Automatic Testing and MiniMax Optimization of System Parameters for Best Worst-Case Performance

Kim Peter Wabersich[1], Marc Toussaint[2]

***Abstract*— Robotic systems typically have numerous parameters, e.g. the choice of planning algorithm, real-valued parameters of motion and vision modules, and control parameters. We consider the problem of optimizing these parameters for best worst-case performance over a range of environments. To this end we first propose to evaluate system parameters by adversarially optimizing over environment parameters to find particularly hard environments. This is then nested in a game-theoretic minimax optimization setting, where an outer-loop aims to find best worst-case system parameters. For both optimization levels we use Bayesian global optimization (GP-UCB) which provides the necessary confidence bounds to handle the stochasticity of the performance. We compare our method (Nested Minimax) with an existing relaxation method we adapted to become applicable in our setting. By construction our approach provides more robustness to performance stochasticity. We demonstrate the method for planning algorithm selection on a pick'n'place application and for control parameter optimization on a triple inverted pendulum for robustness to adversarial perturbations.**

I. Introduction

Robotic research has made great steps torwards intelligent robots in complex environments. At the same time it became more difficult to efficiently test such autonomous systems and measure their performance in simulation or real-world experiments: The main challenge is the high-dimensionality of the space of potential environment configurations over which should be tested. Ideally, testing should also aim to unveil the worst case behavior, that is, find environmental configurations for which the system performs particularly poor.

The problem of measuring systems performance was e.g. addressed for path planning algorithms in form of a generic testing infrastructure inside the robot operating system (ROS) [1]. But this approach only handles a finite, predefined number of test-configurations. There is no framework to consider an infinite set of environment configurations or an adversarial automatic testing method that seeks to find worst cases. To this end, in this paper we first propose an automatic, active learning-based test framework using gaussian processes in combination with the upper confidence bound (GP-UCB) [2]. This method aims to globally optimize environment configuration parameters to be adversarial, that is, particularly difficult for the system.

Second, given an efficient method for automatically testing a system, we can now *optimize system parameters* (e.g., real-valued parameters of a path planner or of an object tracker, categorial parameters for the choice of planning or perception algorithms). We will again tackle this by overlaying another GP-UCB optimization algorithm. In summary, this paper proposes a coherent framework for optimizing system parameters w.r.t. worst-case performance, where worst-case performance is evaluated by an adversarial optimization over environment parameters.

[1]wabersich@kimpeter.de
[2]marc.toussaint@informatik.uni-stuttgart.de

A. Game Theoretic Problem Formulation

The fundamental idea can be understood as a game theoretic minimax optimization approach for autonomous systems. We consider two agents playing a game against each other. The "Min-Agent" aims to minimize costs, which corresponds to good robot system performance; whereas the "Max-Agent" aims to maximize costs and corresponds to an adversarial environment—represented by the automatic testing framework that aims to unveil the worst-case performance of the robot system (see Fig. 1). Formally, let $\mathcal{D}_E$ be the set of modifiable continuous or discrete environment parameters, e.g. position and orientation parameters of static objects or dynamic disturbance parameters. Let $\mathcal{D}_S$ be the set of modifiable parameters of the intelligent system, e.g. execution speed, controller parameters or discrete decision variables like planning-algorithms or higher level configurations. Moreover, assume a noisy black-box function

$$f(\boldsymbol{x}_S, \boldsymbol{x}_E) = f^*(\boldsymbol{x}_S, \boldsymbol{x}_E) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma_s^2) \qquad (1)$$

that quantifies the success (actually cost) of completing the task in the given environment $\boldsymbol{x}_E \in \mathcal{D}_E$ with system $\boldsymbol{x}_S \in \mathcal{D}_S$. We formulate our minimax problem as

$$\boldsymbol{x}_S^* = \underset{\boldsymbol{x}_S \in \mathcal{D}_S}{\operatorname{argmin}} \max_{\boldsymbol{x}_E \in \mathcal{D}_E} f(\boldsymbol{x}_S, \boldsymbol{x}_E) \qquad (2)$$

where $\boldsymbol{x}_S^*$ describes the optimal system parameters.

B. Main Contributions

Our contributions are 1) a novel, nested minimax optimization method that exploits the smoothed mean estimator and confidence bounds (for stopping criteria) provided by Bayesian global optimization methods; 2) an adaptation of an existing relaxation method (introduced in the next section) to make it applicable in our setting and comparable; and 3) the (to our knowledge) first-time application of rigorous minimax optimization to robotic applications:
a) The minimax optimization of the choice of a path planning algorithm.
b) The minimax optimization of parameters of a linear quadratic regulator (LQR) to robustly control a highly nonlinear system (triple inverted pendulum).

Published in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), DOI: 10.1109/IROS.2015.7354161.

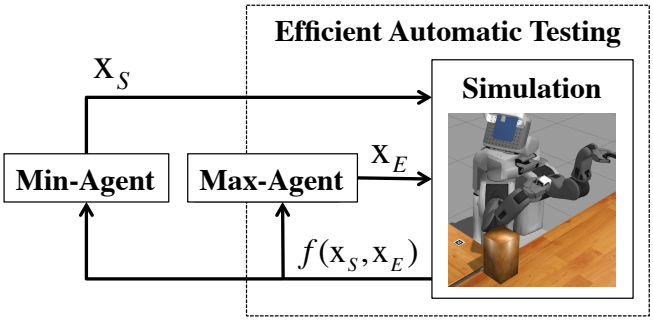


Fig. 1: Basic optimization concept: The Max-Agent provides a worst-case system evaluation by optimizing adversarial environment parameters, while the Min-Agent is optimizing system parameters to minimize the worst case costs.

After revisiting related work we will give an introduction to the GP-UCB algorithm as a central ingredient to our methods. We then describe two methods for solving problem (2) and shortly compare those. The methods are then evaluated in the experimental section.

II. Related Work

A. Applications

As mentioned before, [1] presents a generic test system for planning algorithms. This approach takes only a few predefined scenes into account for the performance test.

Regarding optimization of intelligent or autonomous systems as a whole there are several classical approaches like [3]. In [4] they specifically investigate a bayesian based optimization of a bipedal walking robot. There are very few optimization approaches that take a changing environment configuration into account. In [5] a minimax optimization method is developed for robot design tasks. However, this approach considers only an analytic performance model instead of take costly and stochastic simulations or experiments into account.

In contrast to these existing methods we introduce an efficient automatic testing framework which leverages Bayesian optimization to find adversarial environment parameters and thereby evaluates the worst-case performance of the system. This measure is then used for minimax optimization of system parameters.

B. Efficient Minimax Optimization

The main challenge of problem (2) on the one hand is the very costly (time consuming) evaluation of the cost function f. On the other hand there is a reasonable noise in evaluating f. While there are many approaches to tackle problem (2) (e.g. [6], [7] and [8]), there are only two approches so far ([9] and [10] respectively [11]), that would provide the needed evaluation efficiency by approximating the cost-function throughout kriging in combination with Expected Improvement (EI). However, [9] does not support a noisy cost function. The recent methods [10] respectively [11] have so far only been tested on toy problems independent of robotics.

We will compare our approach with a modification of [10], [11] (combining it with GP-UCB). A core difference is that our method consistently handles noise when evaluating system parameters (by computing a mean estimator with confidence bound in the adversarial optimization) while the other takes the max over a finite set of noisy evaluations.

III. Global Optimization: Multi-armed bandits and Upper Confidence Bound algorithm

In order to solve problem (2) we first need an efficient way to solve general stochastic optimization problems

$$\boldsymbol{x}_{opt} = \underset{\boldsymbol{x}\in\mathcal{D}}{\operatorname{argmin}} f(\boldsymbol{x}) \tag{3}$$

with an objective function

$$f(\boldsymbol{x}_{opt}) = f^*(\boldsymbol{x}_{opt}) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma_s^2) \tag{4}$$

and a feasible set

$$\mathcal{D} = \{\boldsymbol{x} \in \mathbb{R}^d \mid x_i \in [a_i,\ b_i] \subset \mathbb{R},\ a_i < b_i,\ i = 1, 2, .., d\} \tag{5}$$

which is a general hypercube. Because in our case f is very costly to sample and tainted with noise ϵ, we treat (3) as a multiarmed bandit problem. Therefore at least some assumptions on the "black-box" cost functions have to be made. Namely we assume (4), i.e. the infinitely many armed bandit is sampled from a Gaussian process (GP, $\mathcal{GP}(0, k)$, see [12]) with zero mean $\mu = 0$ and kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ that defines the correlation between two points $\boldsymbol{x}$ and $\boldsymbol{x}'$ within the feasible set $\mathcal{D}$. This is not the case in practice but a fairly good approximation and formally easy to handle. That stochastic process assumption about f is the basis for recent global black-box optimization algorithms like the very promising gaussian process upper confidence bound (GP-UCB) algorithm [2]. Using noisy samples $\boldsymbol{y}_T = [y_1, .., y_T]^T$ at points $X_T = [\boldsymbol{x}_1, .., \boldsymbol{x}_T]^T$ where $y_t = f(\boldsymbol{x}_t) = f^*(\boldsymbol{x}_t) + \epsilon$ the GP $\mathcal{GP}(0, k)$ posterior over f is given by

$$\mu_T(\boldsymbol{x}) = \boldsymbol{k}_T(\boldsymbol{x})^T {K_T^*}^{-1} \boldsymbol{y}_T \tag{6}$$

$$k_T(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}_T(\boldsymbol{x})^T {K_T^*}^{-1} \boldsymbol{k}_T(\boldsymbol{x}') \tag{7}$$

$$\sigma_T^2(\boldsymbol{x}) = k_T(\boldsymbol{x}, \boldsymbol{x}). \tag{8}$$

$\boldsymbol{k}_T(\boldsymbol{x}) = [k(\boldsymbol{x}_1, \boldsymbol{x}), .., k(\boldsymbol{x}_T, \boldsymbol{x})]^T$, $k_T(\boldsymbol{x}, \boldsymbol{x}')$ is the posterior covariance and $K_T^* = K_T + \sigma_s^2 I$ is the positive definite kernel or covariance matrix $[k(\boldsymbol{x}, \boldsymbol{x}')]_{\boldsymbol{x},\boldsymbol{x}' \in X_T}$.

A. Exploration vs. Exploitation

A general strategy how to sample f in order to learn about the optimium of f can be expressed as

$$\boldsymbol{x}_t^* = \underset{\boldsymbol{x}\in\mathcal{D}}{\operatorname{argmin}}\, \mu_{t-1}(\boldsymbol{x}) - \sqrt{\beta_t}\sigma_{t-1}(\boldsymbol{x}) \tag{9}$$

where β_t defines an exploration vs. exploitation trade-off strategy between averagely good regions or bandits (small $\mu_{t-1}(\boldsymbol{x})$) and not well discovered regions (large "uncertainty" $\sigma(\boldsymbol{x})$). As feasible set we assume a normalized hypercube

$$\mathcal{D} = \{\boldsymbol{x} \in \mathbb{R}^d \mid x_i \in [0,\ 1] \subset \mathbb{R}, i = 1, 2, .., d\}. \tag{10}$$

Algorithm 1 Global Optimization using Gaussian Processes in the bandit-setting and the Upper Confidence Bound

procedure GP-UCB(input space: $\mathcal{D}$, function f, $\mathcal{GP}$-Prior $\{\sigma_s, k\}$, max. error ε_{err}, exploration policy β_t)
  $\mathcal{D}^* \leftarrow normalize(\mathcal{D})$ ▷ *using (11).*
  initialize $\boldsymbol{X}_0^* = \{\boldsymbol{x}_0\}$, $\boldsymbol{x}_0 \in \mathcal{D}^*$ randomly
  initialize $\boldsymbol{Y}_0 = \boldsymbol{Y}_0^* = [f(\boldsymbol{x}_0)]$
  $t \leftarrow 1$, $\varepsilon_{t-1} \leftarrow 0$, $\varepsilon_t \leftarrow \infty$,
  while $|\varepsilon_t - \varepsilon_{t-1}| > \varepsilon_{err}$ **do**
    $\boldsymbol{x}_t^* = \text{argmin}_{\boldsymbol{x}^* \in \mathcal{D}^*} \mu_{t-1}(\boldsymbol{x}^*) + \sqrt{\beta_t}\sigma_{t-1}^*(\boldsymbol{x}^*)$
    $\boldsymbol{X}_t^* \leftarrow$ append $\boldsymbol{x}_t^*$ to $\boldsymbol{X}_{t-1}^*$
    $\boldsymbol{x}_t \leftarrow denormalize(\boldsymbol{x}_t^*)$ ▷ *using (12).*
    $\boldsymbol{Y}_t \leftarrow$ append $y_t = f(\boldsymbol{x}_t)$ to $\boldsymbol{Y}_{t-1}$
    $\boldsymbol{Y}_t^* = \boldsymbol{Y}_t - \frac{1}{T}\sum_{i=1}^{T} y_i$ ▷ centering mean.
    update $\mathcal{GP}$ with $\boldsymbol{X}_t^*$ and $\boldsymbol{Y}_t^*$
    $t = t + 1$
    calculate ε_t ▷ *stopping criteria, eq.* (18).
  end while
  $\boldsymbol{x}_{opt}^* \leftarrow \text{argmin}_{\boldsymbol{x}^* \in \mathcal{D}^*} \mu_T(\boldsymbol{x})$
  $y_{\mathcal{GP}} \leftarrow \mu_T(\boldsymbol{x}_{opt})$
  return $[denormalize(\boldsymbol{x}_{opt}^*), y_{\mathcal{GP}} + \mathbb{E}[\boldsymbol{Y}_T]]$
end procedure

This is crucial when it comes to choosing the GP-Prior in the next section - otherwise we would have to distinguish between each input dimension. To normalize a general hypercube (5) one can apply the affine transformation

$$normalize(\mathcal{D}) = \{\boldsymbol{x}^* \mid \boldsymbol{x} \in \mathcal{D}, x_i^* = (a_i - b_i)^{-1}(a_i - x_i), \\ i = 1, 2, .., \dim(\mathcal{D})\} \tag{11}$$

with its inverse

$$denormalize(\mathcal{D}^*) = \{\boldsymbol{x} \mid \boldsymbol{x}^* \in \mathcal{D}^*, x_i = (b_i - a_i)x_i^* + a_i, \\ i = 1, 2, .., \dim(\mathcal{D}^*)\} \tag{12}$$

respectively. In [2] a β_t according to $\mathcal{D}$ is presented which is proven to provide regret bounds[1] in our settings. Therefore:

1) Finite feasible set: If ($|\mathcal{D}| < \infty$) choose

$$\beta_t = 2\log\left(\frac{|\mathcal{D}|t^2\pi^2}{30}\right). \tag{13}$$

2) Infinite feasible set: If ($|\mathcal{D}| \not< \infty$) choose

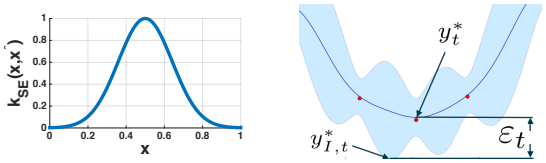
$$\beta_t = 2\log\left(\frac{t^2 2\pi^2}{15}\right) + 2d\log\left(t^2 d\sqrt{\log(20d)}\right). \tag{14}$$

B. Choosing the GP-Prior

The sampling variance σ_s^2 can be chosen appropriately to (4). For practical robustness we use the simple to interpret squared exponential (SE) kernel [12]. It is defined as

$$k_{SE}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|}{2l_{SE}}\right)^p\right) \tag{15}$$

[1]Cumulative regret: $R_T = \sum_{t_1}^{T} r_t$, $r_t = f(\boldsymbol{x}^*) - f(\boldsymbol{x})$, where $\boldsymbol{x}^* = \min_{\boldsymbol{x}} f(\boldsymbol{x})$.



(a) Correlation of x and $x^* = 0.5$ for SE kernel (15) with $\sigma_f^2 = 1$.

(b) Illustration of stopping criterion (18).

Fig. 2: Visualization of GP-UCB configuration.

where we fix hyperparameters $l_{SE} = 0.1$ and $p = 2$ for aggresive interpolation, see Fig. 2 (a). Note that because of transforming the input space (11), (12) we implicitly assume same lengthscales in every dimension. Since the observation variance σ_f^2 is crucial we estimate it from the collected samples using

$$\sigma_f^2 \approx \hat{\sigma}_{f,T}^2 = \frac{1}{T-1}\boldsymbol{y}^T K^{-1}\boldsymbol{y}. \tag{16}$$

Thus by factoring out $\hat{\sigma}_{f,T}^2$ we modify the variance posterior prediction (8) to

$$\sigma_T^{*2}(\boldsymbol{x}) = \hat{\sigma}_{f,T}^2 k_T(\boldsymbol{x}, \boldsymbol{x}). \tag{17}$$

C. Stopping criterion

Because in [2] there is no suggestion for a termination condition we propose

$$\varepsilon_t = \underbrace{\left(\min_{\boldsymbol{x} \in \mathcal{D}} [\mu_{t-1}(\boldsymbol{x}) - \sigma_{t-1}(\boldsymbol{x})]\right)}_{=: y_{I,t}^*} - \underbrace{\left(\min_{\boldsymbol{x} \in \mathcal{D}} \mu_{t-1}(\boldsymbol{x})\right)}_{=: y_t^*} \tag{18}$$

as an intuitive and expressive stopping criterion. The term $y_{I,t}^*$ equals exactly the heuristic (9) by choosing $\beta_t = 1$ uniformly to represent the current optimum plus a potential improvement of one times the standard deviation. The second term y_t^* represents the mean estimated optimum at round $t-1$. The difference in (18) finally quantifies the potential improvement to the currently estimated optimum. We stop sampling further points, if

$$|\varepsilon_t - \varepsilon_{t-1}| < \varepsilon_{rel}, \tag{19}$$

i.e. the optimistic heuristic value for $\beta = 1$ stagnates and we expect no more dramatic changes by further sampling, see Fig. 2 (b). The modified GP-UCB algorithm is shown in algorithm 1. Note that we use $\sigma_t^*(\boldsymbol{x})$ from eq. (17) instead of $\sigma_t(\boldsymbol{x})$ (8). A working example of the resulting GP ($\mathcal{GP}(0, k)$) is presented in section V.

IV. Minimax Optimization Approach for Autonomous Systems

In this section the main contribution of this paper will be explained. First we introduce a naive minimax optimization approach which directly takes the feedback of the efficient testing algorithm into account. Furthermore we apply the methods described in [10] and [11] to solve the same problem. Then we compare the two approaches with respect to their efficiency.

Algorithm 2 Nested Minimax using the UCB

1: **procedure** NESTED-MINIMAX($\mathcal{D}_S$, $\mathcal{D}_E$, f, $\{\sigma, k\}_S$, $\{\sigma, k\}_E$, ε^S_{err}, ε^E_{err})
2: choose $\beta^E_t \leftarrow$ (13) or (14) depending on $|\mathcal{D}_E| < \infty$?
3: choose $\beta^S_t \leftarrow$ (13) or (14) depending on $|\mathcal{D}_S| < \infty$?
4: $f^E_{\boldsymbol{x}_S}(\boldsymbol{x}_E) \leftarrow f(\boldsymbol{x}_S, \boldsymbol{x}_E)$
5: $f^S(\boldsymbol{x}_s) \leftarrow$ GP-UCB($\mathcal{D}_E$, $-f^E_{\boldsymbol{x}_s}$, $\{\sigma, k\}_E$, ε^E_{err}, β^E_t)
6: $\boldsymbol{x}_{OPT} \leftarrow$ GP-UCB($\mathcal{D}_S$, f^S, $\{\sigma, k\}_S$, ε^S_{err}, β^S_t)
7: **return** $\boldsymbol{x}_{OPT}$ ▷ *approximate solution of (2).*
8: **end procedure**

A. Nested Smoothed Minimax Optimization

Algorithm 2 gives a direct nested approach to optimize system parameters $\boldsymbol{x}_s \in \mathcal{D}_S$ for minimal worst-case costs. Note that this algorithm describes nested optimization loops: The other GP-UCB optimization loop (line 6) queries in each iteration the function $f^S(\boldsymbol{x}_s)$ which is defined as the optimum of the inner GP-UCB optimization loop in (line 5). $f^S(\boldsymbol{x}_s)$ corresponds exactly to the worst-case evaluation of system parameters $\boldsymbol{x}_s$, where the "Max-Agent" finds the most difficult environment parameters $\boldsymbol{x}_E$ by maximizing the costs $f(\boldsymbol{x}_s, \cdot)$.

Importantly, $f^S(\boldsymbol{x}_s)$ in line 5 is defined as the *mean estimator* of the worst-case cost provided by the inner GP-UCP loop. This mean estimator is *smooth* and much more reliable than a single evaluation from the noisy f! In our fully nested approach, the outer optimization loop "perceives" the cost function only through this worst-case mean-estimator function $f^S(\boldsymbol{x}_s)$, i.e., it perceives it through a smoothed and less noisy function than f itself.

B. Relaxed Minimax Optimization

In [10] and [11] it was proposed to relax problem (2). "Relaxation" here means that the (adversarial) max is not really taken over the full infinite space $\mathcal{D}_E$ of environment parameters, but only over a small, *representative* finite set $\mathcal{R} \subset \mathcal{D}_E$ of difficult environments. The key idea is to expand this representative set incrementally based on what system parameters have been found so far.

Algorithm 3 describes an implementation of this approach. The input parameters are the same as in algorithm 2 except for the desired relative error $\varepsilon^{\mathcal{R}}_{err}$. The first representative in $\mathcal{R}_0$ is chosen randomly. While in the nested approach f^S called an inner loop max-optimization GP-UCB to return the smoothed worst-case mean estimator, here we define $f^S_{\mathcal{R}}$ (line 6) to be only the worst case over the finite set $\mathcal{R}$ of representatives. We now iterate: We optimize system parameters (line 9) w.r.t. this relaxed worst-case evaluation. Then we find a new adversarial environment parameter setting $\boldsymbol{x}^E_t$ in line 10, which is the worst case for the current $\boldsymbol{x}^S_t$. This environment parameter $\boldsymbol{x}^E_t$ is added to the set $\mathcal{R}_t$ for the next round.

In contrast to the original formulation in [10] and [11], we use GP-UCB instead of Expected Improvement as core optimization engine, and we modified the stopping criteria

Algorithm 3 Adapted MiMaReK [10][11], using the UCB, $\mathcal{GP}$-priors and fixed total iterations for Continiuous Global Minimax Optimizaion

1: **procedure** ADAPTED-MIMAREK($\mathcal{D}_S$, $\mathcal{D}_E$, f, $\{\sigma, k\}_S$, $\{\sigma, k\}_E$, ε^S_{err}, ε^E_{err}, $\varepsilon^{\mathcal{R}}_{err}$)
2: choose $\beta^E_t \leftarrow$ (13) or (14) depending on $|\mathcal{D}_E| < \infty$?
3: choose $\beta^S_t \leftarrow$ (13) or (14) depending on $|\mathcal{D}_S| < \infty$?
4: initialize $\mathcal{R}_0 = \{\boldsymbol{x}_0\}$, $\boldsymbol{x}_0 \in \mathcal{D}_E$ randomly
5: $f^E_{\boldsymbol{x}_S}(\boldsymbol{x}_R) \leftarrow f(\boldsymbol{x}_S, \boldsymbol{x}_E)$
6: $f^S_{\mathcal{R}}(\boldsymbol{x}_S) \leftarrow \max_{\boldsymbol{x}_E \in \mathcal{R}} f(\boldsymbol{x}_S, \boldsymbol{x}_E)$ ▷ *relaxation fcn.*
7: initialize $t = 1$, $y_0 \leftarrow 0$, $y_1 \leftarrow \infty$
8: **while** $|y_t - y_{t-1}| > \varepsilon^{\mathcal{R}}_{err}$ **do** ▷ *relaxed minimax.*
9: $\boldsymbol{x}^S_t \leftarrow$ GP-UCB($\mathcal{D}_S$,$f^S_{\mathcal{R}_t}$,$\{\sigma$,$k\}_S$,ε^S_{err},β^S_t)
10: $[\boldsymbol{x}^E_t, y_t] \leftarrow$ GP-UCB($\mathcal{D}_E$,$-f^E_{\boldsymbol{x}^S_t}$,$\{\sigma$,$k\}_E$,$\varepsilon^E_{err}$,$\beta^E_t$)
11: $\mathcal{R}_t \leftarrow \mathcal{R}_{t-1} \cup \{\boldsymbol{x}^E_t\}$
12: $t = t + 1$
13: **end while**
14: $\boldsymbol{x}_{OPT} \leftarrow$ GP-UCB($\mathcal{D}_S$, $f^S_{\mathcal{R}_t}$, $\{\sigma, k\}_S$, ε^S_{err}, β^S_t)
15: **return** $\boldsymbol{x}_{OPT}$ ▷ *approximate solution of (2).*
16: **end procedure**

with great care (as described previously) because the originally proposed ones would not converge in our practical applications.

A core issue and difference of this relaxed minimax in contrast to our nested smoothed minimax is the handling of the stochasticity of f. While f^S in nested minimax is a smoothed mean-estimator, $f^S_{\mathcal{R}}$ is—in the original algorithm—merely a max over a finite set of noisy cost evaluations. If the cost evaluations are very noisy, taking the max may become problematic. We discuss this in more detail in the following.

C. Comparison of Direct and Relaxed Optimization

It is difficult to make general statements about those two algorithms for practical applications that differ in the underlaying cost function $f(\boldsymbol{x}_S, \boldsymbol{x}_E)$ in a highly nonlinear matter. Nevertheless the direct minimax approach uses a smoothed worst-case cost estimator while the relaxed minimax algorithm must take the noisy values directly from the relaxed max-operations. In theory this is not a problem because GP-UCB in line 9 of Algo. 3 can handle the noise. But in practice high noise can lead to a large number of total iterations and to suboptimal system values in every round. This also leads to a suboptimal environment value that is taken into account for the following round which causes error propagation.

For sanity check we tested both approaches on the non-noisy toy analytical example considered previously in [11]. This shows almost the same performance with slightly less function calls when enabling hyperparameter optimization per GP-UCB iteration, see table I where $\bar{\xi}_2$ and $\bar{T}$ are system parameters and $\bar{\beta}$ is the environment parameter. Adapting the hyperparameters works well because the underlaying cost function is analytical without any noise.

While it is very difficult to compare the general per-

	$\bar{\zeta}_2$	$\bar{T}$	$\bar{\beta}$	f-evaluations
MiMaReK	0.1986	0.861	1.038	640
Nested Minimax	0.1978	0.861	1.0398	606

TABLE I: Analytical example, presented in [11]. Optimized with both approaches.

formance we can analyze the efficiency with respect to total function calls of both algorithms. Let us denote the maximum function calls for system parameter optimization with $N_s := n_c + n_{UCB}^c$ where n_c is the number of an inital grid[2] and n_{UCB}^c denotes the maximum GP-UCB iterations. The maximum number of GP-UCB iterations with respect to the environment is denoted by N_e respectively. Then the actual function calls of $f(\boldsymbol{x}_S, \boldsymbol{x}_E)$ in Algorithm 3 in each iteration t is given by

$$n_{\text{MiMaReK}}(t) = \frac{1}{2}N_c + t\frac{1}{2}N_c + N_e. \tag{20}$$

Using $\sum_{t=1}^{T} t = \frac{T^2+T}{2}$ yields the total number of

$$N_{\text{MiMaReK}}(T) = \sum_{t=1}^{T} n(t) = T(\frac{1}{2}N_c + N_e) + \frac{T^2+T}{4}N_c \tag{21}$$

function calls. Solving the minimax-Problem (2) using algorithm 2 yields

$$N_{\text{Nested}} = N_c N_e \tag{22}$$

total function calls.

The degree of relaxation in algorithm 3 depends on the total iterations T (as $T \to \infty$ there is no relaxation). In theory both algorithms must handle the same function as N_c, N_e and T goes to infinity. We fix N_c, N_e to the same value in both algorithms and compute the total iterations of Mimarek T_{eq} subject to

$$N_{\text{Nested}} = N_{\text{MiMaReK}}(T_{eq}). \tag{23}$$

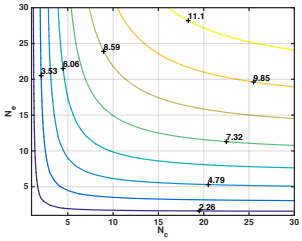
In Fig. 3, T_{eq} is shown over N_c and N_e. It turns out, that by rising N_c and N_e a large number of T_{eq} can be achieved. Especially when N_e is large, T increases steeply over N_c. This is because in eq. (21) $N_{\text{MiMaReK}}(T)$ depends quadratic on N_c and linear on N_e. Thus the relaxation in Mimarek is efficient in respect to N_e which also is the reason for the similar performance of the analytical example presented in [11] since $\dim \mathcal{D}_S = 2 > \dim \mathcal{D}_E = 1$.

V. Examples and Experiments

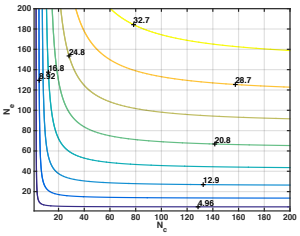
We first consider a pick'n'place application to demonstrated efficient automatic adversarial testing and leverage this to select from a finite set of alternative motion planning engines of MoveIt!. This reveals which path planning algorithm works worst-case optimally.

Second, we demonstrate the efficiency of our minimax methods for finding control parameters for a triple-inverted pendulum with nonlinear perturbations.

[2]In the original work [11] a grid can be used for function calls efficiency enhancements in combination with Expected Improvement.



(a) Contour lines of T_{eq} where (23) holds for $N_c, N_e \in [0, 30]$.



(b) Contour lines of T_{eq} where (23) holds for $N_c, N_e \in [0, 200]$.

Fig. 3: Visualization of possible T_{eq} s.t. (23).

For all experiments we use the termination values $\varepsilon_{err}^S = \varepsilon_{err}^E = 0.01$ and $\varepsilon_{err}^{\mathcal{R}} = 0.08$ for algorithm 2 and algorithm 3 respectively.

A. Pick'n'Place with the PR-2 Robot

As the MoveIt! ROS package[3] gets more and more popular for planning and execution tasks we use it for the realization of the pick'n'place application. Augmented reality (AR) markers are used for tagging the object and the desired goal position in the simulation. Those AR markers are tracked by the ar-track-alvar module.[4] This way a complete "sense-plan-act" system is considered for testing the planners (contrary to [1] that only uses an abstract planning scene for benchmarking).

For picking up and placing the object we use simple cartesian movements vertically to the object. To move the object from the pick to the place region we use the planning and execution methods provided by the MoveIt! interface that automatically takes the current environment into account for planning.

Finding an adversarial environment: We apply Algo. 1 to find adversarial environments with respect to planning and execution of the pick'n'place application. To this end we created the scenario shown in Fig. 4(a). The goal position of the object is on another table to the right of the PR-2 robot. There is also an obstacle on the table which must be taken into account for planning and execution throughout MoveIt!. The feasible set $\mathcal{D}_E$ of environment parameters are the $x_{obstacle}$ coordinate of the obstacle, shown in Fig. 4(b), and the x_{table} coordinate of the table, shown in Fig. 4(c), within $\mathcal{D}_E = [0, 0] \times [0.15, 0.10]$ meters. As an objective function we consider the weighted sum

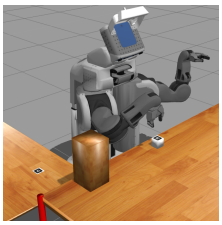
$$f_{feat}(\boldsymbol{x}_E) = \frac{1}{c_1}T_P + \frac{1}{c_2}T_E + \frac{1}{c_3}\|\boldsymbol{x}_{obj} - \boldsymbol{x}_{goal}\|^2 \tag{24}$$

of the features planning time T_P, execution time T_E and squared error between the goal and the object position. The coefficients c_1, c_2, c_3 are chosen to be the maximum of each feature multiplied with 3 such that eq. (24) is made up equally of every feature in the range $[0, 1]$.

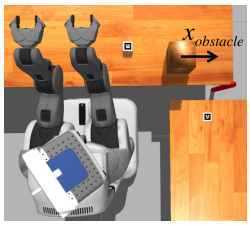
Using the default MoveIt! configuration the most adversarial example is given in the first row of table II. More

[3]`http://moveit.ros.org/`, checked 3/04/2015.

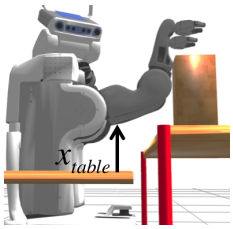
[4]`http://wiki.ros.org/ar_track_alvar`, checked 3/04/2015.



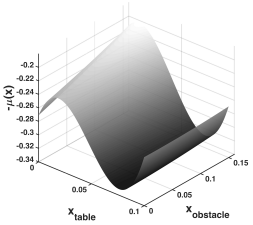
(a) Pick'n'place with obstacle set-up.



(b) Bird view: pick'n'place.



(c) Side view: pick'n'place.



(d) Mean estimator of (24).

Fig. 4: PR-2 Simulation Experiment set-up and result.

	Optimum	Function calls
Pick'n'Place	$\boldsymbol{x}_E = \begin{bmatrix} 0.00 \\ 0.08 \end{bmatrix}$	22
MiMaReK	$\boldsymbol{x}_S = \begin{bmatrix} 33.24 \\ 45.67 \end{bmatrix}, \boldsymbol{x}_E = \begin{bmatrix} 1.71 \\ 1.83 \\ 0.47 \end{bmatrix}$	696
Nested Minimax	$\boldsymbol{x}_S = \begin{bmatrix} 38.28 \\ 47.57 \end{bmatrix}, \boldsymbol{x}_E = \begin{bmatrix} 1.92 \\ 1.78 \\ 0.47 \end{bmatrix}$	704

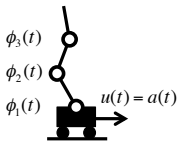
TABLE II: 1st row: Optimally adversarial environment parameters for the pick'n'place evaluation of planners. Rows 2 & 3: Found minimax optima for the control of the triple inverted pendulum.

interestingly, the corresponding response surface of (24) in Fig. 4(d), where we plot *negative cost* on the z-axis, which is minimized by the adversarial optimization.

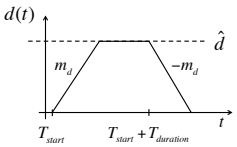
Choosing the planning algorithm worst-case optimal: Now we choose the optimal planner configuration in the adversarial setting. This implies switching the planning algorithm used in the open motion planning library (OMPL) inside the MoveIt! package. For this, the system parameters are elements of the categorial feasible set $\mathcal{D}_S = \{0, 1, .., 5\}$, where each integer identifies a path planning algorithm. Because the planner ID's are independent, we must search for an adversarial environment configuration for every single path planning algorithm. We solve the minimax optimization problem by comparing the "Max-Agent" adversarial mean estimates.

These adversarial mean estimates are shown in table III, where lower cost value is better (for the system). The "RRT-Connect" planner works best in "difficult" static pick'n'place scenes with the PR-2 robot. Remarkable is that the default planning algorithm "LBKPIECE" performs worst.[5]

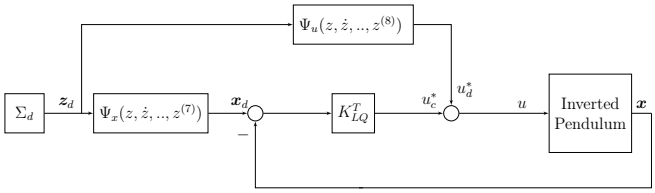
[5]MoveIt! version 0.5.8



(a) Triple Inverted Pendulum.



(b) Velocity disturbance without noise.



(c) Control Concept.

Fig. 5: Triple inverted Pendulum set-up.

Cost	Planner
1.00	Lazy Bi-directional Kinematic Planning by Interior-Exterior Cell Exploration (LBKPIECE)
0.76	Bi-directional Kinematic Planning by Interior-Exterior Cell Exploration
0.58	Probabilistic RoadMap
0.54	**Rapidly-exploring Random Trees Connect (RRTConnect)**
0.94	Single-query Bi-directional Lazy collision checking planner
0.81	Expansive Space Trees

TABLE III: Minimax selection of the planning algorithm with lowest worst-case costs.

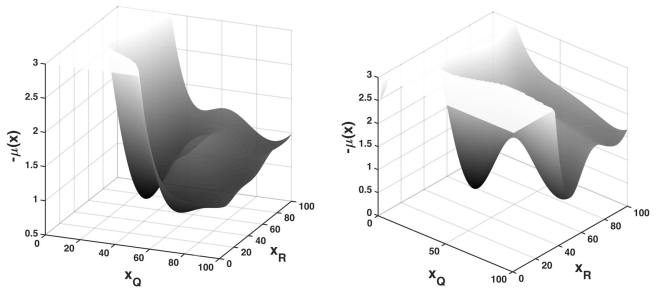
B. *Controlling a triple inverted pendulum*

The inverted pendulum is a commonly used example in control theory. There is a strong relation to upward walking robots. As an analytically not accessable control problem we want to optimize the matrices Q and R of a Linear Quadratic Regulator (LQR) that stabilizes a *triple* inverted pendulum (Fig. 5(a)) along a linearized flatness based feed-forward trajectory [13]. Starting from the upward position, the goal is to control the system to make a *side step* of one meter within 3.5 seconds, while adhering to additional velocity and acceleration constraints as well as velocity disturbances of the cart. For simulation we use the model equation

$$\dot{x}(t) = g(\boldsymbol{x}(t), u(t)) \tag{25}$$

from [14]. Hereby $\boldsymbol{x}(t) = [\boldsymbol{\phi}(t), v(t), s(t)]^T$ is the system state where $\boldsymbol{\phi}(t)$ denotes pair-wise angles and their time derivatives ϕ_i, $\dot{\phi}_i$, $i = 1, 2, 3$ of the vertical as well as the velocity and position of the cart $v(t)$ and $s(t)$. The input $u(t)$ is the cart's acceleration and is constrained to $|u(t)| \leq 10$ [ms^{-2}]. The position $s(t)$ represents the system output. $s(t)$ and its time derivate $v(t)$ are constrained to $|s(t)| \leq 1.3$ [ms] and $|v(t)| \leq 2.5$ [ms^{-1}] respectively. We extended the system (25) by an additive perturbation defined as

$$d^*(t) = d(t) + \epsilon_k \text{ [ms}^{-1}\text{]} \tag{26}$$



(a) Nested minimax. (b) MiMaReK.

Fig. 6: Triple inverted Pendulum mean estimators of max-optimal cost function (31).

of the carts velocity $v(t)$. The first term of (26), $d(t)$, is shown in Fig. 5(b) (m_d is the rising/falling rate) and the second, ϵ_k, is sampled in discrete time steps ($k\Delta T$,$k = 1, 2, ..,$ $\Delta T = 1$ [s]) from a normal distribution $\mathcal{N}(0, 0.0016)$. The additive velocity perturbation models external disturbances of the cart. Concerning the adversarial optimization over environment parameters, we define

$$\begin{aligned} \boldsymbol{x}_E &= [m_d, T_{start}, T_{duration}] \\ &\in \mathcal{D}_E = [0, 0, 0] \times [2.0, 2.0, 0.5] \ . \end{aligned} \tag{27}$$

As shown in Fig. 5(c) the control concept consits of a direct feed forward control u_d and a desired state trajectory $\boldsymbol{x}_d$ (provided by the flatness based approach [13]) along the LQR must stabilize the system. We parameterize our LQR by reweighting precomputed reference matrices Q^* and R^*, namely we assume

$$Q = x_Q Q^*, \ R = x_R R^* \ . \tag{28}$$

Here, Q^* and R^* are predefined to

$$Q = x_Q diag([10, 1, 10, 1, 10, 1, 10, 10]), \ R = x_R 0.01 \tag{29}$$

for nominal stability (without perturbation) of a 3.5 second side step under constraints. The parameter domain for system optimization is

$$\boldsymbol{x}_S = [x_Q, x_R]^T \in \mathcal{D}_S = [0, 0] \times [100, 100] \ . \tag{30}$$

The objective function is defined as

$$f = \int_{t=0}^{t=5} \|\boldsymbol{x}_{\mathrm{d}}(t) - \boldsymbol{x}(t)\|^2 dt \ , \tag{31}$$

where $x_{\mathrm{d}}(t)$ is the *desired* nominal state trajectory according to the feed forward control $u_{\mathrm{d}}(t)$ without perturbations, along the LQR must stabilize the perturbated nonlinear system.

The results of the optimization by MiMaReK and by the Nested approach are both shown in table II. To verify the minimax optimal results we fix the $\boldsymbol{x}_S$ parameters and perform an expensive optimization of the $\boldsymbol{x}_E$ parameters. It turned out that the result of the Nested approach was about 3% better than MiMaReK. Both response surfaces are illustrated in Fig. 6(a) and (b). Thereby the z-axis is the mean estimated *negative* cost over system parameters. I.e. the output of the adversarial inner optimization loop. While the maximum found by the nested approach was better than by the relaxed one, Fig. 6(b) shows that the response surface of the relaxed approach has relatively high values for low x_R values compared to the response surface of the nested method. This is caused by the "filter" effect the nested approach gets from the inner loop.

VI. CONCLUSIONS

We established a novel link between efficient automatic testing and optimization of intelligent systems and general minimax optimization. This idea lead us to a new minimax optimization approach, which we compared to a modified existing minimax optimization method based on relaxation. It turned out that with very few function evaluations remarkable results can be achieved. While the relaxed minimax approach can handle very high dimensional environment sets the nested approach and its resulting response surface is more accurate and robust under noise.

REFERENCES

[1] B. Cohen, I. A. Sucan, and S. Chitta, "A generic infrastructure for benchmarking motion planners," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 589–595.

[2] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Information-theoretic regret bounds for gaussian process optimization in the bandit setting," *Information Theory, IEEE Transactions on*, vol. 58, no. 5, pp. 3250–3265, 2012.

[3] A. Watanabe, A. Ohya, and S. Yuta, "Control parameter design for robot vehicle based on numerical simulation and heuristic optimization - feed-back controller design for trajectory tracking under strict physical constraints in wide speed range -," in *Computational Intelligence in Control and Automation (CICA), 2011 IEEE Symposium on*, April 2011, pp. 137–142.

[4] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "An experimental comparison of bayesian optimization for bipedal locomotion," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1951–1958.

[5] L. Stocco, S. Salcudean, and F. Sassani, "Fast constrained global minimax optimization of robot parameters," *Robotica*, vol. 16, no. 06, pp. 595–605, 1998.

[6] Y.-S. Ong, P. B. Nair, and K. Y. Lum, "Max-min surrogate-assisted evolutionary algorithm for robust design," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 4, pp. 392–404, 2006.

[7] A. M. Cramer, S. D. Sudhoff, and E. L. Zivi, "Evolutionary algorithms for minimax problems in robust design," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 2, pp. 444–453, 2009.

[8] R. I. Lung and D. Dumitrescu, "A new evolutionary approach to minimax problems," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 1902–1905.

[9] A. Zhou and Q. Zhang, "A surrogate-assisted evolutionary algorithm for minimax optimization," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010, pp. 1–7.

[10] J. Marzat, E. Walter, and H. Piet-Lahanier, "Worst-case global optimization of black-box functions through kriging and relaxation," *Journal of Global Optimization*, vol. 55, no. 4, pp. 707–727, 2013.

[11] H. P. Lahanier, J. Marzat, and E. Walter, "Robust minimax design from costly simulations." in *11th International Conference on Structural Safety & Reliability (ICOSSAR 2013)*.

[12] C. E. Rasmussen, "Gaussian processes for machine learning," 2006.

[13] M. Zeitz, "Differential flatness: A useful method also for linear siso systems," *at-Automatisierungstechnik*, vol. 58, no. 1, pp. 5–13, 2010.

[14] K. Graichen and M. Zeitz, "Feedforward control design for finite-time transition problems of nonlinear systems with input and output constraints," *Automatic Control, IEEE Transactions on*, vol. 53, no. 5, pp. 1273–1278, 2008.