Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

A Sequential Algorithm for Sampled Mixed-integer Optimization Problems ⋆

Mohammadreza Chamanbaz * **Roland Bouffanais** *

* *Singapore University of Technology and Design, Singapore 487372, e-mail: (*{chamanbaz, bouffanais}@sutd.edu.sg*)*

Abstract: In this paper, we propose a computationally efficient algorithm for solving mixed-integer sampled optimization problems involving a large number of constraints. The proposed algorithm has a sequential nature. Specifically, at each iteration of the algorithm, the feasibility of a candidate solution is verified for all the constraints involved in the sampled optimization problem and violating constraints are identified. As a second step, an optimization problem is formed whose constraint set involves the current basis—the minimal set of constraints defining the current candidate solution—and a limited number of the observed violating constraints. We prove that the algorithm converges to the optimal solution in finite time. Additionally, we establish the effectiveness of the proposed algorithm using mixed-integer linear, and quadratically constrained quadratic programming problems.

Keywords: Sampled optimization problems, Sequential algorithm, Large-scale optimization.

1. INTRODUCTION

Sampled optimization problems are optimization problems involving a possibly large number of sampled constraints. This class of problems is widely used in the scenario approach (Calafiore and Campi, 2004; Calafiore et al., 2012; Campi and Garatti, 2018a) as well as more general learning paradigms such as statistical learning theory (Vidyasagar, 2001, 2002; Alamo et al., 2009). In these lines of research, a semi-infinite optimization problem—a robust optimization problem involving an infinite number of constraints—is approximated by a sampled optimization problem involving a finite number of random constraints. The number of random samples is then selected such that the solution of the sampled optimization problem attains some desired probabilistic robustness. If the probabilistic parameters defining the robustness of the solution of the sampled optimization problem are stringent, then the sample complexity (the number of samples required to guarantee certain probabilistic robustness guarantee) may be large thereby leading to a computationally complex sampled optimization problem.

There have been few attempts in the literature to reduce the computational complexity associated with the solution of such sampled optimization problems. In Chamanbaz et al. (2013, 2016) a sequential technique has been proposed for solving a scenario optimization problem. At each iteration k of the sequential algorithm, a sample complexity $N(k)$ smaller than the scenario bound is selected, a sampled optimization problem with $N(k)$ samples is solved, and the robustness of the solution is checked through a validation test. If the solution passes this validation test, the algorithm is successfully terminated; otherwise, a larger sample complexity $N(k+1) > N(k)$ is selected and a more complex optimization problem is solved. These two steps are repeated until a solution eventually passes the validation test. Borrowing ideas from statistical learning theory, in Chamanbaz et al. (2014), the sample complexity for solving robust linear and bilinear matrix inequality problems is computed and a sequential randomized method for solving the sampled optimization problem is presented. The proposed techniques in Chamanbaz et al. (2013); Chamanbaz et al. (2014); Chamanbaz et al. (2016) require a possibly large number of validation samples in order to ensure the robustness of the candidate solution at each iteration. In applications where samples are provided by experiments—making them expensive resources—it can be demanding to provide a large number of validation samples. In Calafiore (2016), an approach similar to the one in Chamanbaz et al. (2016) has been proposed for solving the scenario problem. The algorithm in Calafiore (2016) does not increase the cardinality of the design sample set at each iteration, but instead considers a probabilistic characterization of the number of iterations required to find a solution. A sequential approach based on a 'wait-and-judge scenario' optimization (Campi and Garatti, 2018b) has recently been reported in Garatti and Campi (2019) without any validation phase considered. At each iteration of the algorithm, a sampled optimization problem is solved and the number of support constraints is computed. The sample complexity at each iteration is designed such that the final solution meets the desired probabilistic robustness in terms of both accuracy and confidence levels. The presented approach in Garatti and Campi (2019) primarily focuses on reducing the number of scenarios rather than reducing the computational complexity because evaluating the number of support constraints at each iteration can be computationally demanding for problems involving a large number of sampled constraints.

⋆ This work is supported by the National Research Foundation (NRF), Prime Minister's office, Singapore, under its National Cybersecurity R&D Programme (Award No. NSoE_DeST-SCI2019-0007) and administered by the National Cybersecurity R&D Directorate.

In this paper, we propose a deterministic algorithm borrowing ideas from the well-known Clarkson Las Vegas technique for linear and integer programming (Clarkson, 1995) to directly solve the sampled optimization problem. The proposed algorithm is fundamentally deterministic and does not involve any probabilistic validation phase. From this aspect, the proposed algorithm is different from all aforementioned techniques where the objective is mostly to keep the number of sampled constraints as low as possible—leading to a less computationally demanding optimization problem—while guaranteeing the same probabilistic guarantees as the original sampled optimization problem constructed using the scenario approach or statistical learning theory. We rather propose a computationally efficient algorithm to directly solve the sampled optimization problem involving a large number of constraints. The algorithm is motivated by a *distributed* randomized constraints consensus approach presented in Chamanbaz et al. (2017, 2019) for solving robust distributed mixed-integer problems. The approach presented in Chamanbaz et al. (2017, 2019) is a probabilistic approach—unlike the deterministic algorithm presented in the current paper—in which agents perform local computation and communication in order to reach a consensus on their candidate solution. A probabilistic validation step is used in Chamanbaz et al. (2017, 2019) which has the same nature as the one used in Chamanbaz et al. (2016).

In this paper, given a sampled optimization problem—generated using statistical learning theory or the scenario approach, we propose an algorithm that converges in finite time to the optimal solution of the problem. The algorithm has a sequential nature. Specifically, each iteration of the algorithm involves two steps: *(i)* a validation step and *(ii)* an optimization step. In the validation step, the feasibility of a candidate solution for all the constraints involved in the sampled optimization problem is checked and violating constraints are identified. This step is computationally inexpensive as it only involves validation of a candidate solution and no optimization is required. Next, an optimization problem is solved whose constraint set involves: *(a)* a limited number of violated constraints observed in the validation step and *(b)* the current *basis* which is the minimal set of constraints defining the candidate solution. The validation and optimization steps are iterated until no constraint violates the candidate solution which is guaranteed to happen in a finite number of iterations.

The deterministic nature of the proposed algorithm allows one to use the exact same algorithm for any optimization problem with very large number of constraints enjoying Helly-type property; see Amenta (1994); Amenta et al. (2015). In fact, any problem possessing Helly-type property is guaranteed to have a finite-cardinality basis set. We note that this class of problems involves a fairly large category of non-convex problems.

Notation

Throughout this paper, we use capital italic letter, e.g. F to denote constraints and capital calligraphic letter, e.g. $\mathcal{F}$ to denote the set induced by the constraint F. We note that, with this notation, if $A = B \cup C$ with B and C being collections of constraints, then $\mathcal{A} = \mathcal{B} \cap \mathcal{C}$, that is, the set induced by the union of constraints B and C is the intersection of $\mathcal{B}$ and $\mathcal{C}$. An optimization problem

$$\begin{aligned} &\min\ c^T\mathbf{x} \\ &\text{subject to } \mathbf{x} \in \mathcal{F}, \end{aligned}$$

is represented by the pair (F, c). Finally, $J(F)$ is the smallest value of $c^T\mathbf{x}$ while $\mathbf{x} \in \mathcal{F}$.

2. PROBLEM FORMULATION AND PRELIMINARIES

Consider the following robust optimization problem

$$\begin{aligned} &\min\ c^T\mathbf{x} \\ &\text{subject to } \mathbf{x} \in \mathcal{F}(q),\ \ \forall q \in \mathbb{Q} \\ &\mathbf{x} \in \mathbb{R}^{d_R} \times \mathbb{Z}^{d_Z}, \end{aligned} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{d_R} \times \mathbb{Z}^{d_Z}$ is the vector of decision variables, q is the vector of uncertain parameters acting on the system, $\mathcal{F}(q) = \{\mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}, q) \leq 0\}$, with $d = d_R + d_Z$ and, $f(\mathbf{x}, q) : \mathbb{R}^d \times \mathbb{Q} \to \mathbb{R}$, is the constraint function of problem (1). The constraint function $f(\mathbf{x}, q)$ is convex for any fixed value of q if all decision variables $\mathbf{x}$ are considered to be continuous. Without loss of generality, we consider a linear objective function. In fact, a nonlinear convex objective function can be transformed into the epigraph form by introducing an extra decision variable. If $d_Z = 0$, the problem (1) is a usual continuous convex optimization problem; if $d_R = 0$, (1) becomes an integer optimization problem and choosing $d_r \neq 0, d_Z \neq 0$ leads to a mixed-integer optimization.

One of the efficient methods to find an approximate solution to problem (1) with desired probabilistic robustness guarantee is to convert it to a sampled optimization problem using the scenario approach (Calafiore and Campi, 2004, 2006; Calafiore, 2010; Calafiore et al., 2012; Campi and Garatti, 2018a) or statistical learning theory (Vidyasagar, 2001; Alamo et al., 2009)—if the so-called Vapnik–Chervonenkis dimension of the problem (1) is finite. With these approaches, the semi-infinite optimization problem (1) is converted into an optimization problem with a finite number of constraints by extracting random samples from the set of uncertainty $\mathbb{Q}$ and constructing constraints only at the extracted samples. Formally, N independent and identically distributed (i.i.d) samples are extracted from the set $\mathbb{Q}$

$$\mathbf{q} = \{q^{(1)}, \ldots, q^{(N)}\} \in \mathbb{Q}^N$$

where $\mathbb{Q}^N = \mathbb{Q} \times \mathbb{Q} \times \ldots \times \mathbb{Q}$ (N times) and the following sampled optimization problem is formulated

$$\begin{aligned} \mathbf{x}_N^* = \arg\min\ & c^T\mathbf{x} \\ \text{subject to } & \mathbf{x} \in \bigcap_{i=1}^{N} \mathcal{F}(q^{(i)}) \\ & \mathbf{x} \in \mathbb{R}^{d_R} \times \mathbb{Z}^{d_Z}. \end{aligned} \tag{2}$$

The solution to problem (2) can be identified by only a few number of constraints. This concept is formally characterized by the notion of *basis*. Specifically, a basis is the minimal set of constraints that defines a solution. The concept of basis is supported by Helly-type theorems initially introduced by Eduard Helly in Helly (1923), see also Amenta (1994); Amenta et al. (2015). The primary goal of this paper is to find basis for the optimization problem (2) and correspondingly its solution $\mathbf{x}_N^*$ in a computationally efficient manner.

Definition 1. (Basis). Given a collection of constraints F, a subset of minimal cardinality $B \subseteq F$ is a basis of F if the optimal cost of the problem defined by (F, c) is identical to the one defined by (B, c), and the optimal cost decreases if any constraint is removed.

The size of the largest basis for a problem (F, c) is called its *combinatorial dimension*. The following Theorem adopted from (Calafiore et al., 2012, Corollary 1), and (Amenta et al., 2015, Theorem 3.11) defines the combinatorial dimension of the mixed-integer problem (2).

Theorem 1. The combinatorial dimension of problem (2) is $(d_R + 1)2^{d_Z} - 1$.

We make the following assumption regarding the solution of any subproblem of (2).

Assumption 1. (Non-degeneracy). The minimum point of any subproblem of (2) with at least $(d_R + 1)2^{d_Z} - 1$ constraints is unique and there exist only $(d_R + 1)2^{d_Z} - 1$ constraints intersecting at the minimum point.

Assumption 1 is not restrictive. In fact, to ensure uniqueness of the optimal point, one can use a strictly convex objective function, a lexicographic ordering, or any universal tie-breaking rule, see (Amenta, 1994, Observation 8.1) for further details.

The following Theorem form (Calafiore et al., 2012, Theorem 3 and Corollary 2) characterizes the robustness property of $\mathbf{x}_N^*$.

Theorem 2. Suppose Assumption 1 holds. Given probabilistic accuracy and confidence levels $\varepsilon, \delta \in (0, 1)$, let N be the smallest integer satisfying

$$\delta \geq \sum_{\ell=0}^{(d_R+1)2^{d_Z}-2} \binom{N}{\ell} \varepsilon^{\ell} (1-\varepsilon)^{N-\ell}. \tag{3}$$

Then the solution of (2) $\mathbf{x}_N^*$ satisfies

$$\mathbb{P}^N \left\{ \mathbf{q} \in \mathbb{Q}^N : \mathbb{P}\left\{ q \in \mathbb{Q} : \mathbf{x}_N^* \notin \mathcal{F}(q) \right\} \leq \varepsilon \right\} \geq 1 - \delta,$$

where $\mathbb{P}^N$ is the product probability measure on $\mathbb{Q}^N$.

The sample complexity N can be computed by numerically solving (3). Results similar to Theorem 2 can be derived using the Vapnik–Chervonenkis (VC) learning theory (Vapnik and Chervonenkis, 1971). To this end, a combinatorial parameter called VC dimension of problem (1) has to be finite. We remark that reporting the sample complexity using statistical learning theory falls outside the scope of this paper. Interested readers are referred to Vidyasagar (2001); Alamo et al. (2009); Chamanbaz et al. (2014). The sample complexity computed using statistical learning theory is usually very large leading to a computationally challenging sampled optimization (2); see e.g. (Alamo et al., 2009, Theorem 3), (Chamanbaz et al., 2014, Corollary 2). In the scenario approach, the number of samples N is inversely proportional to the accuracy level ε and has a logarithmic relationship with the inverse of confidence parameter δ. Hence, if the accuracy and confidence levels ε, δ are required to be small, the number of scenario samples N computed in (3) can become large leading to a possibly complex sampled optimization problem (2). This calls for the necessity of a computationally efficient algorithm for solving the sampled optimization problem (2). In the next section, we report a deterministic sequential algorithm for solving sampled optimization problem (2).

Algorithm 1 Sequential Algorithm

Input: c, r, d_R, d_Z, $F(q^{(i)}), i = 1, \dots N$,
Output: $\mathbf{x}_{\texttt{seq}}, B_{\texttt{seq}}$
Initialization:
Set $m = (d_R + 1)2^{d_Z}$, $\texttt{feasible} = 0$, $t = 0$
$[\mathbf{x}(0), B(0)] = \texttt{Solve}_{\texttt{MIP}}(F(q^{(1)}) \cup \ldots \cup F(q^{(m)}), c)$
Evolution:
while $\texttt{feasible} == 0$ **do**
  $[F^{\texttt{Viol}}, \texttt{feasible}] = \texttt{Verification}(F(q^{(1)}), \dots, F(q^{(N)}), \mathbf{x}(t), r)$
  $[\mathbf{x}(t+1), B(t+1)] = \texttt{Solve}_{\texttt{MIP}}(F^{\texttt{Viol}} \cup B(t), c)$
end while
Set $\mathbf{x}_{\texttt{seq}} = \mathbf{x}(t+1)$ and $B_{\texttt{seq}} = B(t+1)$
return $\mathbf{x}_{\texttt{seq}}, B_{\texttt{seq}}$

3. SEQUENTIAL ALGORITHM

In this section, we present a sequential algorithm for solving the sampled mixed-integer problem formulated in (2). We first define two primitives. The first primitive is $[F^{\texttt{Viol}}, \texttt{feasible}] = \texttt{Verification}(F(q^{(1)}), \dots, F(q^{(N)}), \mathbf{x}, r)$ which checks the feasibility of a candidate solution $\mathbf{x}$ for all the sampled constraints involved in (2), i.e. $F(q^{(1)}), \dots, F(q^{(N)})$ and—if possible—finds r violated ones. Furthermore, if there is no violation, the flag `feasible` is set to 1 otherwise, $\texttt{feasible} = 0$. The second primitive is $[\mathbf{x}, B] = \texttt{Solve}_{\texttt{MIP}}(F, c)$. This primitive solves the optimization problem defined by the pair (F, c) and returns back the optimal point $\mathbf{x}$ and the corresponding basis B. The two primitives constitute the two main parts of the algorithm. In the first part of the proposed algorithm, a candidate solution $\mathbf{x}$ is examined—using `Verification` primitive—to see if it satisfies all N constraints of the optimization problem (2) and—if possible—r violated constraints are found. Next, the algorithm solves an optimization problem—using the primitive $\texttt{Solve}_{\texttt{MIP}}$—whose constraint set consists of (i) the r violated constraints and (ii) the current set of basis. The algorithm iterates the two steps until there is no violating constraint left. This algorithm is formally presented in Algorithm 1.

We now state few remarks regarding Algorithm 1.

Remark 1. Algorithm 1 is not limited to solving sampled optimization problems. It is a Clarkson-type algorithm and shows significant computational improvement for problems for which: *(i)* the number of constraints is much larger than the number of decision variables and *(ii)* the Helly-type property holds, see Amenta (1994); Amenta et al. (2015).

Remark 2. (Complexity of each iteration). An important feature of Algorithm 1 is that the complexity of the optimization problem being solved at each iteration does not increase with the iteration counter. In fact, the number of constraints involved in the optimization problem at each iteration is *at most* $r + (d_R + 1)2^{d_Z} - 1$. For instance, for a mixed-integer optimization problem with $d_R = 5$, $d_Z = 3$

if we set $r = 10$, the number of constraints can be at most 57; or in a continuous optimization problem in which the dimension of the solution space is $d_R = 5$, the number of constraints can be at most 15. Therefore, at each iteration, an optimization problem of fixed—and small—complexity is solved.

Remark 3. (Complexity of verification step). The verification step of Algorithm 1 is computationally inexpensive since it only requires checking the feasibility of a candidate solution for the N constraints present at problem (2).

Remark 4. (Finding basis). In a continuous optimization problem, basis is defined as the minimal set of active constraints. However, in a mixed-integer problem, finding basis can be computationally more demanding. A computationally manageable way to compute a basis which might not necessarily be of minimal cardinality is to check constraints one by one to see whether or not they can belong to basis. To this end, we consider the optimization problem at hand and drop its ith constraint. If the objective value returned by this optimization problem is smaller than the objective value of the original problem, the dropped constraint can be a part of basis. However, we note that the number of constraints of the optimization problem for which we need to find basis—the problem formed at line 7 of Algorithm 1—is at most $r+(d_R+1)2^{d_Z}-1$. Hence, it is not computationally expensive to find basis as the problem has a limited number of constraints. We also remark that a similar greedy algorithm is being used in the context of scenario with discarded constraints (Campi and Garatti, 2011; Calafiore, 2010) and very recently in wait-and-judge scenario optimization (Campi and Garatti, 2018b; Garatti and Campi, 2019).

Remark 5. The number of violated samples r in Algorithm 1 constitutes a trade-off between the complexity of the optimization problem being solved at each iteration of the algorithm and the number of iterations required for convergence. Increasing r would result in a more complex optimization problem solved at line 7 of the algorithm. However, one would expect a lower number of iterations required to terminate the algorithm; see subsection 4.2 and in particular Table 2 for a numerical confirmation of this observation.

The properties of Algorithm 1 are summarized in the following Theorem.

Theorem 3. Let Assumption 1 holds. Then, the following statements hold.

(i) The objective value of the candidate solution $c^T\mathbf{x}(t) = J(B(t))$ is monotonically increasing while Algorithm 1 is progressing.
(ii) Algorithm 1 terminates in finite time.
(iii) The solution returned by Algorithm 1, $\mathbf{x}_{\mathtt{seq}}$ is identical to $\mathbf{x}_N^*$.

Proof: Note that in constructing the basis at time $t+1$ i.e. $B(t+1)$, we use the basis at time t, i.e. $B(t)$. Hence, $J(B(t+1)) \geq J(B(t))$. On the other hand, there has to be at least one violating constraint $F^{\mathtt{viol}}$ in all the iterations of Algorithm 1 except the last iteration. Indeed, if there had not been a violating constraint, the condition at line 5 would have been satisfied and the algorithm would have terminated. This means that at line 7 of Algorithm 1, we solve an optimization problem whose constraints set involves the current basis $B(t)$ and at least one violating constraint $F^{\mathtt{viol}}$. Therefore, due to the presence of the violating constraint(s) $F^{\mathtt{viol}}$, and due to Assumption 1, the cost has to increase, i.e. $J(B(t+1)) > J(B(t))$. This proves the first statement of Algorithm 1.

There are finite number of constraints involved in (2); hence, there are finite number of candidate basis leading to a finite number of candidate costs $J(B(t))$. Furthermore, the cost $J(B(t))$ is strictly increasing with the iteration counter t—as proved in the first statement. Since, the sequence $\{J(B(t))\}_{t>0}$ is strictly increasing and has a finite number of elements, it will converge in a finite number of iterations leading to the finite-time termination of the algorithm. This proves the second statement of the theorem.

We first note that since at any iteration t of the algorithm, a sub-problem of mixed-integer problem (2) is being solved, $J(B(t))$ cannot be greater than $J(F) = c^T\mathbf{x}_N^*$, where $F \doteq \bigcup_{i=1}^N F(q^{(i)})$; then, $J(B(t)) \leq J(F)$, $\forall t > 0$ and as a result $J(B_{\mathtt{seq}}) \leq J(F)$. We now show that $J(B_{\mathtt{seq}})$ cannot be smaller than $J(F)$. Assume by contradiction that $J(B_{\mathtt{seq}}) < J(F)$ or equivalently $J(B_{\mathtt{seq}}) < J(B_{\mathtt{seq}} \cup F)$ as $B_{\mathtt{seq}} \subseteq F$. By construction, $\mathbf{x}_{\mathtt{seq}}$ must satisfy all the constraints in (2) as it has passed the verification step of Algorithm 1; then, $\mathbf{x}_{\mathtt{seq}} \in \mathcal{F}$ with $\mathcal{F} \doteq \bigcap_{i=1}^N \mathcal{F}(q^{(i)})$. Furthermore, since by definition $\mathcal{F} \subseteq \mathcal{B}_{\mathtt{seq}}$; then, $\mathbf{x}_{\mathtt{seq}} \in \mathcal{F} \cap \mathcal{B}_{\mathtt{seq}}$. Now, considering the point that $\mathcal{B}_{\mathtt{seq}}$ is the set induced by the basis $B_{\mathtt{seq}}$ corresponding to $\mathbf{x}_{\mathtt{seq}}$, we have $J(B_{\mathtt{seq}}) \geq J(F \cup B_{\mathtt{seq}})$ which contradicts our earlier assumption that $J(B_{\mathtt{seq}}) < J(B_{\mathtt{seq}} \cup F)$. Hence, $J(B_{\mathtt{seq}})$ can neither be greater nor smaller than $J(F) = c^T\mathbf{x}_N^*$. Therefore, $J(B_{\mathtt{seq}}) = c^T\mathbf{x}_N^*$ or equivalently, $c^T\mathbf{x}_{\mathtt{seq}} = c^T\mathbf{x}_N^*$. This combined with Assumption 1 concludes the proof.

4. NUMERICAL EXAMPLE

We ran extensive numerical simulations to test performance of Algorithm 1. We considered different problems including Linear Programming (LP), Mixed-integer Linear Programming (MILP), Quadratically Constrained Quadratic Programming (QCQP) and Mixed-integer Quadratically Constrained Quadratic Programming (MIQCQP) of various sizes to prove the effectiveness of the proposed algorithm. We compared the performance in terms of the time required to solve the problem. The performance of the algorithm is compared against the widely used commercial solver Cplex (Cpl, 2018). We note that Cplex exhibits a superior performance compared to solvers such as Mosek (Andersen and Andersen, 2000), SDPT3 (Toh et al., 1999), and SEDUMI (Labit et al., 2002), which is the reason for choosing Cplex over other commercial solvers as benchmark. All simulations are performed on a Dell $Z660$ Workstation running Linux with 12 Cores and 48 GB of RAM.

Table 1. Average—over 20 randomly generated sampled optimization problems—CPU time it takes for the Cplex solver to find the solution to the sampled optimization problem (5), average CPU time it takes for Algorithm 1 to converge, and average number of iterations required for Algorithm 1 to converge. The simulation is performed 20 times for each row and average results are reported. The entry NA indicates that Cplex requires more than 48 GB of RAM to run.

Problem Type	# Constraints N	# Continuous Variables (d_R)	# Discrete Variables (d_Z)	Average CPU Time Cplex (sec)	Average CPU Time Algorithm 1 (sec)	Average Iteration Counter at Convergence
LP	1×10^4	10	0	0.2198	0.2779	4.33
LP	5×10^5	10	0	10.15	0.3521	9.28
LP	2×10^6	10	0	52.04	0.5549	10.23
LP	1×10^4	100	0	2.37	2.27	196.33
LP	5×10^5	100	0	103.98	18.69	487
LP	2×10^6	100	0	NA	66.91	600
MILP	1×10^4	5	5	0.9239	6.49	4.28
MILP	5×10^4	5	5	26.09	8.23	4.8
MILP	1×10^6	5	5	NA	25.51	7.52
MILP	1×10^4	20	5	1.41	42.28	6.88
MILP	1×10^5	20	5	621.44	48.12	10.87
MILP	2×10^6	20	5	NA	89.49	16.8

Table 2. Average—over 20 randomly generated sampled optimization problems—CPU time it takes for the Cplex solver to find the solution to the sampled optimization problem (7), average CPU time it takes for Algorithm 1 to converge and average number of iterations required for Algorithm 1 to converge. The simulation is performed 20 times for each row and average results are reported. The entry NA indicates that Cplex requires more than 48 GB of RAM to run.

Problem Type	# Constraints N	# Continuous Variables (d_R)	# Discrete Variables (d_Z)	# Violated Constraints (r)	Average CPU Time Cplex (sec)	Average CPU Time Algorithm 1 (sec)	Average Iteration Counter at Convergence
QCQP	1×10^4	10	0	20	51.57	2.17	4.71
QCQP	5×10^5	10	0	20	1.98×10^3	2.50	7.66
QCQP	5×10^5	10	0	100	1.98×10^3	1.62	3.85
QCQP	5×10^5	10	0	5	1.98×10^3	3.69	19.9
QCQP	1×10^6	10	0	20	4.35×10^3	4.65	10.23
MIQCQP	1×10^4	5	5	20	4.97×10^3	179.34	3.61
MIQCQP	5×10^5	5	5	20	NA	303.53	5.04
MIQCQP	1×10^6	5	5	20	NA	560.34	9.61

4.1 Robust Mixed-integer Linear Programs

We consider robust Linear Programming (LP) and robust Mixed integer Linear Programming (MILP) problems having the following structure

$$\begin{aligned} &\text{minimize} && c^T\mathbf{x} \\ &\text{subject to: } && (A^0 + A_q)\mathbf{x} \leq b, \\ &&& \mathbf{x} \in \mathbb{R}^{d_R} \times \mathbb{Z}^{d_Z}, \end{aligned} \tag{4}$$

where the vector $c \in \mathbb{R}^d$ defines the objective direction, $A^0 \in \mathbb{R}^{d\times d}$, $b \in \mathbb{R}^d$ are the nominal (fixed) matrix and vector defining the nominal constraint set of problem (4), and $A_q \in \mathbb{R}^{d\times d}$ is an interval matrix—a matrix whose entries vary in given intervals—defining the uncertainty in optimization problem (4). The vectors b, c and nominal matrix A^0 are generated such that problem (4) is feasible. To this end, we follow the methodology presented in (Dunham et al., 1977). In particular, elements of A^0 and c are drawn from a standard Gaussian distribution, i.e. zero mean and unit standard deviation. The ℓ-th element of b is defined by $b_\ell = \gamma\left(\sum_{m=1}^{d}\left(A^0_{\ell m}\right)^2\right)^{1/2}$, where $\gamma > 1$ manipulates the feasibility region of the MILP problem. The larger the variable γ, the bigger the volume of feasible region. Note that the procedure presented in Dunham et al. (1977) generates random feasible LP however, a feasible LP can become infeasible for the case of MILP. Therefore, in the set of simulations reported here, we set $\gamma = 10$ to make sure that the generated MILP is indeed feasible. The entries of A_q are bounded in $[-0.2, 0.2]$. The sampled version of problem (4) is constructed by extracting random samples $\{A_q^{(i)}\}_{i=1}^N$ from the set of uncertainty

$$\begin{aligned} &\text{minimize} && c^T\mathbf{x} \\ &\text{subject to: } && (A^0 + A_q^{(i)})\mathbf{x} \leq b,\ i = 1, \ldots, N \\ &&& \mathbf{x} \in \mathbb{R}^{d_R} \times \mathbb{Z}^{d_Z}. \end{aligned} \tag{5}$$

For computational study, we vary the dimension of the solution space d_R, d_Z and the number of sampled constraints N and solve problem (5) using Algorithm 1 and

the commercial solver Cplex—using `cplexlp` command for the case $d_Z = 0$ and `cplexmilp` command for the case $d_Z \neq 0$. The performance in terms of the amount of time it takes to solve the problem is compared in Table 1. As can be seen in Table 1, for small problems, the commercial solver Cplex outperforms Algorithm 1 however, for large problems, Algorithm 1 outperforms Cplex as the simulation time is considerably smaller compared to Cplex. Note that for some of the problems reported in Table 1, the Cplex solver requires more than 48 GB of RAM to solve the formulated problem and hence, it is not possible to solve the problem on our workstation using Cplex. This shows an important advantage of Algorithm 1 in significantly saving memory required for solving sampled optimization problems involving large number of constraints. For all the simulations reported in Table 1, the number of violated constraints r is set to 50.

4.2 Robust Mixed-integer Quadratically Constrained Quadratic Programs

The problem we deal with in this subsection is the following robust Mixed-integer Quadratically Constrained Quadratic Program (MIQCQP)

$$\begin{aligned} &\text{minimize } \mathbf{x}^T P\mathbf{x} + s^T\mathbf{x} \\ &\text{subject to: } \mathbf{x}^T P_q\mathbf{x} + (s^0 + s_q)^T\mathbf{x} + (r^0 + r_q) \leq 0 \\ &\qquad \mathbf{x} \in \mathbb{R}^{d_R} \times \mathbb{Z}^{d_Z} \end{aligned} \tag{6}$$

where $P_q = (D^0 + D_q)^T(D^0 + D_q) \in \mathbb{S}_+^{d\times d}$—with $\mathbb{S}_+^{d\times d}$ being the set of all d-dimensional symmetric positive semi-definite matrices—$s^0, s_q \in \mathbb{R}^d$, and $r^0, r_q \in \mathbb{R}$ define the robust MIQCQP.

In order to make sure that the nominal problem is feasible, we select a point $\mathbf{x}_0 \in \mathbb{R}^{d_R} \times \mathbb{Z}^{d_Z}$ at random and then compute r^0 such that $\mathbf{x}_0^T P^0\mathbf{x}_0 + (s^0)^T\mathbf{x}_0 + r^0 \leq -\gamma$ with $P^0 = (D^0)^T D^0$ and $\gamma \geq 0$. The uncertain matrix D_q is selected to be an interval matrix whose entries vary in $[-0.1,\ 0.1]$ and s_q, r_q are drawn from the Gaussian distribution with zero mean and unit standard deviation. The sampled version of MIQCQP (6) can be constructed by extracting random samples $\{D_q^{(i)}\}_{i=1}^N, \{s_q^{(i)}\}_{i=1}^N, \{r_q^{(i)}\}_{i=1}^N$ from the set of uncertainty

$$\begin{aligned} &\text{minimize } \mathbf{x}^T P\mathbf{x} + s^T\mathbf{x} \\ &\text{subject to: } \mathbf{x}^T P_q^{(i)}\mathbf{x} + (s^0 + s_q^{(i)})^T\mathbf{x} \\ &\qquad + (r^0 + r_q^{(i)}) \leq 0,\ i = 1, \ldots, N \\ &\qquad\qquad \mathbf{x} \in \mathbb{R}^{d_R} \times \mathbb{Z}^{d_Z}. \end{aligned} \tag{7}$$

We remark that problem (7) will not be unbounded from below because the feasible set is the intersection of N ellipsoids and hence bounded as far as ellipsoids are non-degenerate. Problem (7) is solved using Algorithm 1 for different values of N, d_R, d_Z. The results in terms of the time it takes for Algorithm 1 to converge is compared with the commercial solver Cplex—using `cplexqcp` function for the case $d_Z = 0$ and `cplexmiqcp` function for the case $d_Z \neq 0$—are reported in Table 2. The effect of the number of violated constraints r is explored in Table 2. Increasing r from 20 to 100 decreases the average CPU time as well as the number of iterations is takes for Algorithm 1 to converge. Similarly, decreasing r to 5 increases the CPU time as well as the number of iterations.

We note that more complex uncertainty structure than the ones used in (4) and (6) can be handled using sampling-based techniques such as the scenario approach. In fact, one of the major advantages of using a randomized technique over a deterministic one in solving robust optimization problems is that they are not limited to the uncertainty structure.

For continuous optimization problems, e.g. LP and QCQP, it is straightforward to find the set of bases. The procedure for finding a basis involves: (i) finding active constraints by checking which constraints hold with equality in (5) and (7) which is computationally inexpensive even for problems involving a very large number of constraints and (ii) checking which of the active constraints belong to the set of bases by firstly, forming an optimization problem whose constraints set includes only active constraints and secondly, removing the active constraints one at a time and solving the resulting optimization problem and checking if the resulting objective value becomes smaller. If any improvement in the objective value is observed, the removed constraint can belong to the set of bases.

It is more computationally demanding for a problem involving integer decision variables, e.g. MILP and MIQCQP than a continuous problem to find the set of bases. In fact, for a mixed-integer problem, we need to use a brute-force approach and check the constraints one by one to find out which ones belong to the set of bases. However, as mentioned earlier, the number of constraints involved in the optimization problem being formed at each iterations of the algorithm is limited. Hence, it is not computationally expensive to find the set of basis in Algorithm 1.

5. CONCLUSION

In this paper, we presented a deterministic algorithm for solving sampled optimization problems. The algorithm exhibits a significant saving in the time and memory required for solving sampled optimization problems. The algorithm can be immediately used for solving a far more general class than sampled optimization problems: any problem possessing Helly-type property in which the number of constraints is much larger than the dimension of the solution space. The algorithm has two main steps: verification and optimization. In the verification step, feasibility of a candidate solution is examined and violating constraints are identified. In the optimization step, an optimization problem whose constraint set involves current basis and few violated constraints is solved. The convergence properties of the algorithm are analyzed and its performance in solving mixed-integer sampled optimization problems is compared against a widely used commercial solver.

REFERENCES

(2018). ILOG CPLEX Optimization Studio - Overview. URL `https://www.ibm.com/sg-en/products/ilog-cplex-optimization-studio`.

Alamo, T., Tempo, R., and Camacho, E. (2009). Randomized strategies for probabilistic solutions of uncertain feasibility and optimization problems. *IEEE Transactions on Automatic Control*, 54, 2545–2559.

Amenta, N. (1994). Helly-type theorems and Generalized Linear Programming. *Discrete & Computational Geometry*, 12(3), 241–261.

Amenta, N., De Loera, J., and Soberòn, P. (2015). Helly's Theorem: New Variations and Applications. *arXiv:1508.07606 [math]*. URL `http://arxiv.org/abs/1508.07606`.

Andersen, E. and Andersen, K. (2000). The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization*, 197–232. Springer.

Calafiore, G. and Campi, M. (2004). Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102(1), 25–46.

Calafiore, G. and Campi, M. (2006). The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51(5), 742–753. doi:10.1109/TAC.2006.875041.

Calafiore, G., Lyons, D., and Fagiano, L. (2012). On mixed-integer random convex programs. In *Proc. 51st IEEE Annual Conference on Decision and Control (CDC)*, 3508–3513. IEEE.

Calafiore, G.C. (2016). Repetitive scenario design. *IEEE Transactions on Automatic Control*, 62(3), 1125–1137.

Calafiore, G.C. (2010). Random convex programs. *SIAM Journal on Optimization*, 20(6), 3427–3464.

Campi, M. and Garatti, S. (2011). A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. *Journal of Optimization Theory and Applications*, 148(2), 257–280.

Campi, M. and Garatti, S. (2018a). *Introduction to the Scenario Approach*. Society for Industrial and Applied Mathematics.

Campi, M.C. and Garatti, S. (2018b). Wait-and-judge scenario optimization. *Mathematical Programming*, 167(1), 155–189. doi:10.1007/s10107-016-1056-9.

Chamanbaz, M., Dabbene, F., Tempo, R., Venkataramanan, V., and Wang, Q. (2013). Sequential randomized algorithms for sampled convex optimization. In *Proc. 2013 IEEE Conference on Computer Aided Control System Design (CACSD)*, 182–187. doi:10.1109/CACSD.2013.6663480.

Chamanbaz, M., Dabbene, F., Tempo, R., Venkataramanan, V., and Wang, Q. (2016). Sequential randomized algorithms for convex optimization in the presence of uncertainty. *IEEE Transactions on Automatic Control*, 61(9), 2565–2571. doi:10.1109/TAC.2015.2494875.

Chamanbaz, M., Dabbene, F., Tempo, R., Venkataramanan, V., and Wang, Q.G. (2014). A statistical learning theory approach for uncertain linear and bilinear matrix inequalities. *Automatica*, 50(6), 1617–1625. doi:10.1016/j.automatica.2014.04.005.

Chamanbaz, M., Notarstefano, G., and Bouffanais, R. (2017). Randomized constraints consensus for distributed robust linear programming. *IFAC-PapersOnLine*, 50(1), 4973 – 4978. doi:https://doi.org/10.1016/j.ifacol.2017.08.763. Proc. 20th IFAC World Congress.

Chamanbaz, M., Notarstefano, G., Sasso, F., and Bouffanais, R. (2019). Randomized constraints consensus for distributed robust mixed-integer programming. *arXiv:1907.04691*.

Clarkson, K.L. (1995). Las vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2), 488–499. doi:10.1145/201019.201036.

Dunham, J., Kelly, D., and Tolle, J. (1977). Some Experimental Results Concerning the Expected Number of Pivots for Solving Randomly Generated Linear Programs. Technical Report TR 77-16, Operations Research and System Analysis Department, University of North Carolina at Chapel Hill.

Garatti, S. and Campi, M.C. (2019). Complexity-based modulation of the data-set in scenario optimization. In *2019 18th European Control Conference (ECC)*, 1386–1391. doi:10.23919/ECC.2019.8796160.

Helly, E. (1923). Über mengen konvexer körper mit gemeinschaftlichen punkte. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32, 175–176.

Labit, Y., Peaucelle, D., and Henrion, D. (2002). SEDUMI INTERFACE 1.02: a tool for solving LMI problems with SEDUMI. In *Proc. 2002 IEEE International Symposium on Computer Aided Control System Design, 2002. Proceedings*, 272 – 277.

Toh, K.C., Todd, M.J., and Tütüncü, R.H. (1999). SDPT3 – a MATLAB software package for semidefinite programming. *Optimization Methods and Software*, 11, 545–581.

Vapnik, V. and Chervonenkis, A. (1971). On the uniform convergence of relative frequencies to their probabilities. *Theory of Probability and Its Applications*, 16, 264–280.

Vidyasagar, M. (2001). Randomized algorithms for robust controller synthesis using statistical learning theory. *Automatica*, 37, 1515–1528.

Vidyasagar, M. (2002). *Learning and Generalization: With Applications to Neural Networks*. Springer, 2nd edition.