Adapting Control Policies for Expensive Systems to Changing Environments

Matthew Tesch, Jeff Schneider, and Howie Choset

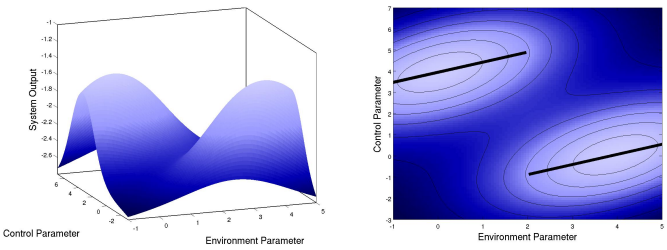
***Abstract*— Many controlled systems must operate over a range of external conditions. In this paper, we focus on the problem of learning a policy to adapt a system's controller based on the value of these external conditions in order to always perform well (i.e., maximize system output). In addition, we are concerned with systems for which it is expensive to run experiments, and therefore restrict the number that can be run during training. We formally define the problem setup and the notion of an optimal control policy. We propose two algorithms which aim to find such a policy while minimizing the number of system output evaluations. We present results comparing these algorithms and various other approaches and discuss the inherent tradeoffs in the proposed algorithms. Finally, we use these methods to train both simulated and physical snake robots to automatically adapt to changing terrain, and demonstrate improved performance on test courses with changing environments.**

I. Introduction

During typical operation of many robotic and industrial systems, the environment can change significantly. For example, a locomoting humanoid robot may move over gently upsloped terrain, traverse a horizontal, slightly bumpy area, and move downhill through many large obstacles. Assume there is a parameterized controller which can be tuned to perform well in each of these environments. Obviously, a static set of parameters for this controller would be a suboptimal method for controlling the system in multiple environments, as one would expect the controller parameters for uphill motion to be different than those for downhill. However, one would expect some continuity in the robot's performance – similar contol parameters should lead to similar performance – and in the optimal controller – two similar environments would likely engender similar optimal parameters (Fig. 1). In this paper, we seek to intelligently generate *control policies* that adapt to changes in the environment by selecting the best controller parameters for a given environment.

Unfortunately, for some systems it is infeasible to test every possible controller in every possible environment. In particular, we focus on systems for which evaluation of even a single controller/environment pair may take significant effort, and therefore we must minimize the necessary number of these evaluations. The choice of experiments (points at which to evaluate the system output) can significantly affect the quality of the resulting policy. The goal of *experiment selection* algorithms is to select parameters at which to evaluate in a manner that enables generation of the best possible policies.

Matthew Tesch, Jeff Schneider, and Howie Choset are with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA. This work was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.



(a) . (b) .

Fig. 1: **(a)** We are interested in problems for which the optimal control parameter changes significantly depending on the environmental conditions. Ideally similar environment/control combinations lead to similar system output; we therefore assume this function is continuous, although these methods can still operate with some discontinuities. **(b)** The resulting policy (dark lines that indicate mapping from environment to control) tends to be piecewise continuous; similar environments usually result in similar controllers, but there are likely to be some discontinuities. A good policy can be estimated from a low-cost model of the true expensive system output, requiring only a handful of carefully chosen points.

This resourceful policy generation is made possible by the assumption of continuous system output with respect to the controller parameters and the environment, allowing us to infer reasonable values for an unsampled system output based on nearby sampled values. We use statistical metrics such as *expected improvement* in order to choose where to sample the system output, so that we ensure that our evaluations are useful (rather than wasted on poor regions of our search space).

In this paper, we first outline the terms necessary to formally define the optimization problem and the goals of our algorithms. We then propose two algorithms which use a statistical measure to select points at which to sample the system output. We demonstrate the efficacy of these methods on a set of test functions, and then show results of several control policies created using these methods, applied to both simulated systems and physical robots. Specifically, we demonstrate improved locomotion over changing terrain for the snake robots described in [1], as compared to a control policy generated through random sampling of the environment and control parameter spaces.

II. Related Work

One of the keys to this work is the idea of using a *surrogate function* to represent a function which is expensive to evaluate, and basing search methods on this cheap model of the true function. These ideas have been extensively explored in the global optimization community [2], [3], often relying on stochastic processes to create a surrogate function [4], [5], [6]. In this paper, we use Gaussian Processes ([7]) as a function approximation method that generates an estimate of

the true system output along with a measure of confidence in that estimate.

Given a surrogate function, the goal of global optimization of expensive functions becomes using the surrogate to choose subsequent true function evaluations in order to minimize the number of total evaluations while maximizing performance. In cases where the function evaluations are costly (hours to days), computational requirements are not a significant issue; careful choice of the sample is more important. Note the difference from active learning, where the quality of the surrogate is paramount (e.g., [8]); instead, we are only interested in the model as a means to optimize a function, and do not require high confidence over the entire surface.

A number of heuristics and statistical methods have been derived to use information from the surrogate function to choose this sample location ([3] provides a survey of many existing methods). These include sampling at an upper confidence bound of the predicted function [9], [10], using the probability of improvement[11], [12], and the use of expected improvement[13], [14]. The latter has shown to effectively trade off exploration of the parameter space and exploitation of the known good areas, without requiring algorithm parameters to be carefully tuned. It has been used in the context of global optimization, and expanded to multi-objective optimization [15], [16], noisy black-box function optimization [17], [18], [19], and other domains. However, none of these methods are directly applicable to the addition of environment parameters. In contrast, our algorithms take these parameters into consideration through the optimization of a policy which defines a mapping on the function's domain. Inspired by the success of the expected improvement metric, we adapt it to be meaningful in selecting points which aim to improve our policy.

Perhaps the most closely related relevant work is that done in the field of robust controller selection [20], [21]. In particular this work explicitly breaks the parameter space into a control and environment subspace, as we do in this paper. However, robust controller selection methods seek to find a *robust* controller, rather than an *adaptive* controller. Their formulation treats the environment parameter as noise, which is useful when the environment is unobservable or changes at a timescale much shorter than the control bandwidth. Instead, we create a control policy, which adapts controller performance to observable, changing environments. In environments where the controller is fast relative to the changes in environment, adaptive control will produce a system with better overall performance.

III. Problem Definition

This work aims to find a control policy that optimally adapts to external variables, while minimizing the number of experiments done to perform the optimization. The two subgoals then become:

1) **Policy Generation:** After the completion of a predetermined number of system output evaluations, predict an optimal control policy based on the samples.
2) **Experiment Selection:** Select subsequent parameters for evaluation, based on the previous system output evaluations, which maximize the score of the policy predicted by the policy generation algorithm.

In order to make these notions more concrete, we define several terms and then restate these subgoals more precisely.

Definition 1 (Control Parameter): The control parameter space X_c is a compact subset of $\mathbb{R}^{m_c}$. Each $x_c \in X_c$ represents a particular value of the system of interest that can be fully specified during normal operation. Some examples of control parameters include the value of a set of gains in a PID controller, the relative concentration of two reactants in an industrial process, or the prescribed dosage of a drug during drug development.

Definition 2 (Environment Parameter): The environment parameter space X_e is a compact subset of $\mathbb{R}^{m_e}$. This space contrasts with the control space in that values $x_e \in X_e$ cannot be controlled under normal real-world operation, but can be specified in laboratory trials. Furthermore, the value of x_e can be measured during normal operation. Therefore, these parameters represent continuous valued external factors of the system, such as terrain steepness for a locomoting system, wind strength and direction for a UAV, particulate size in an industrial process, or disease strain during drug development.

Definition 3 (System Output): The system output, denoted as $f\colon X_e \times X_c \to \mathbb{R}$, is a continuous, real-valued function of the environment and control parameters. This function represents the performance of the system, given some environmental conditions and some specified control parameter. Example system outputs include the speed of a locomoting system over a terrain, the efficiency of a mechanical process, or the turbulance of a wing design calculated from a wind tunnel or computational fluid dynamics experiment. For the methods we propose here, we assume that sampling this system output is time intensive or computationally expensive, and therefore there is a limit to the number of times this function can be evaluated.

Definition 4 (Control Policy): The control policy defined in this work is a mapping $\gamma\colon X_e \to X_c$ (not necessarily surjective), such that $\gamma(x_e)$ represents the control parameter set by γ in reaction to sensing environment parameter x_e.

The *score* $\mathcal{S}$ of a control policy,

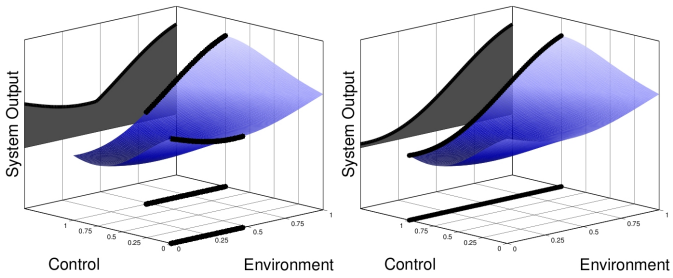
$$\mathcal{S}(\gamma) = \int_{X_e} \omega(x_e) f(x_e, \gamma(x_e)) dx_e, \tag{1}$$

represents how well this policy adapts to varied environmental parameters. The $\omega\colon X_e \to \mathbb{R}^+$ term is an optional weighting function that reflects the relative importance of learning controllers for various environments.

We define the optimal control policy for the system, γ^*, as the policy which maximizes $f(x_e, \gamma(x_e))\ \forall\ x_e \in X_e$. In other words,

$$\gamma^* = \underset{\gamma}{\operatorname{argmax}}\, \mathcal{S}(\gamma). \tag{2}$$

Note that γ^* is independent of ω, because $\hat{\gamma}^*(x_e)$ can be independently determined for each $x_e \in X_e$. Although the



(a) (b)

Fig. 2: An example system output for one dimensional environment and control spaces. The policy is shown below the function and its performance is projected to the left. An optimal policy is illustrated in (a), presenting the best control parameter for every environment parameter. Note that the policy shown in (b), which also maps to a control parameter that maximizes the system output at $x_e = 1$, results in a significantly lower overall score because of its poorer choices in other regions of the environment space.

weighting function can make a significant difference during experiment selection and policy comparison, the optimal policy is unaffected by the relative importance of different environments.

To illustrate these ideas, we use one-dimensional environment and control spaces (Fig. 2). In Fig. 2(a), the optimal policy γ^* is shown projected onto the control-environment plane, and the score integral is visualized on the system output-environment plane; Fig. 2(b) shows a suboptimal policy. Note that the system output is continuous, whereas the control policy does not have this restriction.

The goal of this work is to find the highest scoring γ after a number of system output evaluations. As above, we break this problem into *policy generation* and *experiment selection* subproblems.

1) **Policy Generation:** Given the results of n system output evaluations, choose the best estimate for γ^*.
2) **Experiment Selection:** Choose the sequence of points $X = \{x^1, x^2, \dots x^n\}$, $x^i \in X_e \times X_c$, where the choice of x^{k+1} is informed by $\{f(x^i) \mid i \leq k\}$, which maximizes the score of the policy produced by the chosen policy generation algorithm.

IV. Proposed Methods

The difficulty of applying a standard optimization technique to this problem is twofold. First, the true objective function we are maximizing during policy generation is $\mathcal{S}$. Unfortunately, evaluating $\mathcal{S}(\gamma)$, the score of a single policy, involves an integral of our expensive system output $f(x_e, x_c)$. As the policy generation task admits no new evaluations of f, and the experiment selection only allows n evaluations, it is clear that it is impossible to calculate $\mathcal{S}(\gamma)$ for any single policy γ, let alone apply a standard optimization technique to $\mathcal{S}$. Secondly, rather than directly searching the infinite-dimensional policy space, the problem setup requires selection of a series of $x^i \in X_e \times X_c$ which will yield the information necessary to build a good policy.

The first of these problems is addressed in our approach by using a surrogate function, $\hat{f}$, to model the system output and make decisions. This technique is widely used in global optimization of expensive functions, but the surrogate usually directly represents the function to be optimized. Here, we instead use it to represent a function that is an intermediary in the computation of our score function. However, we share with the global optimization community the key idea of replacing an expensive function with a surrogate, using the surrogate to make informed decisions, and then updating the surrogate when new information is available.

In particular, we use a Gaussian Process (GP) [7] for $\hat{f}$. Using GPs for regression has the benefit of providing a full posterior distribution (which is Gaussian) at each point; we denote the variance of this distribution as $\hat{\sigma}^2$. This means it is easy to determine confidence intervals or integrate an objective criterion over possible values of the function.

Use of a surrogate function entails the assumption that the model is a reasonable representation of knowledge about the function. If the hypothesis space is not rich enough to capture behavior of the system output, or the hypothesis space is too rich, the model could under or overfit the data, resulting in poor algorithm performance. For the experimental results, we have used the squared exponential kernel as the covariance for the GP, but the methods described below do not rely on any particular choice of covariance; in general we recommend selecting between model complexity (for example, different covariance functions for the GP) via a comparison of the leave-one-out predictive likelihood of each model. In addition, any prior knowledge about the system output should be encoded in the model prior.

Unsurprisingly, standard local and global optimization methods directly applied to optimizing system output perform poorly. This is because these methods are not optimizing the true objective; rather they focus on improving knowledge in environments with high system output results, and ignore environments with low system output results. This causes the resulting policy to be very weak in "difficult" environments (ones with low system output results), and therefore lowers the overall score. However, we have included results from the expected improvement global search algorithm EGO [14] as a point of comparison.

To respond to the problem encountered by such a method, one might choose to incorporate a method such as information gain, uniform sampling, or maximum dispersion point selection to ensure that all types of environments are equally sampled. Unfortunately, these methods cause poor regions of the control space to be sampled at the same frequency as good regions of the control space, resulting in an ineffective use of the limited sample budget. We have included a random point selection algorithm as a baseline to compare our efforts against.

These methods amount to heuristic approaches for this problem, and although heuristics can often perform well, we seek a more principled solution. The methods proposed in § IV-B and § IV-C attempt to provide a tractable solution which maximizes a statistical quantity related to the score function: approximate expectation of improvement above the current predicted policy score.

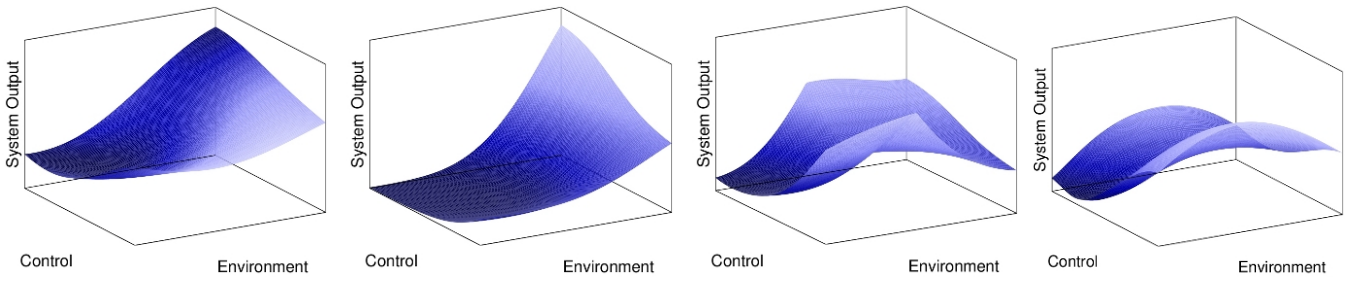


Fig. 3: **(a)** A surrogate for system output; the color represents the confidence (dark = high, light = low). **(b)** The value of a potential experiment using expected improvement. The expected improvement metric biases point selection towards environments with high system output values. **(c)** The value of a potential experiment using expected improvement over the best predicted system output for that environment. This reduces this bias toward easy environments and more directly optimizes the policy score. **(d)** The estimated policy score improvement as a result of sampling each point. This selection criterion is computationally intensive, but results in better performance than (c).

A. *Policy Generation*

Using the assumption that the surrogate function generated by the GP, $\hat{f}$, is our best estimate of the true function, the appropriate course of action is to select the policy which maximizes our best estimate of the score,

$$\hat{\mathcal{S}}(\gamma) = \int_{X_e} \omega(x_e)\hat{f}(x_e, \gamma(x_e))dx_e. \tag{3}$$

Therefore, we choose $\hat{\gamma}^* = \text{argmax}_\gamma \hat{\mathcal{S}}(\gamma)$. Although more efficient approximations could be applied, in low dimensional spaces $\hat{\gamma}^*$ can be estimated via a dense sampling of the (relatively) cheap $\hat{f}$.

Alternatively, one could also choose to take a lower-risk approach, and use a lower confidence bound of the surrogate function to define $\hat{\mathcal{S}}$. Using this definition of $\hat{f}$ to find $\hat{\gamma}^*$ biases the generated policy towards control/environment combinations that are more well-known, and reduces the likelihood that the policy will choose a controller that will perform extremely poorly in an unexplored region of the parameter space.

As this work mainly focuses on experiment selection methods, we assume a policy generation method based on maximization of (3) via dense sampling.

B. *Unbiased Expected Improvement*

Among many surrogate function based methods, one of the most popular is Jones et. al's EGO algorithm [14]. This method seeks to optimize an expensive system output by using a surrogate function to choose a sequence of points at which to evaluate the function. Each subsequent selected point x^k is a maximum of the *expected improvement*,

$$EI(x^t) = \int_{y_m}^{\infty} p(y^t)(y^t - y_m)dy^t, \tag{4}$$

over $y_m = \max_{1\leq i\leq k-1}(f(x^i))$, the best previously sampled system output. Here, y^t represents a sample from the predictive distribution given by the GP at x^t, and $p(y^t) = \mathcal{N}(\hat{f}, \hat{\sigma}^2)$ is the probability density of this sample. This metric seeks to provide a balanced method for trading off exploration of the search space and focus on the best regions discovered so far ("exploitation"), and does so in a principled fashion without requiring hand-tuned parameters. In doing so, expected improvement considers the whole distribution of possible experiment results. Furthermore, given a normal probability distribution for $p(y^t)$ as is true for GPs, the integral in (4) admits an analytic solution.

Although this method has shown considerable success, a naïve application of this approach to the selection of $x^k \in X_e \times X_c$ produces poor results for control policy generation. As the algorithm quickly finds the maximum regions of f, environments which have no control parameter that performs very well will not be explored at all. Instead, the algorithm biases its search towards environments that have high values for the system output (Fig. 3(b)). This results in a policy that performs well in "easy" environments, but suboptimally in "difficult" environments.

The approach proposed here adapts this basic idea to the explicit separation of the environment and control space. We consider the expected improvement of sampling at some $x^t = (x_e^t, x_c^t)$, but measure improvement over the maximum predicted value when $x_e = x_e^t$, or $\max_{x_c \in X_c}(\hat{f}(x_c, x_e^t))$, instead of over the best system output evaluation so far, $\max_{1\leq i\leq k-1}(f(x^i))$. This gives the *unbiased expected improvement* (UEI):

$$UEI(x^t) = \omega(x_e^t) \int_{\hat{\gamma}^*(x_e^t)}^{\infty} p(y^t)(y^t - \hat{\gamma}^*(x_e^t))dy^t. \tag{5}$$

The x^k chosen by this algorithm is then given by $\text{argmax}\, UEI(x)$.

By applying this method, we remove the inherent bias towards environments containing high values for f (Fig. 3(c)). Furthermore, this approach is related to the expected improvement of $\hat{\mathcal{S}}$. Since the calculation of $\hat{\mathcal{S}}$ involves integration of the performance of the controller $\gamma(x_e)$ over all environments, this method is equivalent to finding the (x_e, x_c) which maximizes the expected improvement of a single infinitesimal element of this integral.

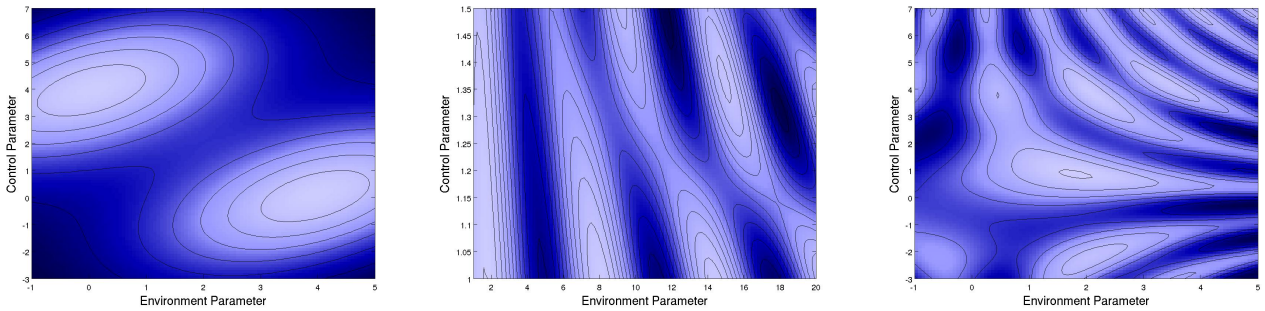


Fig. 4: Three analytic test functions designed for the comparison of point selection algorithms. The exact formulae are too lengthy for inclusion, but consist mainly of a collection of trigonometric functions.

C. Expected Policy Score Improvement

Although the unbiased expected improvement method begins to approximate improvement of the true policy score function $\mathcal{S}$, there is one crucial limitation. The improvement to the score calculated by $UEI(x^t)$ only considers improvement at one point in the environment space, independent of the other environments, and therefore is only an infinitesimal element of the true expected improvement of the policy score. However, sampling a point will add information to $\hat{f}$ about an entire region, not just a single point. To approximate the full expected improvement, we must find a way to extrapolate the effect of sampling a single point to an entire region. To measure this effect, we generate new GP surrogate functions for every potential system output value in the predictive distribution of a point, and integrate the improvement to the policy predicted by each new surrogate.

Using this method generates a more complete estimate of the effect of sampling a point on the policy score. Of course, computing a large numeric integral can also take significantly more time, and the quality of the solution can vary based on the resolution of the integral.

More formally, let us define the expected policy score improvement at a test point $x^t = (x_e^t, x_c^t)$ as:

$$EPSI(x^t) = \int_{-\infty}^{\infty} p(y^t) \int_{X_e} \omega(x_e) \max(\hat{\gamma}^*_{y^t}(x_e) - \hat{\gamma}^*(x_e), 0) dx_e dy^t, \quad (6)$$

where $\hat{\gamma}^*_{y^t}$ is the optimal policy generated by the surrogate function conditioned on the addition of a sample at x^t with value y^t. The point chosen for evaluation by this algorithm is $\operatorname{argmax} EPSI(x)$.

By maximizing (6), we are choosing a point to evaluate which maximizes expectation of improvement in the policy score function, rather than choosing a point which maximizes improvement in one differential element of the policy score integral. An example of this criterion is shown in Fig. 3(d).

D. Method Comparison and Discussion

As the unbiased expected improvement uses an analytic expression to calculate expected improvement it is relatively quick to calculate, but only considers one infiniteseminal element of the policy score integral (albeit arguably the most significant element). Expected policy score improvement provides a more complete approximation, but requires a numeric double integral, for which the integrand requires conditioning the GP.

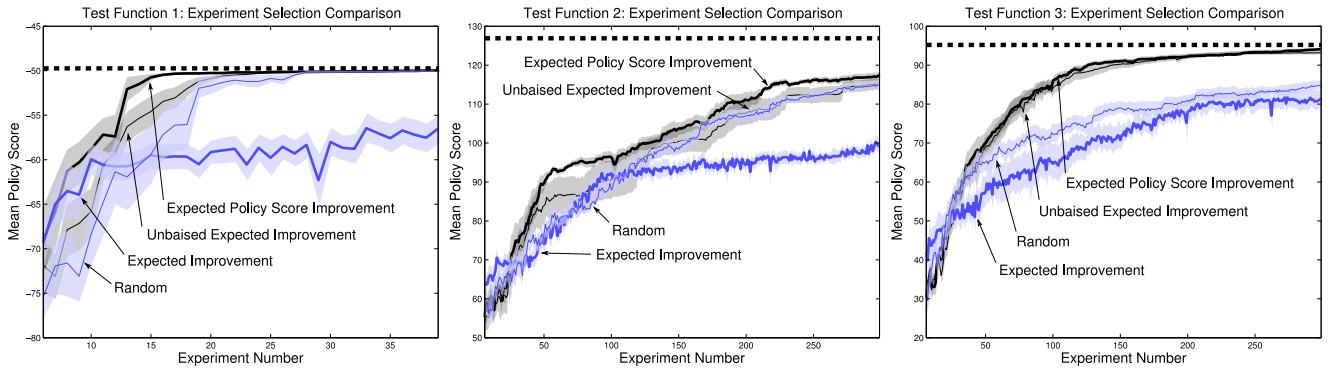
This produces a resulting selection surface which is smoother with respect to the environment parameter as compared to unbiased expected improvement (Fig. 3). Qualitatively the locations of the maxima chosen by each algorithm are similar. However, the EPSI approach gives a slightly more complete approximation of the true policy score improvement. This results in improved performance but suffers from a computationally burdensome numeric integral which may reduce its usefulness for some applications.

One notable difference in the computation of the expected improvement used here and that typically used for global optimization is that here we are calculating improvement over maxima of our surrogate function (an approximation to system output), whereas other approaches calculate this improvement over the best previous system output evaluation. We calculate improvement in this manner due to the fact that a randomly selected environment has zero probability of containing a previously evaluated point. The full implications of this difference are not explored in this paper.

V. Experimental Results

To evaluate the performance of these algorithms we first used analytic "test functions" in leiu of a physical system's system output, which allowed the completion of enough tests to enable one to draw reasonable conclusions. The algorithms were then used to create policies $\hat{\gamma}^*$ (as previously described in § IV-A) for both physical and simulated systems, which were then tested on a course with a changing environment. We created three test functions of varying complexity, shown in Fig. 4, which could be used to compare point selection algorithms. Specifically, these environments were created such that a static control policy would not be effective over the entire space.

To measure the score of a particular algorithm, we non-deterministically generate an initial set of k points through guaranteed-coverage sampling method, such as a latin hypercube [22], and then sequentially select $n - k$ more points, evaluating the system output after each choice. The predicted $\hat{f}$ is used after each evaluation to generate a policy, which is



(a) (b) (c)

Fig. 5: Comparison of algorithms on three analytic test functions. Each algorithm was run 10 times, and the scores averaged. The dotted black line represents the best possible policy for that function. Expected policy score improvement performed best, followed by unbiased expected improvement. Expected improvement selected initial experiments intelligently, but its inherent bias lead to poor overall performance. Random point selection suffered from no such bias, and resulted in steady policy score improvement with increasing experiments. Each line represents the mean of 20 trials, and the shaded regions indicate ± 1 standard error.

scored via a numeric integral on system output. This entire process was repeated 10 times for each algorithm, the results averaged, and standard errors calculated to provide a rigorous comparison of methods. The algorithms were written in MATLAB, and used the open-source Gaussian Processes for Machine Learning package provided by Rasmussen and Williams [23].

Although the algorithms described herein do not have any intrinsic parameters to tune, there are several implementation details which must be considered. Specifically, the choice of covariance function for the GP, the sampling strategy, the sampling density, and the density of the numeric integrals all can have an effect on overall performance. Therefore, to keep the comparison as fair as possible, we kept these choices constant when comparing the algorithms. In particular:

- The squared exponential covariance function was used for the Gaussian process.
- A grid of points was used as the set of all points at which to evaluate the metric. The location of this grid was defined by independent random offsets in each parameter, which were recalculated for the selection of each subsequent point x^i.
- Varying densities of this grid, from 10 to 40 points in the environment parameter space and 10-20 points in the control parameter space, were compared. Each algorithm was run for 10 trials at each grid density. The inner numeric integral of the expected policy score improvement method was a summation over the environmental component of this grid.
- The outer integral of the expected policy score improvement method was run at various resolutions, and with varying limits: integration limits of ± 3 and ± 4 were both used, each with sampling density of 31 and 61 points. These values were selected after conducting a small study of the effect of limits and sampling density on the accuracy of the numeric integration of the expected improvement metric.

We reiterate here that the expected policy score improvement method had greater computational requirements given the same implementation choices; however, we are interested in quality of the algorithm rather than computational performance, as the time required by the algorithm is assumed to be relatively insignificant compared to system output evaluation times for real systems.

A. *Test Functions*

A summary of the results of these methods is shown in Fig. 5. As expected, random point selection performs suboptimally, showing that it is important to carefully select experiments. However, as coverage is guaranteed, random selection results in continual policy improvement. Standard expected improvement also shows an unsurprising trend: initially, performance is comparable to the other algorithms, as it seeks out the best area of $X_e \times X_c$. However, the overall expected improvement is low in regions of X_e that do not have high values for f, and so policy score tends to stagnate quickly, ignoring potential score function improvements in these regions.

The unbiased expected improvement performs much better, eliminating the bias of standard expected improvement. This suggests that it is a reasonable, simple choice to use for tackling such point selection problems. Finally, the expected policy score improvement algorithm improves upon unbiased expected improvement, but only slightly. In all of the results, it was always shown to be at least as good, but often the margin of improvement was very slight. This indicates that although expected policy score improvement potentially gives a better approximation to the expected improvement of the policy score, unbiased expected improvement provides a much simpler and quicker method which produces similarly high quality results. The final choice between these methods involves several factors, and is largely application dependent.

While not strictly algorithm parameters, implementation decisions can have significant effects on performance. Figure 6(a) illustrates that a more dense sampling of candidate points improves the average generated policy's performance,

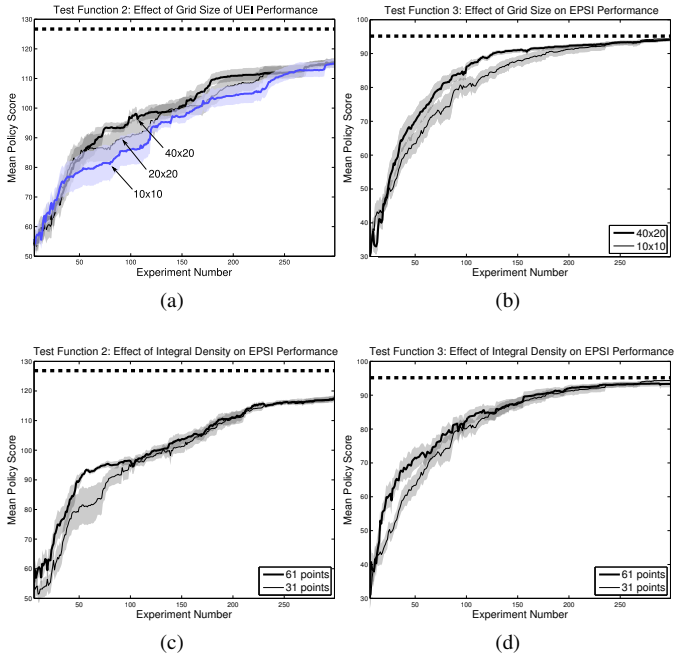


Fig. 6: A comparison of the effect of grid resolution (top row) and numeric integral density (bottom row) on algorithm performance. In general, higher density candidate point grids and numeric integrals resulted in improvements in the quality of results. All EPSI results shown use ± 3 standard deviations of $p(y^t)$ as the limits to the numeric integral. Each line represents the mean of 20 trials, and the shaded regions indicate ± 1 standard error.

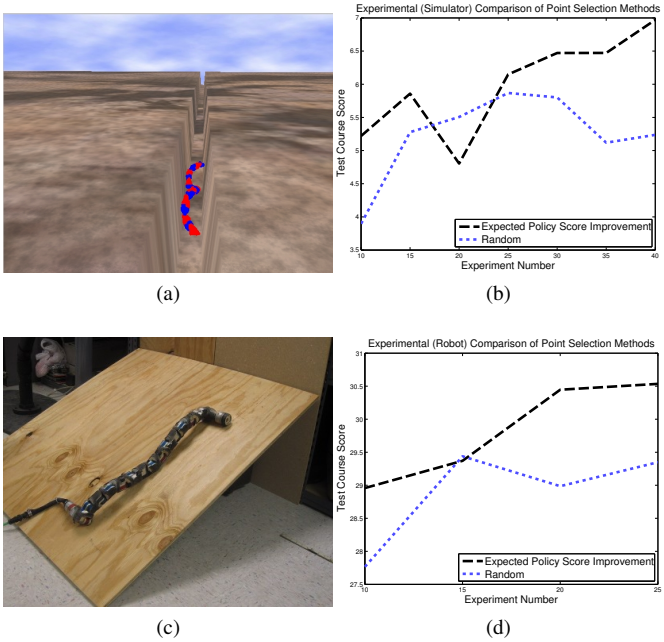


Fig. 7: Performance of policies generated from points selected randomly versus using the expected policy score improvement method. In the top row, a simulated snake robot crawls through a crevice of varying width using a helical rolling motion. The system performance is a measure of locomotive energy efficiency; high amplitude controllers do well in wide cracks but waste energy in small cracks. The bottom row shows a physical robot climbing up an incline; higher amplitudes work well for flat ground, but smaller amplitudes allow the robot to climb steeper inclines without slipping backwards. In both cases, 5 initial experiments were selected randomly before use of the point selection method. Candidate points were chosen from a grid 40 environment points by 20 control points. Only one trial was conducted for each setup.

relative to the start of the test, by as much as 50% (near experiment 75) for the UEI algorithm. During computation of the numeric integrals of the EPSI algorithm, this grid size was the resolution of the integral over the environment space (see (6)), and was also shown to have a notable effect (see Fig. 6(b)). The resolution of the integral over the predictive distribution for EPSI can be a limiting factor as well, as seen in Fig. 6(c) and 6(d). The dependence of EPSI on high density numeric integrals is one of the most significant limitations of this algorithm.

B. Simulation And Physical System Results

As such a complete analysis could not be run on physical systems due to the expensive nature of system output evaluation and the inability to compute a true policy score, we instead set up a range of environmental conditions in a "test course", and then used the above algorithms to generate policies which were scored on this test course. These policies map environment parameters (slope and crevise width) into a 2-D gait parameter control space (see [1]). The EPSI algorithm was compared to random point selection.

Two test courses were defined, one for a simulation of a snake robot and one for the physical mechanism in Choset's lab [24]. The tests involved crawling through a crevice and crawling up slopes of various steepness. As the control policies generated did not consider transition effects, the environment changes during evaluation on the test course involved instantaneous terrain and parameter changes (when using the physical snake this was accomplished by pausing the test in order to change these parameters).

Finally, the additional difficulty of noisy function evaluations is encountered when working with the robot and simulations. Although there is no explicit mention of noise in the algorithms, an appropriate choice of GP covariance function attempts to characterize this by fitting a noise parameter as well.This allows the algorithms to remain effective even in the presence of stochastic system output evaluations.

Results of evaluation of the policies generated during testing are shown in Fig. 7. Overall, expected policy score improvement caused superior policies to be generated, as compared to random point selection (this comparison is not as unfair as it might seem; in the analytic tests random performed second only to the proposed algorithms, because standard approaches are not appropriate for this problem). The difference can be noted even after only 10 samples of the space (the first 5 of which are the randomly generated initial sampling). These results also show that using surrogate functions still carries risk, as a bad surrogate function fit can

result in a bad policy, such as that seen after 20 experiments from the simulated snake. This result suggests the use of a low-risk policy generation method when few sampled points are available, and a less conservative method when more data is available.

VI. Conclusion

This paper has formulated an optimization problem that applies to expensive systems operating in varying environmental conditions. By carefully selecting training experiments, a superior control policy can be generated with a low number of system evaluations.

This framework is applicable to a rich set of problems. Locomoting systems, industrial processes, and prescription drugs all operate in changing environmental conditions, are expensive to test, and could benefit from optimal adaptive control policies. We have described two potential approaches for this experiment selection, both inspired by the statistical notion of expected improvement. One approach provides a fast, efficient computation that performs reasonably well, while the other is more complex and computationally intensive, but produces better results overall. We have also proposed a simple method for policy generation, given the experiments chosen by such an algorithm. We have demonstrated the efficacy of these algorithms, and presented a summary of results on analytic test functions as well as a physical snake robot system.

In addition to producing superior control policies as compared to naïve approaches, these methods have the advantage of requiring no algorithm parameters to tune for particular applications. However, there are still several implementation details that must be considered, including reasonable covariance functions for fitting Gaussian processes to the system output and numeric integral resolution.

Improving the algorithms so that these implementation details are less important choices is one direction for future work; by improving the efficiency of the computation in the numeric integral, we remove the limiting factor for high resoltion integrals. This could be approached by applying methods to quickly update a GP distribution given one additional sample, or deriving analytic expressions that could be used in place of numeric ones.

Other plans for extending this work include demonstrating the efficacy of the algorithms on higher dimensional problems, more extensive use for improving adaptability of real world systems, and comparing the proposed methods against a wider range of possible approaches to this problem. We are also interested in proving theoretic properties of the algorithms, such as completeness, and more rigorously accounting for noisy system output.

References

[1] M. Tesch, K. Lipkin, I. Brown, R. Hatton, A. Peck, J. Rembisz, and H. Choset, "Parameterized and Scripted Gaits for Modular Snake Robots," *Advanced Robotics*, vol. 23, pp. 1131–1158, June 2009.

[2] G. E. P. Box and N. R. Draper, *Empirical model-building and response surfaces*. Wiley, 1987.

[3] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[4] R. G. Regis and C. A. Shoemaker, "A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions," *INFORMS Journal on Computing*, vol. 19, no. 4, 2007.

[5] H.-M. Gutmann, "A Radial Basis Function Method for Global Optimization," *Journal of Global Optimization*, vol. 19, 1999.

[6] K. Holmström, "An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization," *Journal of Global Optimization*, vol. 41, no. 3, 2008.

[7] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[8] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active Learning with Gaussian Processes for Object Categorization," *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, Oct. 2007.

[9] A. W. Moore and J. Schneider, "Memory-based stochastic optimization," *Advances in Neural Information Processing Systems*, pp. 1066–1072, 1996.

[10] D. Cox and S. John, "A statistical method for global optimization," in *1992 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1241–1246, Ieee, 1992.

[11] H. J. Kushner, "A new method for locating the maximum point of an arbitrary multipeak curve in the presence of noise.," *Journal of Basic Engineering*, vol. 86, pp. 97–106, 1964.

[12] A. Žilinskas, "A review of statistical models for global optimization," *Journal of Global Optimization*, vol. 2, pp. 145–153, June 1992.

[13] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Towards Global Optimization*, vol. 2, pp. 117–129, 1978.

[14] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, vol. 13, no. 4, 1998.

[15] J. Knowles, "ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 50–66, Feb. 2006.

[16] M. Emmerich and J.-w. Klinkenberg, "The computation of the expected improvement in dominated hypervolume of Pareto front approximations," 2008.

[17] E. Vazquez, J. Villemonteix, M. Sidorkiewicz, and E. Walter, "Global optimization based on noisy evaluations: An empirical study of two statistical approaches," *Journal of Physics: Conference Series*, vol. 135, p. 012100, Nov. 2008.

[18] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng, "Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models," *Journal of Global Optimization*, vol. 34, pp. 441–466, Mar. 2006.

[19] V. Picheny, D. Ginsbourger, and Y. Richet, "Noisy Expected Improvement and on-line computation time allocation for the optimization of simulators with tunable fidelity," in *2nd International Conference on Engineering Optimization*, (Lisbon, Portugal), pp. 1–10, 2010.

[20] J. Lehman, *Sequential Design of Computer Experiments for Robust Parameter Design*. PhD thesis, Ohio State University, 2002.

[21] B. Williams, T. Santner, and W. Notz, "Sequential design of computer experiments to minimize integrated response functions," *Statistica Sinica*, vol. 10, no. 4, pp. 1133–1152, 2000.

[22] M. D. McKay, R. J. Beckman, and W. J. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, vol. 21, no. 2, pp. 239 – 245, 1979.

[23] C. E. Rasmussen and C. K. I. Williams, "Gaussian Processes for Machine Learning."

[24] C. Wright, A. Johnson, A. Peck, Z. McCord, A. Naaktgeboren, P. Gianfortoni, M. Gonzalez-Rivero, R. Hatton, and H. Choset, "Design of a modular snake robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2609–2614, IEEE, Oct. 2007.