Parallel Sampled Convex Problems

David Eckman (dje88) and Calvin Wylie (cjw278)

November 3, 2015

1 Introduction

Uncertain convex programming aims to optimize a convex function subject to constrains that are not precisely known. Without loss of generality, these problems can be written as

$$\begin{aligned} \underset{x \in X}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & f(x, \delta) \leq 0, \end{aligned} \tag{UCP}$$

where $\delta \in \Delta$ is a (possibly random) *uncertain* parameter.

The *robust* formulation of UCP seeks to satisfy the constraints for all possible values of δ. That is,

$$\begin{aligned} \underset{x \in X}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & f(x, \delta) \leq 0 \quad \forall\, \delta \in \Delta. \end{aligned} \tag{RCP}$$

For general uncertainty sets Δ, RCP can become intractable, or produce solutions that are in a sense too "conservative".

If we assume that δ is distributed on Δ according to some probability measure $\mathbb{P}$, instead of seeking a solution that satisfies the constraints for all possible values of δ, we may seek a solution that is feasible with high probability. Specifically, the *chance-constrained* program for acceptable risk of constraint violation ϵ is

$$\begin{aligned} \underset{x \in X}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & \mathbb{P}\left[f(x, \delta) > 0\right] \leq \epsilon. \end{aligned} \tag{CCP$_\epsilon$}$$

Chance-constrained programs are, in general, non-convex, so we often settle for convex approximations.

2 Sampled Convex Programs

Assuming that we are able to independently sample δ from Δ according to $\mathbb{P}$, one method to obtain a solution that is feasible for CCP$_\epsilon$ with high probability is through *sampled* or *random* convex programs.

Let $\delta_1, \ldots, \delta_N$ be iid samples of our unknown parameter δ. The associated sampled convex program is

$$\begin{aligned} &\underset{x \in X}{\text{minimize}} && c^T x \\ &\text{subject to} && f(x, \delta_i) \le 0,\ i = 1, \ldots, N. \end{aligned} \tag{SCP}$$

The sampled convex program is attractive because it does not require any distributional information, and preserves the complexity class of the original uncertain convex program UCP. Moreover, given some N, we can exactly quantify with what probability a solution to the sampled program will be feasible for the chance-constrained program (see [1] and [2]). However, to achieve a high probability, N may need to be quite large, limiting the computational tractability for some classes of problems.

This motivates our primary research question: can we solve in parallel multiple sampled convex programs, each with a smaller number of constraints, and somehow recover a solution that is still feasible for the chance constrained program with high probability?

3 Parallel Sampled Convex Programming

Motivated by the fact that a convex program in n dimensional space will have at most n support constraints at the optimal solution, the problem of finding an optimal solution to SCP can be viewed as the problem of identifying the n supporting constraints. Can we build a consensus on what the supporting constraints are by looking at solutions of smaller sampled convex programs?

For an example, suppose we determine that N samples are needed to reach our desirable CCP_ϵ feasibility probability, and suppose that we have available one "master" processor and p "slave" processors available. For simplicity, suppose that $N = mp$ for some positive integers $m > n$. Consider the following procedure:

1. For each processor $1 \le j \le p$, solve the sampled convex program

$$\begin{aligned} &\underset{x \in \mathcal{X}}{\text{minimize}} && c^T x \\ &\text{subject to} && f(x, \delta_i) \le 0 \quad i = m(j-1)+1, \ldots, mj. \end{aligned}$$

   Let x_j^* be the corresponding solution.

2. For each slave processor $1 \le j \le p$, determine the supporting constraints of the optimal solution x_j^*. Assuming that each sampled convex program is fully-supported with probability one, label the sample corresponding to these constraints $\delta_1^j, \ldots, \delta_n^j$. Send these samples to the master processor.

3. On the master processor, solve the sampled convex program

$$\begin{aligned} &\underset{x \in \mathcal{X}}{\text{minimize}} && c^T x \\ &\text{subject to} && f(x, \delta_i^j) \le 0 \quad i = 1, \ldots, k \text{ and } j = 1, \ldots, p. \end{aligned}$$

   Let x^{**} be the corresponding solution.

A recent paper by Carlone et al. [3] analyzes a similar algorithm, where instead of a master processor, each slave processor passes their corresponding supporting constraints around in a ring topology. They prove that (almost surely) a finite number of iterations is necessary to achieve CCP$_\epsilon$ feasibility with high probability, but do not quantify the number required.

4 Experiments

We intend to conduct experiments on the performance of the parallel procedure versus the traditional serial RCP procedure on some suitably chosen CCP$_\epsilon$ optimization problems. For measures of performance of the solutions x^* and x^{**}, we will look at the objective function values, the expected violation probabilities, and the probabilities that the violations probabilities are greater than ϵ. For estimating the last two of these performance measures, we will use Monte Carlo simulation and test over problems for which one can explicitly characterize the feasible region of CCP$_\epsilon$.

We will also experiment with variations of the basic procedure. One variation would involve communicating the support constraints of the master's problem back to the workers and doing new sampling before solving new subproblems for each workers. This could be done over a number of iterations and we could study the performance of the returned solutions at each iteration.

Another variation would again involve communicating support constraints of the master's problem back to the workers, but instead of doing new sampling, we re-solve the subproblems with the additional shared support constraints. After sending the support constraints of these new subproblems to the master, we could perform one final solve and get a solution that is no worse than the previous. This is because passing support constraints from the master to the workers would allow for constraints that were not previously support constraints to become support constraints.

We will set up strong and weak scaling experiments to compare the quality of solutions and wall-clock time to solution against the number of processors used. We will also attempt to model the performance and communication costs of the algorithm implementations.

References

[1] G. Calafiore and M. C. Campi. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102(1):25–46, 2005.

[2] M. C. Campi and S. Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230, 2008.

[3] L. Carlone, V. Srivastava, F. Bullo, and G. C. Calafiore. Distributed random convex programming via constraints consensus. *SIAM Journal on Control and Optimization*, 52(1):629–662, 2014.