2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Regression-based LP Solver for Chance-Constrained Finite Horizon Optimal Control with Nonconvex Constraints

Ashis Gopal Banerjee*, Masahiro Ono*, Nicholas Roy, and Brian C. Williams

Abstract— **This paper presents a novel algorithm for finite-horizon optimal control problems subject to additive Gaussian-distributed stochastic disturbance and chance constraints that are defined over feasible, *non-convex* state spaces. Our previous work [1] proposed a branch and bound-based algorithm that can find a near-optimal solution by iteratively solving non-linear convex optimization problems, as well as their LP relaxations called *Fixed Risk Relaxation (FRR)* problems.**

The aim of this work is to significantly reduce the computation time of the previous algorithm so that it can be applied to practical problems, such as a path planning with multiple obstacles. Our approach is to use machine learning to efficiently estimate the objective function values of FRRs within an error bound that is fixed for a given problem domain and choice of model complexity. We exploit the fact that all the FRR problems associated with the branch-and-bound tree nodes are similar to each other, both in terms of the solutions as well as the objective function and constraint coefficients. A standard optimizer is first used to generate a training data set in the form of optimal FRR solutions. Matrix transformations and boosting trees are then applied to generate learning models; fast inference is performed at run-time for new but similar FRR problems that occur when the system dynamics and/or the environment changes slightly. By using this regression technique to estimate the lower bound of the cost function value, and subsequently solving the convex optimization problems exactly at the leaf nodes of the branch-and-bound tree, we achieve 10-35 times reduction in the computation time without compromising the optimality of the solution.

I. Introduction

A. Motivation and Overview

We consider the finite-horizon robust optimal control of dynamic systems under unbounded Gaussian-distributed uncertainty, with non-convex state constraints. Stochastic uncertainty with a probability distribution, such as Gaussian, is a more natural model for exogenous disturbances, such as wind gusts and turbulence, than previously studied set-bounded models such as [2]. An effective framework to address robustness with stochastic unbounded uncertainty is optimization with a *chance constraint* [3]. A chance constraint requires that the probability of violating the state constraints (i.e., the probability of failure) is below a user-specified bound known as the *risk bound*. The non-convex chance-constrained optimal control has important applications such as robust path planning in the presence of obstacles.

In our previous work, we developed two novel methods called *risk allocation* [4] and *risk selection* [1] that decompose a joint non-convex chance constraint into a disjunctive set of individual chance constraints, which again can be converted to deterministic constraints. The optimal solution to the resulting disjunctive convex programming problem is obtained by using a branch and bound algorithm, where non-linear convex programming problems are solved repeatedly. We also introduced a novel LP relaxation of the non-linear convex programing problem called *Fixed Risk Relaxation (FRR)*, whose solution gives the lower bound of the cost. It was empirically shown that the FRR makes the branch-and-bound algorithm faster by 10-20 times. However, the computation time is still not fast enough for several problem domains. For example, it requires about 35 seconds to solve a path planning problem with 10 time steps and only one obstacle. We found that most of the computation time is consumed in solving thousands of FRRs repeatedly.

Now, FRR problems in a particular branch and bound tree often share multiple common constraints and always contain the same objective function and number of decision variables. Uncertainties in the operating conditions may result in slightly different problems with some changes in the objective function and constraint coefficient values and generation of some new constraints and/or deletion of old ones without altering the number of decision variables. Henceforth, we refer to problems that contain the same number of decision variables, similar objective function coefficients, and similar constraint coefficients with some variation in the number of equality constraints as a set of *similar* LP problems.

We observed that the solutions of similar LP problems are usually quite similar themselves in the sense that most of the optimum decision variable values do not change much. Hence, we show that supervised machine learning, and more specifically, regression, can be used to learn from the solutions of given feasible FRR problems to predict the solutions of new but similar LP problems. Fast inference is then performed over such regression models at run-time to quickly estimate the LP solutions instead of computing them using standard optimizations solvers.

In this paper, we present a boosting tree-based approach to learn a set of functions that map the objective function and constraints to the individual decision variables of similar LP problems. Matrix transformations are used to convert

This material is based upon work supported in part by the National Science Foundation under Grant No. IIS-1017992, by the Boeing Company under Grant No. MIT-BA-GTA-1, and by the Office of Naval Research Science of Autonomy program under Contract No. N000140910625. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the sponsoring agencies.

Ashis Gopal Banerjee is a Postdoctoral Associate at MIT. ashis@mit.edu
Masahiro Ono is a Ph.D. student at MIT. hiro_ono@mit.edu
Nicholas Roy is an Associate professor at MIT. nickroy@mit.edu
Brian C. Williams is a Professor at MIT. williams@mit.edu
*Both authors contributed equally to this work.

the objective function vector and constraint equation coefficients to a single predictor variable vector. The decision variables themselves constitute the response variables. This formulation enables us to provide absolute worst-case bounds on the objective function prediction error resulting from potential approximation errors induced by the regressor, which are used in the branch and bound framework for pruning purposes. In order to bias the boosting trees to avoid predicting infeasible values when feasible solutions exist, we modify the standard loss function to penalize response values that lie outside the feasible region of the training set LP problems as the constant modeled value inside the boosting tree regions.

We use this regression technique to solve the FRRs approximately to estimate the lower bound on the overall cost function and then solve the convex chance-constrained optimization problems at the leaf nodes exactly in the branch-and-bound algorithm. This approach significantly reduces the computation time of the algorithm without compromising the optimality of the solution. Empirical results show that the regression-based LP solver enhances the speed of the branch-and-bound algorithm by 10-35 times, thereby enabling us to solve chance-constrained path planning problems with multiple obstacles (up to 5) and long planning time horizons (up to 30) within a few seconds.

B. Related Work

There is a significant body of work that solves convex joint chance constrained optimization problems, such as [5], [6], [7]. On the other hand, to the best of our knowledge, there are only three prior methods [1], [8], [9] that handle non-convex chance-constrained optimization. Although the sampling based method [8] is very general, slow computation is a major bottleneck. Moreover, it may result in an infeasible solution due to sampling errors. These issues are addressed by [9] that used the Boole's inequality to decompose the chance constraint into a set of individual chance constraints, which can be evaluated analytically. Since the set of individual chance constraints provides a sufficient condition for the original chance constraint, this approach always results in a feasible solution. Although it is efficient, the solution has significant suboptimality since the risk bounds of the individual chance constraints are arbitrarily fixed. The state-of-the-art approach proposed by [1] formulated the risk bounds as explicit optimization parameters in order to obtain near-optimal solutions. The resulting disjunctive convex programming problem is solved by a branch-and-bound algorithm, where convex optimization problems, as well as FRRs, are solved iteratively. Each single FRR is an LP which must be solved repeatedly; it is this substantial computational cost which is addressed in this paper.

In terms of using machine learning to solve optimal planning and control problems, a lot of work has been done on reinforcement learning (both in direct and inverse forms) for problems that can be cast in the form of Markov decision processes [10], [11]. However, relatively little work has been done to learn the solutions of problems that are cast in combinatorial optimization form. Few such representative work in the planning domain includes application of temporal difference learning by Zhang and Dietterich [12] and usage of naïve Bayes and decision trees by Vladušič et al. [13] for job scheduling problems. Different evolutionary techniques, such as genetic algorithm, ant colony optimization etc. have also been used in [14], [15] to solve vehicle routing problems.

II. Review of Non-convex Chance-Constrained Optimal Control Problem

Notation: The following notation is used throughout this paper except in Section III.

x_t	:	State vector at t'th time step (random variable)
u_t	:	Control input at t'th time step.
w_t	:	Additive disturbance at t'th time step (random variable)
$\bar{x}_t$	$:=$	$E[x_t]$: Nominal state at t'th time step
$\mathbb{U}$	:	Convex feasible set for U

$$X := \begin{bmatrix} x_0 \\ \vdots \\ x_T \end{bmatrix} \quad U := \begin{bmatrix} u_0 \\ \vdots \\ u_{T-1} \end{bmatrix} \quad \bar{X} := \begin{bmatrix} \bar{x}_0 \\ \vdots \\ \bar{x}_T \end{bmatrix}$$

A. Problem Statement

The finite-horizon optimal control problem with a non-convex chance constraint is formally stated as follows:

Problem 1: Finite-horizon optimal control with a non-convex chance constraint

For all $t = 0, 1, \cdots, T$,

$$\min_{U \in \mathbb{U}} \quad J(\bar{X}, U) \tag{1}$$

$$\text{s.t.} \quad x_{t+1} = Ax_t + Bu_t + w_t \tag{2}$$

$$w_t \sim \mathcal{N}(0, \Sigma_w) \tag{3}$$

$$x_0 \sim \mathcal{N}(\bar{\boldsymbol{x}}_0, \Sigma_{x,0}) \tag{4}$$

$$\Pr\left[\bigwedge_{i=1}^{N_i} \bigvee_{j=1}^{N_j(i)} h_{i,j}^T X \leq g_{i,j}\right] \geq 1 - \Delta, \tag{5}$$

where $\Delta \leq 0.5$ is the risk bound that is specified by the user. We assume that the cost function J is a linear function of the mean states $\bar{X}$ and control inputs U. Without loss of generality, we also assume that the non-convex chance constraint is in conjunctive normal form as (5), since any combination of logical conjunctions and disjunctions can be transformed to conjunctive normal form. The specification of chance constraints given in equation (5) requires that all N_i disjunctive clauses of state constraints must be satisfied with a probability $1 - \Delta$. The i'th disjunctive clause is composed of $N_j(i)$ state constraints. For example, the path planning problem in an environment with an obstacle illustrated in Fig. 1 requires the following chance constraint:

$$\Pr\left[\bigwedge_{t=1}^{T} \bigvee_{j=1}^{4} h_{t,j}^T x_t \leq g_{t,j}\right] \geq 1 - \Delta, \tag{6}$$

where T is the number of time steps in the planning horizon.

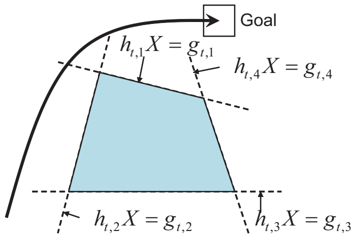


Fig. 1: Path planning with obstacles requires satisfying disjunctive clauses of linear state constraints (6).

B. Deterministic Approximation

Evaluating whether or not the chance constraint in Problem 1 has been satisfied requires computing an integral of a multivariate probability distribution over an arbitrary region, which can be computationally very costly. Our past work [1] has shown that a feasible, near-optimal solution to Problem 1 is obtained by solving the following problem, which does not involve random variables and probabilistic constraints:

Problem 2: Deterministic approximation of Problem 1

$$\min_{U\in\mathbb{U},\delta_{1:N_i}>0} \quad J(\bar{X},U) \tag{7}$$

$$\text{s.t.} \quad \bar{x}_{t+1} = A\bar{x}_t + Bu_t \tag{8}$$

$$\bigwedge_{i=1}^{N_i}\bigvee_{j=1}^{N_j(i)} h_{i,j}^T\bar{X} \le g_{i,j} - m_{i,j}(\delta_i) \tag{9}$$

$$\sum_{i=1}^{N}\delta_i \le \Delta, \tag{10}$$

where $-m_{i,j}(\cdot)$ is the inverse of the cumulative distribution function of the univariate Gaussian distribution with variance $h_i^T\Sigma_X h_i$ (note the negative sign):

$$m_{i,j}(\delta_i) = -\sqrt{2h_{i,j}^T\Sigma_X h_{i,j}}\ \mathrm{erf}^{-1}(2\delta_i - 1). \tag{11}$$

erf^{-1} is the inverse of the Gauss error function and Σ_X is the covariance matrix of X. Note that $m_{i,j}(\cdot)$ is a convex function for $\delta_i \le 0.5$. Since we assume that $\Delta \le 0.5$, its convexity is guaranteed from (10). Our approach of solving Problem 2 instead of Problem 1 is justified since the solution to Problem 2 is a feasible and near-optimal solution to Problem 1.

a) Feasibility: The following lemma guarantees that a solution to Problem 2 is a feasible solution to Problem 1:

Lemma 1: Satisfying the set of constraints (9-10) is a sufficient condition of the non-convex chance constraints given in (5).

Proof:

$$\begin{aligned}(5) \quad &\Longleftarrow \quad \bigwedge_{i=1}^{N_i}\Pr\left[\bigvee_{j=1}^{N_j(i)} h_{i,j}^T X \le g_{i,j}\right] \ge 1-\delta_i \wedge (10)\\ &\Longleftarrow \quad \bigwedge_{i=1}^{N_i}\bigvee_{j=1}^{N_j(i)}\Pr\left[h_{i,j}^T X \le g_{i,j}\right] \ge 1-\delta_i \wedge (10)\\ &\Longleftrightarrow \quad (9)\wedge(10).\end{aligned}$$

The first logical implication follows from the Boole's inequality. See [1] for a detailed proof. ■

b) Near optimality: Although the optimal solution of Problem 2 is not the optimal solution to Problem 1, our past work [1] showed that the suboptimality is significantly smaller than other bounding approaches such as [9]. This is explained by the fact that the probability of violating more than two disjunctive clauses of constraints is smaller than the probability of violating just one by orders of magnitude in many practical cases, where $\Delta \ll 1$.

Our new algorithm presented in Section IV, as well as our previous algorithm proposed by [1], optimally solves Problem 2 in order to obtain a feasible, near-optimal solution of Problem 1.

C. Disjunctive Convex Programming

Problem 2 is a disjunctive convex programming problem, which can be solved by a branch-and-bound algorithm. We proposed a bounding approach in [1], whereby the relaxed problems are constructed by removing all constraints below the corresponding disjunction. This approach was used by [16] and [17] for a different problem known as disjunctive linear programming.

At each loop of the branch-and-bound algorithm, we pick i, and assign an index to j. Let $\zeta(i)$ be the assignment to j for the i'th disjunctive clause. We set $\zeta(i) = \phi$ if an index is not assigned to the i'th clause. The branch-and-bound algorithm searches the optimal assignments $\zeta^\star$ by recursively solving the following convex optimization problem. We denote by $J^\star_{CCCO}(\zeta)$ its optimized cost function value of the convex optimization problem given an assignment ζ.

Problem 3: Convex Chance-Constrained Optimization

$$\begin{aligned} J^\star_{CCCO}(\zeta) \quad = \quad & \min_{U\in\mathbb{U},\delta_{1:N_i}>0} J(\bar{X},U)\\ \text{s.t.} \quad & \bar{x}_{t+1} = A\bar{x}_t + Bu_t\\ & \bigwedge_{i=1}^{N_i} h_{i,\zeta(i)}^T\bar{X} \le g_{i,\zeta(i)} - m'_{i,\zeta(i)}(\delta_i) \quad (12)\\ & \sum_{i=1}^{N}\delta_i \le \Delta. \quad (13)\end{aligned}$$

where

$$m'_{i,j}(\delta_i) := \begin{cases} -\infty & (\text{if } j = \phi)\\ m_{i,j}(\delta_i) & (\text{otherwise}).\end{cases}$$

By setting $m'_{i,j}$ to $-\infty$, the state constraint clauses with unassigned index j are virtually removed. Since the constraints are relaxed by removing unassigned clauses, $J^\star_{CCCO}(\zeta)$ with partially assigned ζ gives an lower bound to the one with fully assigned ζ.

D. Fixed Risk Relaxation

Since $m_{i,j}(\delta_i)$ is a non-linear function, Problem 3 is a non-linear convex programming problem, whose solution time is significantly greater than linear programming (LP) problems. We proposed a further relaxation of Problem 3 in [1], namely Fixed Risk Relaxation (FRR), which only has linear constraints. Since we assume that the cost function J is linear in this paper, FRR is an LP problem, which can be solved using any standard optimizer. The FRR of Problem 3 is obtained by fixing all the individual risk bounds δ_i to Δ, which is a constant:

Problem 4: Fixed Risk Relaxation of Problem 3

$$\begin{aligned} J^\star_{FRR}(\zeta) \quad &= \quad \min_{U\in\mathbb{U}} J(\bar{X}, U) \\ \text{s.t.} \quad & \quad \bar{x}_{t+1} = A\bar{x}_t + Bu_t \\ & \quad \bigwedge_{i=1}^{N_i} h^T_{i,\zeta(i)}\bar{X} \le g_{i,\zeta(i)} - m'_{i,\zeta(i)}(\Delta) \quad (14)\end{aligned}$$

Note that the non-linear constraint (12) is turned into a linear constraint (14) since the nonlinear term $m'_{i,\zeta(i)}$ becomes constant by fixing δ_i.

Lemma 2: Problem 4 gives a lower bound to Problem 3:

$$J^\star_{FRR}(\zeta) \le J^\star_{CCCO}(\zeta)$$

Proof: $m_{i,j}(\cdot)$ is a monotonically decreasing function. Since $\delta_i \le \Delta$, all individual chance constraints (14) of the Fixed Risk Relaxation are less stricter than (12) . ■

In the branch-and-bound algorithm, FRRs of the subproblems are solved to obtain lower bounds.

III. Regression-based Approximate LP Solver

Although each FRR is an LP problem, most of the computation time in the branch-and-bound algorithm is consumed by FRRs since they must be solved repeatedly for different ζ. This section proposes a novel regression-based approximate LP solver that has a *fixed* worst-case error bound for a given system of similar LP problems and choice of model complexity, so that the FRR cost $J^\star_{FRR}(\zeta)$ can be estimated efficiently.

A. Formulating as Regression Problem

A training set of N feasible LP problems $\{LP_1, \dots, LP_N\}$ and their optimum solutions $\{x^*_1, \dots, x^*_N\}$ is generated by solving all the FRRs present in the branch and bound trees of one or more optimal control problems using a standard optimizer, where any LP_k can be represented in the standard form as:

$$\min z_k = c_k^T x, \quad (15)$$

$$\text{s. t. } A_k x = b_k \quad (16)$$

$$x \ge 0 \quad (17)$$

Here, $c_k, x \in \Re^n$, $A_k \in \Re^{m_k,n}$, $b_k \in \Re^{m_k}$ $\forall k$ and $A_k \in A^s$, $b_k \in b^s$, $c_k \in c^s$, where the 3-tuple (A^s, b^s, c^s) represents the similar LP problem space. The decision variable vector x and the objective function vector c_k of all the LP problems have identical dimensionality n. However, no restrictions are imposed on the number of equality constraints m_k in the different LP problems.

We are interested in developing a regression model of this LP system in order to predict the solution of any new (test) LP problem whose parameters belong to the similar LP problem space. A separate regressor function is used for inferring each component of the vector x to avoid the computational complexity associated with learning multiple response variables simultaneously. The inter-dependence of the decision variables is captured by incorporating all the problem constraints in the predictor variable vector and also by modifying the loss function suitably (discussed at the end of this section).

We want to learn a set of n regressor functions

$$f_i : (A^s, b^s, c^s) \mapsto x_i, 1 \le i \le n \quad (18)$$

which are used to estimate the optimum x for any test LP problem. Any regression model requires a set of predictor variables (vectors denoted by v) and a set of response variables (scalars denoted by y). In our case, the optimum value of each LP decision variable x^*_i acts as the training set response variable for the corresponding function f_i. In order to generate the predictor variable vector for the training set, we first transform the system of equalities in (16) to a slightly different form as given by

$$A'_k x = b'_k \quad (19)$$

where $A'_k \in \Re^{m'_k,n}$ and $b'_k \in \Re^{m'_k}$ represent truncated forms of A_k and b_k that only contain the m'_k *active constraints* at the optimum solution x^*_k. By introducing a new matrix W_k, we can transform the above matrix constraint on x to a vector constraint that is given by

$$\hat{x} = (W_k^T A'_k)^{-1} W_k^T b'_k = d_k \quad (20)$$

where $W_k \in \Re^{m'_k,n}$ should be non-negative and the product matrix $W_k^T A'_k$ must be non-singular. The form of (20) is similar to a pseudoinverse solution of (19); W_k is used instead of A'_k to avoid the problem of singularity for non-unique x^*.

In order to construct the matrix W_k, let us represent it as $\{w_1, \dots, w_{m'_k}\}^T$ and A'_k as $\{a_1, \dots, a_{m'_k}\}^T$. We select each $w_i = e_{q_i}$, where $e_{q_i} \in \Re^n$ is a unitary vector and $q_i \in [1, n]$ denotes the position of the unity element such that $e^T_{q_i} a_i \ne 0 \forall i$. This ensures that $W_k^T A'_k$ is strongly non-singular if A'_k is of full rank, as the determinants of all of the principal submatrices are non-zero.

If multiple choices of q_i exist, then we choose q_i in such a manner that at least one non-zero entry exists in every column of W_k. Ties are broken randomly. This heuristic enables us to associate *relevant* constraints for every x_i (constraints where coefficients of x_i in A'_k are non-zero), weight the corresponding relevant constraint coefficients equally by unity, perform matrix division, and utilize the obtained value as the predictor variable vector value for the particular problem LP_k. If a column only contains zero elements, we eliminate this particular column from both W_k and A'_k and put $d_{i,k} = 0$, which takes care of the non-singularity problem for non-unique x^*. The overall choice of W_k is also useful in bounding the worst-case LP solution prediction errors, details of which for a slightly different formulation are given in [18].

Given this transformation (20), the common predictor variable vector for all the functions f in a problem LP_k is formed by augmenting c_k with d_k. Thus, effectively, we are learning $f_i : v \mapsto x_i, 1 \leq i \leq n$, where a specific training problem predictor variable vector instance is given by

$$v_k = [c_{1,k}, \dots, c_{n,k} \vdots d_{1,k}, \dots, d_{n,k}]^T \tag{21}$$

Clearly, the size of v_k is always equal to $p = 2n$ for any value of k. Thus, transformation (20) not only provides a compact way of encoding all the LP problem parameters A, b, and c, it also results in a constant predictor variable vector size that is independent of $m_k, 1 \leq k \leq N$. The set of active constraints is, however, unknown for the test LP problem. So, the step given by (19) is omitted and the entire matrix A and vector b is used for generating d. We also assume that the test LP problem parameters are such that all the components of the corresponding v vector lie within the range defined by the training set problems. If this is not the case, it is hypothesized that the LP problem is potentially infeasible and a standard optimizer is invoked to validate the hypothesis and obtain a feasible solution if necessary.

B. Developing Boosting Tree Models

We have developed a modified version of the standard boosting tree algorithm given in [19] to learn each regressor function f_i. The multivariate regressor function is represented as a sum of trees $T(v; \Theta_m)$, each of which is given by

$$T(v; \Theta_m) = \sum_{j=1}^{Q} \gamma_{jm} I(v \in R_{jm}) \tag{22}$$

where $\Theta_m = \{R_{jm}, \gamma_{jm}\}$ encodes the parameters of the m-th regression tree having terminal regions $R_{jm}, j = 1, \dots, Q$, and the indicator function I is defined as

$$I(v \in R_{jm}) = \begin{cases} 1 & \text{if } v \in R_{jm}, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

It can be seen from (22) that the response variable (y) is modeled as a constant within every tree region R_{jm}. Using the additive form, the overall boosted tree can then be written as the sum of M regression trees

$$f_M(v) = \sum_{m=1}^{M} T(v; \Theta_m) \tag{24}$$

Two child regions R_1 and R_2 are created at every internal parent node of a tree by selecting a splitting variable $o, 1 \leq o \leq p$ and a split value s that define a pair of half-planes

$$R_1 = \{v \mid v^o \leq s\}, R_2 = \{v \mid v^o > s\} \tag{25}$$

where v^o represents the the oth component of the vector v. We select o and s using a greedy stategy to minimize the residual sum of squares error that solves

$$\min_{o,s}[\min_{\gamma_1} \sum_{v_k \in R_1} (r_k - \gamma_1)^2 + \min_{\gamma_2} \sum_{v_k \in R_2} (r_k - \gamma_2)^2] \tag{26}$$

where v_k is the predictor variable vector corresponding to LP_k and r_k is the target value in each tree region. r_k is chosen as the negative gradient of the loss functional that is given by $-\partial L(y_k, f(v_k))/\partial f(v_k)$. Here, $L(y, f(v))$ denotes any of the standard loss functions, such as L_2 or the squared-error loss, L_1 or the absolute-error loss, or the more robust Huber loss L_H [19]. For any choice of o and s, the inner minimization in (25) is solved by

$$\hat{\gamma}_1 = \frac{\sum_{v_k \in R_1} r_k}{N_1}, \hat{\gamma}_2 = \frac{\sum_{v_k \in R_2} r_k}{N_2} \tag{27}$$

where N_1 and N_2 denote the number of training data points in R_1 and R_2 respectively. Each regressor tree is grown by binary partitioning until the number of leaf nodes equals or exceeds the fixed size Q that is chosen *a priori*. If required, the tree is then pruned using the technique of cost-complexity pruning described in [19] to reduce the number of leaf nodes to Q.

In order to prevent overfitting, we consider a slightly modified version of (22) in an iterative form as

$$f_m(v) = f_{m-1}(v) + \nu \sum_{j=1}^{Q} \gamma_{jm} I(v \in R_{jm}), m = 1, \dots, M \tag{28}$$

where $\nu \in (0, 1)$ is the shrinkage parameter that controls the learning rate; the modeling variable is given by

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{v_k \in R_{jm}} L(y_k, f_{m-1}(v_k) + \gamma) \tag{29}$$

We adopt a modified form of the Huber loss function that heavily penalizes selection of any γ_{jm} that lies outside the common feasible region of all the given LP problems for which v lie in R_{jm}. We refer to this as the *penalization loss* L_p and represent it as

$$L_p = L_H + h\gamma'_{jm} \tag{30}$$

Here, h is a very large positive number and $\gamma'_{jm} = \{y : y \notin P_c\}$, where P_c is the common feasible region of all the LPs whose $v \in R_{jm}$. Although this modification cannot guarantee generation of feasible solution for an arbitrary test LP problem whose feasible solution exists, it significantly increases the possibility of doing so (shown in Section V). Infeasibilities are detected based on the significantly higher predicted FRR cost values and are handled by pruning the corresponding branch and bound nodes.

The geometric interpretation of this algorithm is that it iteratively partitions the predictor variable vector space into a set of regions (stored as tree leaf nodes), each of which is an axis-aligned hyperbox. Inference is simply performed by identifying the region corresponding to the location of the point specified by the test LP problem predictor vector and selecting the fixed response variable encoded in the identified region. Thus, every decision variable vector component can be obtained efficiently, enabling us to estimate the objective function quickly by computing the dot product given in (15).

IV. Overall Algorithm

This section presents the new algorithm that solves Problem 2 optimally and efficiently. The key idea is to solve the FRR (Problem 4) by the regression-based approximate LP solver described in Section III in order to enhance the computational speed, while solving the convex chance-constrained optimization (Problem 3) at the leaf nodes of the branch-and-bound tree exactly in order to obtain the strictly optimal solution to Problem 2.

The pseudo code is outlined in Algorithm 1. The algorithm is initialized by an empty assignment (Line 3). For each assignment ζ, the FRR of the corresponding subproblem is solved by the approximate LP solver in order to obtain the estimated lower bound of the optimal cost function value (Line 7). The approximate LP solver may incorrectly judge that the problem is infeasible, although such a case occurs infrequently. In that case, to prevent incorrect pruning of feasible solutions, the algorithm solves the FRR without approximation using a standard optimizer. As mentioned earlier in Section III, a very important property of our approximate LP solver is that a worst-case bound on the prediction error is known *a priori*, which is constant for a given training data set and regression model and is applicable for any test FRR that belongs to the same similar LP problem space. In other words, there exists a positive finite real number ϵ such that:

$$\left|\hat{J}_{FRR}(\zeta) - J^{\star}_{FRR}(\zeta)\right| \leq \epsilon,$$

where $\hat{J}_{FRR}(\zeta)$ is the estimated optimal cost of the FRR given an assignment ζ. Therefore, we can guarantee that the optimal solution is found by the branch-and-bound algorithm by pruning the branch only if the estimated cost lower bound $\hat{J}_{FRR}(\zeta)$ exceeds the incumbent by more than ϵ (Line 8). Otherwise, the branch is expanded by invoking the function branchAndBound recursively (Line 14). If a branch is expanded to the leaf (i.e., $i = N_i$) without being pruned, the convex chance-constraint optimization problem (Problem 3) is solved exactly (Line 17). If its optimal cost function value is less than the incumbent, the incumbent and the optimal solution $\boldsymbol{U}^{\star}$ are updated (Lines 19 and 20). Although Algorithm 1 uses the approximate LP solver to estimate the lower bounds on the cost in the branch-and-bound algorithm, it results in a *globally optimal* solution of Problem 2 since the solution $\boldsymbol{U}^{\star}$ is always obtained by solving Problem 3 exactly.

Algorithm 1 Non-convex Chance Constrained Optimal Control with Regression-based LP Solver

```
1:  global $Incumbent, \boldsymbol{U}^\star$
2:  $Incumbent \leftarrow \infty$
3:  $\zeta(i) \leftarrow \phi$ for $i = 1 \cdots N_i$
4:  branchAndBound$(1, \zeta)$
5:  return $\boldsymbol{U}^\star$

function branchAndBound$(i, \zeta)$
6:  global $Incumbent, \boldsymbol{U}^\star$
7:  Solve Problem 4 approximately by the regression-based
    LP solver
8:  if $J^\star_{FRR}(\zeta) \geq Incumbent + \epsilon$ then
9:      //Prune this branch; Do nothing.
10: else
11:     if $i < N_i$ then
12:         for $j = 1 \cdots N_j(i)$ do
13:             $\zeta(i) \leftarrow j$
14:             branchAndBound$(i + 1, \zeta)$      //Expand
15:         end for
16:     else
17:         Solve Problem 3
18:         if $J^\star_{CCCO}(\zeta) < Incumbent$ then
19:             $Incumbent \leftarrow J^\star_{CCCO}(\zeta)$
20:             $\boldsymbol{U}^\star \leftarrow \boldsymbol{U}$ //Update the optimal solution
21:         end if
22:     end if
23: end if
```

V. Results

All the results presented in this section are obtained on an Intel Core2 Duo CPU, having 2.00 GHz processor speed and 2.9 GB of RAM, in Ubuntu 9.0.4 OS, using C++ as the programming language and IBM ILOG CPLEX Optimization Solver Academic Edition version 12.2 as the optimization solver. The total number of regression trees, M, is always chosen as 1000, the number of leaf nodes in any tree, Q, as 16, and the shrinkage factor, ν, as 0.1.

We tested our approach on 2D path planning problems under Gaussian uncertainty with a single chance constraint, involving obstacle avoidance and finding paths through waypoints at desired time instants. A discrete-time, point-mass dynamics model is used for the vehicle, which always starts from [1,1] and heads to a rectangular goal location with center at [12,12]. The system matrices are given by

$$A = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and

$$B = \begin{pmatrix} \Delta t^2/m & 0 \\ 0 & \Delta t^2/m \\ \Delta t/m & 0 \\ 0 & \Delta t/m \end{pmatrix}$$

TABLE I: Performance evaluation on different planning problem scenarios. The proposed algorithm uses regression for approximately solving FRRs whereas the previous algorithm solves FRRs exactly using CPLEX optimizer. All the reported data are for average values; standard deviation values are not presented as they are of the order of 0.1% of the average.

Performance metric	Scenario number			
	1	2	3	4
Previous algorithm comp. time (s)	135.21	219.76	79.99	80.15
Proposed algorithm comp. time (s)	7.51	8.14	6.15	5.73
Speed-up	18X	27X	13X	14X
Cost (using both algorithms)	4.23	5.86	3.45	3.60
Theoretical LP solution error (%)	4.2	4.1	3.5	3.6
Observed LP solution error (%)	3.3	3.4	2.7	2.7
Incorrect feasible predictions (%)	0.5	0.7	0.4	0.5

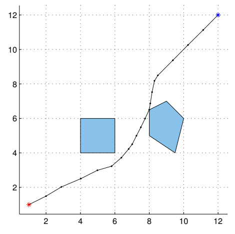
where m is the mass of the vehicle. The risk bound, Δ, is always set to 0.01, the time interval, Δt, to 0.5 and m to 1.0. The control inputs (forces along the X and Y axes) are bounded by $u_{min} = [-5, -5]$ and $u_{max} = [5, 5]$. The disturbance covariance matrix is given by

$$\Sigma_w = \begin{pmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where $\sigma_x, \sigma y \in [0.1, 0.001]$. The cost is the total control effort during the planning horizon T as given by $J(\overline{X}, U) = \sum_{t=1}^{T}(\mid u_{x,t} \mid + \mid u_{y,t} \mid)$. Although this cost function is not linear, we linearize it by introducing slack variables.

Table I enumerates the performance of our algorithm for four different planning scenarios. Scenario 1 deals with planning in an environment consisting of two obstacles, scenario 2 with avoiding two obstacles and passing through two waypoints at specified time instants, scenario 3 with different levels of disturbances (Gaussian distribution variance) in the vehicle location, and scenario 4 with varying maximum limits on the vehicle acceleration respectively. One obstacle and one waypoint are used in the last two scenarios. All the obstacles and waypoints are rectangular except in the case of scenario 1. T is always selected as 20 in all the scenarios; σ_x and σ_y are both chosen to be 0.01 in all the scenarios excepting 3 and u_{min} and u_{max} are taken to be -2.5 and 2.5 respectively in the first three scenarios.

Optimum solutions of feasible FRRs arising in 16 different problem instances are used as the training data set and the FRRs occurring in 4 new instances are utilized for testing purposes in each of the four scenarios. The locations of the obstacles and the waypoints are varied randomly in the first two scenarios, whereas the values of the location disturbance variance and the maximum acceleration are altered randomly in the next two scenarios, keeping the obstacle and waypoint locations fixed. The generated trajectories for one of the training and test problem instances for scenario 1 are shown in Fig. 2.


(a) Sample training problem

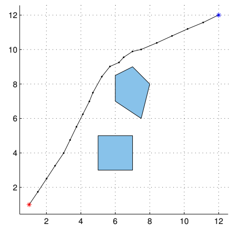

(b) Test problem

Fig. 2: Generated trajectories for planning with obstacle avoidance using non-convex chance constrained optimal control algorithm. The obstacles are displaced in the test problem from their locations in the sample training problem, showing that regression models learnt from different but similar problems can be utilized to compute the optimum solutions for new problems.

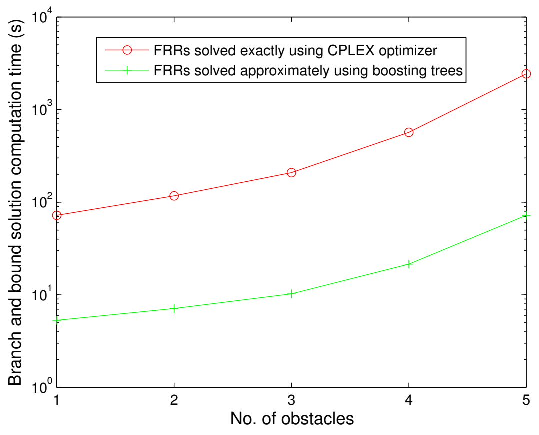


Fig. 3: Computation time comparison for varying number of obstacles. Note that the plot is in semi-log scale and error bars are not shown as they are negligibly small.

Table I shows that significant speed-up is obtained by using the proposed algorithm as compared to the previous one for all the different scenarios. At the same time, identical cost (objective) function values are returned by both the algorithms. This fact clearly indicates that the optimality of the overall branch and bound algorithm is strictly preserved, even though its LP subproblems are solved approximately. The average estimation errors of the objective values of the LP subproblems lie between (2.5-3.5)%, which are always within the theoretically-predicted values. This shows the effectiveness of using the theoretical worst-case bound during pruning. The number of incorrectly predicted feasible solutions for infeasible FRRs is also quite small, indicating the fact that only a few branches are not pruned when they should have been.

Figures 3 and 4 show the enhanced effect of computational speed-up for greater number of obstacles and longer planning time horizons respectively. The values of σ_x, σ_y, u_{min},

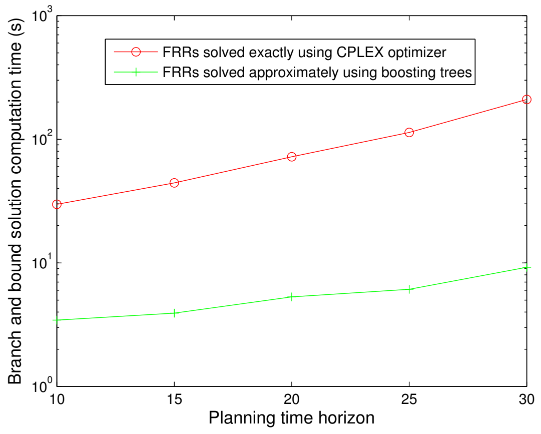


Fig. 4: Computation time comparison for varying planning time horizons. Note that the plot is in semi-log scale and error bars are not shown as they are negligibly small.

u_{max}, as well as the number of training and test problem instances are identical to those used for the scenarios in Table I. T is chosen as 20 for all the problem instances in Fig. 3, only one obstacle is present for the problem instances in Fig. 4, no waypoints are present, and all the obstacles are rectangular.

It may be noted here that although solving FRRs approximately using regression does not prevent the exponential growth in computational time, it does reduce the *rate* of exponential growth. This happens because the regression inference time remains the same (for identical model complexity) at the internal nodes of the branch and bound trees in all the problem instances, unlike the CPLEX optimizer, whose running time increases significantly with the number of constraints and decision variables. The plots are not extended any further as the trends do not change and the computation time of the proposed algorithm also becomes significantly more than a few seconds, thereby rendering the approach less useful for practical applications. Again, it should be noted here that even though individual FRR problems are solved approximately, we ensure that we obtain an optimal solution to the overall control problem by conservatively pruning branches and invoking the exact convex optimization solver at the leaf nodes.

VI. Conclusions

We present a novel regression-based approximation technique for solving Fixed Risk Relaxation LP problems that occur in the tree nodes of a branch and bound-based non-convex, chance constrained finite horizon optimal control algorithm. Similarity of the solutions and the objective function and constraint coefficients of the LP problems are exploited to develop boosting tree models over which fast inference can be performed. Matrix transformations are applied to construct predictor variable vectors with desirable properties that enable us to come up with absolute worst-case bounds on the solution prediction errors. Such errors bounds are used to prune branches conservatively and exact convex optimization is used at the leaf nodes of the branch and bound trees to obtain optimal solutions. Empirical results demonstrate significant computational speed-up over our previous algorithm that relied on standard optimizers to solve the individual FRR problems. Future work would include validating this approach on real robotic hardware platforms.

Acknowledgments

We would like to thank Michael Kerstetter, Scott Smith, Ronald Provine, and the Boeing Company for their support.

References

[1] M. Ono, L. Blackmore, and B. C. Williams, "Chance constrained finite horizon optimal control with nonconvex constraints," in *Proceedings of the American Control Conference*, 2010.

[2] Y. Kuwata, A. Richards, and J. How, "Robust receding horizon control using generalized constraint tightening," in *Proceedings of the American Control Conference*, 2007.

[3] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Management Science*, vol. 6, pp. 73–79, 1959.

[4] M. Ono and B. C. Williams, "An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008.

[5] P. Li, M. Wendt, and G. Wozny, "A probabilistically constrained model predictive controller," *Automatica*, vol. 38, pp. 1171–1176, 2002.

[6] A. Nemirovski and A. Shapiro, "Convex approximations of chance constrained programs," *SIAM Journal on Optimization*, vol. 17, pp. 969–996, 2006.

[7] A. Prékopa, "The use of discrete moment bounds in probabilistic constrained stochastic programming models," *Annals of Operations Research*, vol. 85, pp. 21–38, 1999.

[8] L. Blackmore, "A probabilistic particle control approach to optimal, robust predictive control," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2006.

[9] L. Blackmore, H. Li, and B. C. Williams, "A probabilistic approach to optimal robust path planning with obstacles," in *Proceedings of the American Control Conference*, 2006.

[10] R. S. Sutton and A. G. Burto, *Reinforcement Learning: An Introduction*, 1st ed. The MIT Press, 1998.

[11] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to acrobatic helicopter flight," in *Advances in Neural Processing Information Systems*, 2007.

[12] W. Zhang and T. G. Dietterich, "High performance job-shop scheduling with a time-delay TD(λ) network," in *Advances in Neural Processing Information Systems*, 1996.

[13] D. Vladušič, A. Černivec, and B. Slivnik, "Improving job scheduling in grid environments with use of simple machine learning methods," in *Proceedings of the 6th International Conference on Information Technology: New Generations*, 2009.

[14] J. E. Bell and P. McMullen, "Ant colony optimization for the vehicle routing problem," *Advanced Engineering Informatics*, vol. 18, no. 1, pp. 41–48, 2004.

[15] R. Nallusamy, K. Duraiswamy, R. Dhanalakshmi, and P. Parthiban, "Optimization of multiple vehicle routing problems using approximation algorithms," *International Journal of Engineering Science and Technology*, vol. 1, no. 3, pp. 129–135, 2009.

[16] E. Balas, "Disjunctive programming," *Annals of Discrete Mathematics*, 1979.

[17] H. Li and B. C. Williams, "Generalized conflict learning for hybrid discrete linear optimization," in *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming*, 2005.

[18] A. G. Banerjee and N. Roy, "Learning Solutions of Similar Linear Programming Problems using Boosting Trees," Massachusetts Institute of Technology, CSAIL Technical Report MIT-CSAIL-TR-2010-045, 2010, available at http://dspace.mit.edu/handle/1721.1/58609.

[19] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2008.