar random variable Z. The objective function and chance constraint are

$$f(x) = x^3 e^x \qquad g(x, Z) = 50Ze^x - 5 \quad \text{with } Z \sim \operatorname{Exp}\left(\frac{1}{20}\right). \tag{28}$$

The problem also has a deterministic constraint: $x^3 + 20 \leq 0 \iff x \leq (-20)^{1/3}$, which we verify is satisfied but do not take into account in our gradients.
For this problem, we have $1 - \delta = 0.9$.

Example 3

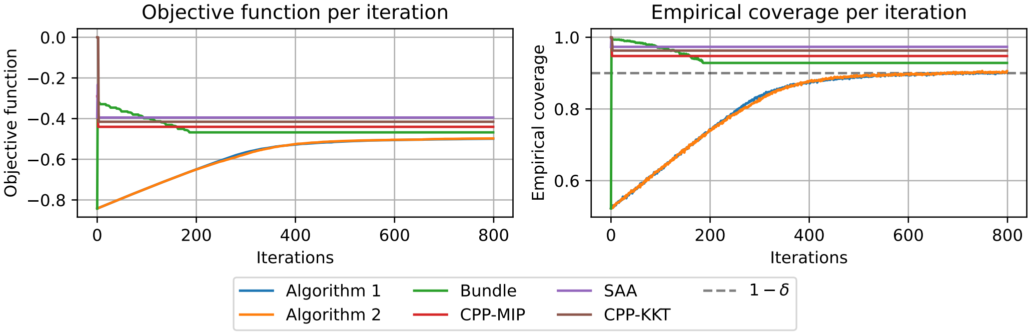

Figure: Convergence of all algorithms

Example 3

Algorithm	Objective function	Empirical coverage
Algorithm 1	-0.499	0.9020
Algorithm 2	-0.4978	0.9030
Bundle	-0.4677	0.9284
CPP-KKT	-0.4155	0.9624
CPP-MIP	-0.4404	0.9477
SAA	-0.3947	0.9733

Table: Algorithm performance: average of the last iterations

- Algorithms 1 and 2, and the `Bundle algorithm`, while converging in many more steps, outperform all algorithms from [Zha+24]
- All solutions are feasible

Example 4

We present Example 4, a tri-variate resource-allocation problem from [Zha+24]. The objective function and chance constraint are

$$f(x) = c \cdot x \qquad \mathbf{g}(x, Z) = Z - Ax \quad \text{with } Z \sim \text{lognormal}\left(0, \frac{1}{4}\right) \times 3 \in \mathbb{R}^3$$
$$c = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \quad A = \begin{pmatrix} 3 & 12 & 2 \\ 10 & 3 & 5 \\ 5 & 3 & 15 \end{pmatrix}. \tag{29}$$

We notice that unlike previous problems, our chance function is no longer a scalar but a vector of size s. We call them *Joint Chance Constrained Optimization* (JCCO) problems and can define them as

$$\min_{x \in \mathcal{X}} f(x) \text{ s.t. } \mathbb{P}\{g_i(x, Z) \leq 0, \forall i \in \{1, \dots, s\}\} \geq 1 - \delta. \tag{30}$$

Example 4

How to reduce JCCO to the generic CCO? *Pointwise Maximum*.
Intuitively, we only keep the largest element from the vector $\mathbf{g}(x, Z)$ to satisfy the constraint. It leads to this equivalent reformulation of the JCCO problem:

$$\min_{x \in \mathcal{X}} f(x) \text{ s.t. } \mathbb{P}\left\{ \max_{i \in \{1,\ldots,s\}} g_i(x, Z) \leq 0 \right\} \geq 1 - \delta. \tag{31}$$

For this problem, we have $1 - \delta = 0.9$.

Example 4

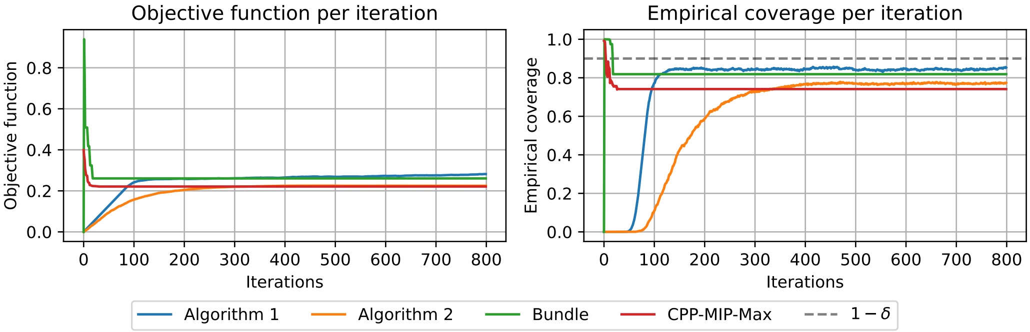

Figure: Convergence of Algorithms 1, 2, the `Bundle algorithm` and `CPP-MIP` using the `Pointwise Maximum` method

Example 4

Algorithm	Objective function	Empirical coverage
Algorithm 1	0.2814	0.8513
Algorithm 2	0.2249	0.7719
Bundle	0.2606	0.8187
CPP-MIP-Max	0.221	0.7414

Table: Algorithm performance: average of the last iterations

- Again, Algorithms 1, 2 and the `Bundle algorithm` require more iterations to converge but output competitive solutions.
- None of the solutions is feasible but Algorithms 1 and the `Bundle algorithm`'s solutions have the highest Empirical coverage.

Conclusion

- Methodology:
  - Bi-level reformulation of chance-constrained optimization
  - Relaxation to unconstrained form
  - Developed two solution methods:
    - First-order method
    - Zeroth-order method
- Experimental evaluation:
  - Tested on 4 distinct problems
  - Included both uni-dimensional and multi-dimensional cases
- Key findings:
  - Both methods exhibit similar solution quality
  - Consistently produce nearly feasible solutions
  - Solutions competitive across different scenarios
- Zeroth-order method tradeoffs:
  - Higher computational cost
  - Particularly valuable for non-differentiable objectives

References I

[Hab96] Shelby J. Haberman. "Quantiles". In: *Advanced Statistics: Description of Populations*. New York, NY: Springer New York, 1996, pp. 367–404. ISBN: 978-1-4757-4417-0. DOI: 10.1007/978-1-4757-4417-0_7. URL: https://doi.org/10.1007/978-1-4757-4417-0_7.

[KHVR20] Willem K. Klein Haneveld, Maarten H. van der Vlerk, and Ward Romeijnders. "Chance Constraints". In: *Stochastic Programming: Modeling Decision Problems Under Uncertainty*. Cham: Springer International Publishing, 2020, pp. 115–138. ISBN: 978-3-030-29219-5. DOI: 10.1007/978-3-030-29219-5_5. URL: https://doi.org/10.1007/978-3-030-29219-5_5.

[LMA21] Yassine Laguel, Jérôme Malick, and Wim Ackooij. *Chance constrained problems: a bilevel convex optimization perspective*. 2021. arXiv: 2103.10832 [math.OC]. URL: https://arxiv.org/abs/2103.10832.

References II

[NS17] Yurii Nesterov and Vladimir Spokoiny. "Random Gradient-Free Minimization of Convex Functions". In: *Foundations of Computational Mathematics* 17.2 (2017), pp. 527–566. ISSN: 1615-3383. DOI: 10.1007/s10208-015-9296-2. URL: https://doi.org/10.1007/s10208-015-9296-2.

[ROC70] R. TYRRELL ROCKAFELLAR. "Convex Functions". In: *Convex Analysis*. Princeton University Press, 1970, pp. 23–31. ISBN: 9780691015866. URL: http://www.jstor.org/stable/j.ctt14bs1ff.8 (visited on 04/12/2025).

[Zha+20] Jingzhao Zhang et al. *Why gradient clipping accelerates training: A theoretical justification for adaptivity*. 2020. arXiv: 1905.11881 [math.OC]. URL: https://arxiv.org/abs/1905.11881.

References III

[Zha+24] Yiqi Zhao et al. *Conformal Predictive Programming for Chance Constrained Optimization*. 2024. arXiv: 2402.07407 [eess.SY]. URL: https://arxiv.org/abs/2402.07407.

Appendix

Algorithms: First-order method

To minimize F, we will first use a *first-order method*, the *gradient descent algorithm*. We therefore need to compute the partial derivatives $\frac{\partial F(x,s^*(x))}{\partial x}$ and $\frac{\partial}{\partial x}s^*(x)$.

$$\begin{aligned}\frac{\partial F(x,s^*(x))}{\partial x} &= \nabla f(x) + \frac{\partial s^*(x)}{\partial x}\left[\frac{s^*(x)}{\mu}\right]_+ + s^*(x)\begin{cases}\frac{\partial s^*(x)}{\mu} & \text{if } s^*(x) > 0\\ 0 & \text{otherwise}\end{cases}\\ &= \nabla f(x) + 2\frac{\partial s^*(x)}{\partial x}\left[\frac{s^*(x)}{\mu}\right]_+ .\end{aligned} \tag{32}$$

Algorithms: First-order method

We do not have a closed-form formula for $s^*(x)$ because it results from a minimization problem. We know that $s^*(x)$ is a minimizer of $G_{\delta,\theta}(x, s)$, therefore we have:

$$\begin{aligned}\frac{\partial G_{\delta,\theta}(x, s^*(x))}{\partial s} = 0 \implies & \frac{d}{dx}\left[\frac{\partial G_{\delta,\theta}(x, s^*(x))}{\partial s}\right] = 0 \\ \iff & \frac{\partial^2}{\partial x \partial s} G_{\delta,\theta}(x, s^*(x)) + \frac{\partial}{\partial x} s^*(x) \frac{\partial^2}{\partial s^2} G_{\delta,\theta}(x, s^*(x)) = 0 \\ \iff & \frac{\partial}{\partial x} s^*(x) = -\frac{\partial^2}{\partial x \partial s} G_{\delta,\theta}(x, s^*(x)) \left[\frac{\partial^2}{\partial s^2} G_{\delta,\theta}(x, s^*(x))\right]^{-1}.\end{aligned} \tag{33}$$

We know that $\frac{\partial^2}{\partial s^2} G_{\delta,\theta}(x, s^*(x))$ is *invertible* because $s^*(x)$ is unique.

Algorithms: Zeroth-order

- 2-point estimator performance:
  - Effective for uni-variate problems (Section 4)
  - Reason: Matches the search space's uni-dimensionality
- Limitations in higher dimensions:
  - Performance becomes suboptimal
  - Reason: Infinite-dimensional space makes directional exploration insufficient
- Proposed enhancements (Algorithm 2):
  - *Uni-variate problems:* Maintain standard 2-point estimator
  - *Multi-dimensional problems:* Introduce refined **4-point estimator** to address limitations

Algorithms: Zeroth-order (4-point estimator)

- 4-point estimator for bi-dimensional problems:
  - Uses orthogonal perturbations to span full 2D space
  - Ensures robust gradient estimation
- Algorithm steps:
  1. Sample random perturbation δ_1 uniformly on unit circle
  2. Generate orthogonal perturbation δ_2 $(\delta_1 \perp \delta_2)$
  3. Evaluate objective function at four points: $x \pm \delta_1$ and $x \pm \delta_2$
- Gradient calculation:
  - Compute directional gradients independently using finite differences:
    - For δ_1 direction: $\nabla_1 \approx f(x+\delta_1) - f(x-\delta_1)$
    - For δ_2 direction: $\nabla_2 \approx f(x+\delta_2) - f(x-\delta_2)$
  - Final estimate: $\nabla = \frac{1}{2}(\nabla_1 + \nabla_2)$
- Key advantages:
  - Captures curvature information in all directions
  - Eliminates directional bias of 2-point estimator
  - Provides accurate approximation in $\mathbb{R}^2$

Algorithms: Zeroth-order (4-point estimator)

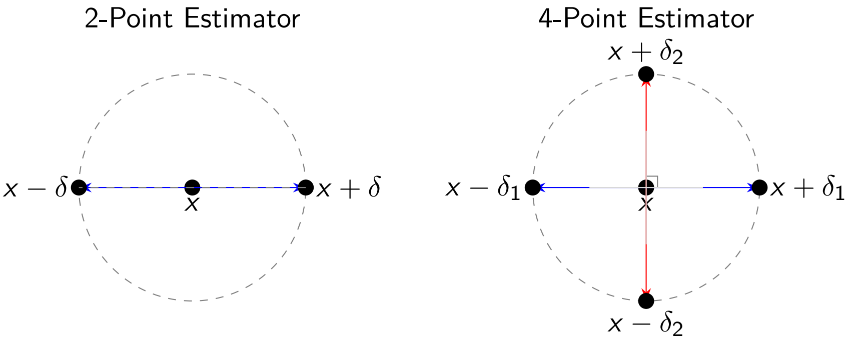

Figure: Gradient estimation methods in 2D space: Left: 2-point estimator with symmetric perturbations $\pm\delta$ in a single direction, spanning only a line. Right: 4-point estimator with orthogonal perturbations $\pm\delta_1$ and $\pm\delta_2$, enabling full 2D span through linear combinations.

Algorithms: Zeroth-order (Stochastic perturbations)

- Challenge in zeroth-order optimization:
  - Premature convergence to suboptimal solutions
  - Root cause: Inappropriate perturbation magnitudes
  - Fixed scales may be:
    - Too small: Unable to escape local optima
    - Too large: Cause noisy updates
- Proposed solution:
  - Stochastic scaling factor α applied at each iteration
  - Multiplies perturbation vector: $\alpha \cdot \delta$
  - Dynamically adjusts perturbation amplitude
  - Enables adaptive exploration

Algorithms: Zeroth-order (Stochastic perturbations)

- Implementation:
  - Sample $\alpha \sim \mathcal{U}\left(\frac{1}{a}, a\right)$ each iteration [2]
  - Example parameter: $a = 3/2$ (balanced range)
  - Maintains computational efficiency
- Key benefits:
  - Balances exploration (larger α) and exploitation (smaller α)
  - Adapts to landscape without manual tuning
  - Mitigates premature convergence

[2] $\mathcal{U}$ denotes uniform distribution

Algorithms: Zeroth-order (Update clipping)

- Problem: Unstable zeroth-order updates
  - Caused by high variance in: Objective function $f(x)$, Chance function $g(x, Z)$
  - Particularly severe when Z exhibits significant variability
  - Consequence: Updates $\|\Delta x\|$ may be very large, causing numerical instability
- Solution: Update clipping [Zha+20]
  - Enforce bound: $\|\Delta x\| \leq C$
  - Implementation:
    - Compute update Δx
    - If $\|\Delta x\| > C$, project onto ℓ_2-ball: $\Delta x \leftarrow C \cdot \frac{\Delta x}{\|\Delta x\|}$
- Key benefits:
  - Prevents extreme parameter shifts
  - Maintains numerical stability
  - Robust to high-variance scenarios

Example 1 (Optimal point analytical derivation)

We can derive the analytical optimal point for this problem by computing $\{x : xZ - 1 \leq 0\}$ and picking x^* from this set that minimizes f:

$$\begin{gathered}
\mathbb{P}(xZ - 1 \leq 0) \geq 1 - \delta \iff \mathbb{P}\left(Z \leq \frac{1}{x}\right) = \Phi\left(\frac{1}{x} - 1\right) \geq 1 - \delta \\
x \leq \frac{1}{\Phi^{-1}(1-\delta) + 1} \quad \text{where } \Phi \text{ is the CDF of } \mathcal{N}(0,1)
\end{gathered} \tag{34}$$

With $\delta = 0.05$, we have $x \leq 0.378092$. We know that f is minimized at $x = 2$, therefore we have to pick the highest value that satisfies the constraint. Hence, we have $x^* = \{\Phi^{-1}(1-\delta) + 1\}^{-1}$.

Example 2

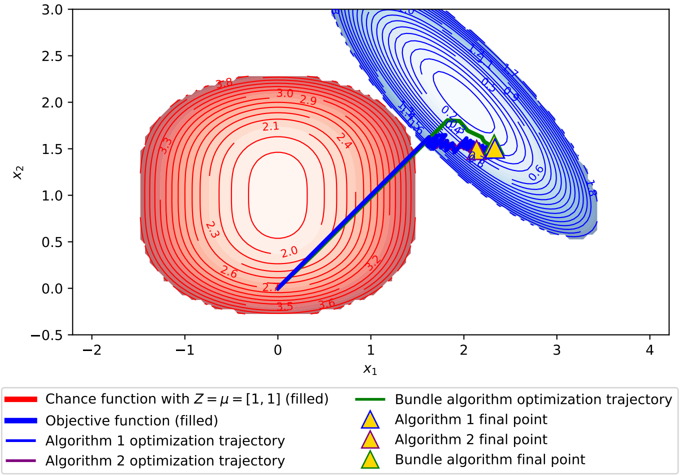

Figure: Optimization trajectories of Algorithms 1 and 2, and the `Bundle algorithm`, starting from $[0, 0]$