Probability Collectives in Optimization

David Wolpert
Stefan Bieniawski
Dev Rajnarayan

SFI WORKING PAPER: 2011-08-033

SANTA FE INSTITUTE

Probability Collectives in Optimization

David H. Wolpert[‡] Stefan R. Bieniawski[§] Dev G. Rajnarayan[¶]

June 25, 2011

Abstract

This article concerns "blackbox optimization" algorithms in which one iterates the following procedure: Choose a value $x \in X$, getting statistical information about an associated value $G(x)$, then use the set of all pairs $\{(x, G(x))\}$ found so far to choose a next x value at which to sample G, the goal being to find x's with as small $G(x)$ as possible, and to do so as fast as possible. Examples of conventional blackbox optimization algorithms are genetic algorithms, simulated annealing, etc. These conventional algorithms work directly with values x, stochastically mapping the set $\{(x, G(x))\}$ to the next x. The distribution over new x's that gets sampled is never explicitly optimized. In contrast, in the Probability Collectives (PC) approach, one explicitly uses the set $\{(x, G(x))\}$ to optimize the probability distribution over x that will be sampled. This article reviews some of the work that has been done on Probability Collectives, in particular presenting some of the many experiments that have demonstrated its power.

1 Introduction

1.1 Intuition behind Probability Collectives

This article concerns "blackbox optimization" algorithms. In these algorithms one iterates a process of choosing a value $x \in X$, getting statistical information

[*] Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM, 87501

[†] Center for NonLinear Studies, MS B258, Los Alamos, NM, 87545

[‡] NASA Ames Research Center MailStop 269-1, Moffett Field, CA 94035-1000 (650) 604-3362 (V), (650) 604-3594 (F), david.h.wolpert@nasa.gov

[§] Boeing Research & Technology, P.O. Box 3707 MC 42-51, Seattle, WA 98124-2207, (206) 544-4511 (V), stefan.r.bieniawski@boeing.com

[¶] Desktop Aeronautics, Inc., 1900 Embarcadero Road, Suite 101, Palo Alto, CA 94303, (650) 380-4617, dgorur@desktopaero.com

about an associated value $G(x)$, and then using all pairs $\{(x, G(x))\}$ found so far to choose a next x value at which to sample G. The goal is to find x's with as small $G(x)$ as possible. In most the versions of the blackbox optimization problem we consider here, at no point in this iterative process does one have any knowledge concerning G beyond the set of sample pairs found so far. We also presume that it is evaluations $G(x)$ that are most expensive, and so treat any "off-line" processing that determines the next x at which to evaluate $G(x)$ as cost-free.

Conventional approaches to blackbox optimization work directly with values x, in that they specify a map K from any set of pairs $\{(x, G(x))\}$ to a next sample x. In the "Probability Collectives" (PC) approach, one instead works with probability distributions over x, in that one specifies a map κ from any set of pairs $\{(x, G(x))\}$ to a next *distribution* over X, $q(x)$, which gets sampled to form a next x.[1] In conventional approaches to blackbox optimization one tries to design K so that after iterating it one is likely to produce x's with small values of $G(x)$. In PC one instead tries to design κ so that after iterating it one is likely to produce $q(x)$'s that are peaked about small values of $G(x)$. There are many ways to formalize this goal, e.g., as finding the q that minimizes the expected value $\int dx\ G(x)q(x)$. Indeed, the optimal q is one whose support is restricted to the minimizers of $G(x)$, so that sampling those q's will give x's that minimize $G(x)$.

There are many advantages to working with distributions over X, as in PC, rather than working with x values directly. For example, many PC algorithms do not need to be modified even if one changes the type of the space X, so long as X has some very general characteristics (e.g., is finite). This is because whatever the type of X, a probability distribution over X — which is what the PC algorithm works on — is the same kind of object, namely a vector of real numbers. Indeed, PC's goal of iteratively modifing the distribution q to find low values of $\int dx\ G(x)q(x)$ amounts to iteratively modifying a real-valued vector q to minimize a real-valued function of that vector. This is exactly what is done in optimization schemes like gradient descent or Newton's method. Accordingly PC can leverage the power of such schemes. In particular, we can do this even if X is a categorical, finite space. So by using PC, "gradient descent for categorical variables" is perfectly well-defined.

Another advantage is that often the q produced in a PC-based approach will be tightly peaked in certain dimensions, while being broad in other dimensions. This provides sensitivity information concerning the relative importance to solving the optimization problem of getting the values of those dimensions precisely correct. Another advantage we get by optimizing q is that we can initialize it to a set of

[1] Throughout this paper, the measure over X is implicit, and we will loosely use terms like "probability distribution" even if X is uncountable and we should really refer to "probability density functions". Similar we will typically refer to "integrals" even when there is a point mass measure, so that we should really refer to "sums".

broad peaks each centered on a solution x^i produced by some other optimization algorithm. Then as that initial q gets updated, the set of solutions provided by those other optimization algorithms are in essence combined, to produce a solution that should be superior to any of them individually.

1.2 Delayed and Immediate Sampling PC

There are two basic types of PC that have been explored to date, **delayed sampling** and **immediate sampling**. Historically, delayed sampling PC was investigated first. It was originally motivated by restricting q to product distributions, and casting the goal of the PC algorithm as finding such a q that minimizes $\int dx\ G(x)q(x)$. For this setup one can derive simple algebraic formulas for how q should be updated from one iteration to the next, if one knew certain expectation values perfectly. Since in practice one doesn't know those expectation values perfectly, as a final step in the derivation of the update rule one has to introduce Monte Carlo (MC) techniques. (The name derives from the fact that the MC arises after the algebraic manipulations.)

Using product distributions has the advantage that it lends itself to parallel implementation (the separate component distributions of the product can be allocated to the separate parallel systems). That makes is particularly well-suited to distributed control applications. However product distributions have the disadvantage that they sometimes result in slow convergence of the associated PC algorithm, e.g., when the value that is best for one component of x depends critically on the value for another component of x. Although there are some tricks for circumventing this problem (e.g., the "semi-coordinate transformations" described in [1]), it is often advantageous to avoid the problem entirely and use distributions that are not product distributions.

Allowing such arbitrary classes of probability distributions is the primary motivation for immediate sampling PC. We write the goal of immediate sampling PC as finding the parameter θ that minimizes an integral transform

$$\mathscr{F}_{\mathscr{G}}(q_\theta) \triangleq \int dxdg\ P(g \mid x, \mathscr{G}) F(g, q_\theta(x)) \tag{1}$$

for a distribution q_θ in some class of θ-parameterized distributions. To motivate this notation, let $\mathscr{G}$ specify a "blackbox oracle" that stochastically returns real values g in response to any input x. Also take $F(g, q_\theta(x)) = gq_\theta(x)$. Then $\mathscr{F}_{\mathscr{G}}(q_\theta)$ is the average value of $g(x)$, evaluated under the x-distribution $q_\theta(x)$, with the g value at x given stochastically by $P(g \mid x, \mathscr{G})$. In particular, if $P(g \mid x, \mathscr{G}) = \delta(g, G(x))$ for some function G (i.e., if g is a deterministic function of x), and θ runs over the space of product distributions, then $F(g, q_\theta(x))$ is just the quantity we wish

to minimize in delayed sampling PC. However the notation (and techniques) of immediate sampling PC applies more broadly.

In immediate sampling PC we use MC techniques "immediately", to estimate $\mathscr{F}_{\mathscr{g}}(q_\theta)$ from the given set of samples for all values of θ. We then use powerful techniques from machine learning to choose among the possible θ's based on those associated estimates. The algebraic manipulations used in delayed sampling before any MC estimation is done are absent in immediate sampling, since those manipulations are only possible for the special case where q_θ is the set of all product distributions.

1.3 Relevant literature and roadmap of this paper

In [2] can be found pedagogical examples where there are closed-form solutions for the q produced by an elementary delayed sampling PC optimization algorithm. Also presented there is a proof that in the infinitesimal limit, many techniques for updating q become identical; these techniques all become a variant of Evolutionary Game theory's replicator dynamics in that limit. See [3] for an explicit formulation of how to apply delayed sampling PC to scenarios where the underlying variable is the trajectory of a multi-dimensional variable through time, i.e., to a policy-optimization scenario. Related connections between game theory, statistical physics, information theory, and PC are discussed in [4].

See [5, 6, 7, 8, 3, 9, 10, 11] for other work on delayed sampling PC, including both software and hardware experiments, for both optimization and control. In particular, see [12, 2, 3] for work showing, respectively, how to use delayed sampling PC to improve Metropolis-Hastings sampling, how to relate delayed sampling PC to the mechanism design work in [13, 14, 15, 16], and how to extend it to continuous move spaces and time-extended strategies.

See [17] for discussion of the relation of delayed sampling PC to other probability-based approaches to optimization and control, including [18, 19, 20, 21, 22, 23, 24]. See [25, 14] for earlier, less formal work related to delayed sampling PC.

Finally, see [1] for an overview of delayed sampling PC, and for many extensions of the basic delayed sampling PC algorithms, including extensions for the case where we have constraints on X, and extensions that overcome the restrictions imposed by product distributions, while still retaining the distributed nature of the algorithm.

Since immediate sampling is a more recent body of techniques, fewer papers have been produced so far on it. See [26, 27, 28, 29].

In Sec. 2 we present a quick summary of the theory of delayed sampling PC. Sec. 3 then presents some implementation details and experiments. In Sec. 4 we present a quick summary of the theory of immediate sampling PC. Sec. 5 then presents some implementation details and experiments.

2 Delayed Sampling Theory

2.1 The Maxent Lagrangian

Say we are given the following problem:

(P3): Find

$$\min_{\{q_i\}} \int dx\ G(x) \prod_i q_i(x_i)$$
$$\text{such that}$$
$$\int dx_i\ q_i(x_i) = 1\ \forall i$$
$$q_i(x_i) \geq 0\ \forall i, x_i$$

where each q_i is a single-dimensional real variable, which we will sometimes refer to as the "move" or "choice" of "agent i".

In conventional continuous optimization one standard way to solve (P3) is by replacing the inequality constraints with **barrier function**s $\phi_i : \mathbb{R} \to \mathbb{R}$ each of which is nowhere negative and which is also infinite for negative values of its argument [30] . This replaces the inequality constrained problem (P3) with the following problem having no inequality constraints:

(P1): Find

$$\min_{\{q_i\}}[\int dx\ G(x) \prod_i q_i(x_i) + \sum_{i=1}^{N} \int dx_i\ \mu(i, x_i)\phi_i(q_i(x_i))]$$
$$\text{such that}$$
$$\int dx_i\ q_i(x_i) = 1\ \forall i$$

where the non-negative real values $\mu(i, x_i)$ are the **barrier parameters**. Due to the nature of the barrier functions, we know that the solution to (P1) must obey the inequality constraints for any allowed values of the barrier parameters. Enforcing the equality constraints of (P1) as well guarantees that out q meets all of our constraints, i.e., is feasible.

Typically with barrier function methods one solves a sequenced of optimization problems, each of which has fixed barrier parameters. Each such problem starts with the q produced by the solution to the preceding problem. The difference between two successive problems is that the barrier parameter has been slightly reduced. As such an annealing progresses the barrier parameters disappear and (P1) becomes (P3), i.e., our final q is both feasible and (at least locally) solves the

minimization over q of problem (P3). For convex objective functions, this process has certain guarantees of converging to the global optimum of (P3). (Note though that our objective function, $\mathbb{E}_q(G)$, is not a convex function of the components of q; due to the fact that q is a product distribution, $\mathbb{E}_q(G) = \int dx \prod_i q_i(x_i) G(x)$ is a multinomial function of q.)

One common choice of barrier function is $\phi(y) = y\ln(y)$ for $y > 0$, $\phi(y) = \infty$ otherwise.[2] Say we also take $\mu(i, x_i)$ to be independent of x_i, so we can write it as the vector with components $\{T_i\}$. For this case, when q is normalized (i.e., satisfies the equality constraints), the operand of the min operator becomes $\mathbb{E}_q(G) - \sum_i T_i S(q_i)$, where $S(q)$ is the Shannon entropy of q_i. If T_i is independent of i, this difference becomes $\mathbb{E}_q(G) - TS(q)$. This expression is called the **free energy** in statistical physics, if we identify G with the "Hamiltonian" of the system and T with its "temperature". It is also the **Kullbach-Leibler** distance from q to the Boltzmann distribution $p(x) \propto exp(-G(x)/T)$, up to an irrelevant q-independent additive constant.[3] See [8] for a discussion of the shape of the function taking q to qp KL distance for product distributions q.

Continuing with our choice of entropic barrier function, we can solve (P1) via Lagrange parameters in the usual way, getting the Lagrangian

$$\mathscr{L}(q, \vec{T}) \quad = \quad \mathbb{E}_q(G) - \sum_i T_i S(q_i) + \sum_i \lambda_i [\int dx_i \; q_i(x_i) - 1] \tag{2}$$

with the obvious extension of the definition of E and S to all of the space of real-valued possible functions over x. For simplicity from now on we will take all the T_i to be the same. In this case we refer to the expression in Eq. 2, as the **Maxent Lagrangian**, since it can also be justified using the Maximum entropy principle of information theory [4]. It is just the free energy plus the dot product of the Lagrange parameter vector with the equality constraint functions. (Many other Lagrangians have also been explored; see some of the literature mentioned above.)

2.2 The gradient of the free energy

Say that T is fixed. In that case the entropy term in the free energy is globally convex in the remaining free variable, q. The $\mathbb{E}_q(G)$ term in the free energy would

[2]Formally, for ϕ to be a barrier function, we must add a constant large enough so that ϕ is nowhere negative, but such a constant is irrelevant for our purposes.

[3]Recall that the Kullbach-Leibler distance from an arbitrary distribution $q(x)$ to an arbitrary distribution $p(x)$ is $-\int dx \; q(x)\ln[p(x)/q(x)]$. We will sometimes call this the **qp KL distance**, to distinguish it from the **pq KL distance** which is the Kullbach Leibler distance from p to q. These two distances are also sometimes called the "M-projection" and the "I projection", respectively, in the literature. In general, KL distance between two distributions is non-negative, and equals 0 iff those distributions are identical. See [31, 32, 33].

also be convex, if we had only a single agent (in that case $\mathbb{E}_q(G)$ would be linear in q.) Accordingly, the free energy would be convex over $\mathcal{Q}$, the space of all product distributions over X. That is also true of the equality constraints, no matter how many agents we have. So the full Maxent Lagrangian would be convex if we had only a single agent, and the usual associated guarantees about duality gaps, saddle point solutions, etc., would apply. However for multiple agents $\mathbb{E}_q(G)$ is not linear in q, and we do not have those convexity guarantees.

For this more general case, the simplest thing to do is an iterative descent of the free energy over $\mathcal{Q}$, where at every iteration we step in the direction within $\mathcal{Q}$ that, to first order, maximizes the drop in free energy. Since the step is restricted to lie within $\mathcal{Q}$, the equality constraints of the Maxent Lagrangian are obeyed automatically. Expanding, to first order the step that maximizes the drop in $\mathscr{L}$ is given by (-1 times)

$$\nabla_q \mathscr{L}(q) - \eta(q) \tag{3}$$

where the $\eta(q)$ term is chosen to ensure that we stay on $\mathcal{Q}$.[4] To evaluate the $q_i(x_i)$ component of the gradient (one such component for every agent i and every possible move x_i by that agent) we write

$$[\nabla_q \mathscr{L}(q)]_{q_i(x_i)} = \frac{\partial \mathscr{L}}{\partial q_i(x_i)} = \mathbb{E}_{q_{-i}}(G \mid x_i) + T \ln[q_i(x_i)] \tag{4}$$

where

$$\mathbb{E}_{q_{-i}}(G \mid x_i) = \int dx_{-i} q_{-i}(x_{-i}) G(x_i, x_{-i}) \tag{5}$$

with $x_{-i} \triangleq [x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n]$ and $q_{-i}(x_{-i}) \triangleq \prod_{j=1|j\neq i}^{n} q_j(x_j)$.

For finite move spaces, the $q_i(x_i)$ component of $\eta(q)$ is independent of x_i, and given by

$$[\eta(q)]_i \triangleq \frac{1}{|X_i|} \int dx_i' \, [\nabla_q \mathscr{L}(q)]_{q_i(x_i')} \tag{6}$$

where $|X_i|$ is the number of possible moves x_i, i.e., this choice ensures that $\int dx_i q_i(x_i) = 1$ after the gradient update to the values $q_i(x_i)$.

2.3 Monte Carlo-based gradient descent and shrink-wrapping

Eq. 3 gives the (negative of the) change that each agent should make to its distribution to have them jointly implement a step in steepest descent of the Maxent Lagrangian. These updates are completely distributed, in the sense that each agent's update at time t is independent of any other agents' update at that time. Typically at any t each agent i knows $q_i(t)$ exactly, and therefore knows $\ln[q_i(j)]$. However

[4] N.b., we do *not* project onto $\mathcal{Q}$ but rather add a vector to get back to it. See [8].

often it will not know G and/or the q_{-i}. In such cases it will not be able to evaluate the $\mathbb{E}(G \mid x_i = j)$ terms in Eq. 3 in closed form.

One way to circumvent this problem is to have those expectation values be simultaneously estimated by all agents by repeated Monte Carlo sampling of q to produce a set of $(x, G(x))$ pairs. Those pairs can then be used by each agent i to estimate the values $\mathbb{E}(G \mid x_i = j)$, and therefore how it should update its distribution. In the simplest version of this scheme such an update to q only occurs once every L time-steps. Note that only one set of Monte Carlo samples is needed for all players to determine how to update their mixed strategy, no matter how many players there are.

In this simple Monte Carlo scheme only the samples $(x, G(x))$ formed within a block of L successive time-steps are used at the end of that block by the agents to update their distributions (according to Eq. 3). More sophisticated approaches modify the G values returned by the Monte Carlo on a player-by-player basis, etc. [34, 3].

In addition, in general it may be that some agent i has no sample for the value x_i. Some approaches to address this are presented below. These approaches are fairly elaborate, being explicitly designed to be able to work for uncountable X_i. They are based on "just in time" evaluation of how to update the probability of a move, together with interpolation to estimate quantities like $\mathbb{E}(G \mid x_i)$.

There are a set of simpler approaches, similar to Levenberg-Marquardt descent, that are particularly appropriate for finite X_i. These approaches are known as **shrink-wrapping**. We present them here in a slightly more general context than that of minimizing the free energy.

2.4 Brouwer updating

It is possible to write down, explicitly, the q that forms the fixed point of gradient descent of the free energy. Setting $\tilde{\nabla}_q \mathscr{L}(q)$ to zero in Eq. 4 gives the solution

$$q_i(x_i) \propto \exp[-\mathbb{E}_{q_{-i}}(G \mid x_i)/T] \tag{7}$$

This is a set of coupled nonlinear equations for the vectors $\{q_i\}$. Brouwer's fixed point theorem guarantees the solution of Eq. 7 exists for any G [4, 34]. Hence we call update rules based on this equation **Brouwer updating**. Note that just like gradient-descent, Brouwer updating also involves the expression $\mathbb{E}(G \mid x_i)$. Accordingly, when that term cannot be evaluated in closed form, it can be estimated using Monte Carlo techniques, just like in gradient-descent.

In **serial** Brouwer updating, one agent at a time updates its distribution, jumping to the optimal distribution given by Eq. 7. The order in which the agents update their distributions can be pre-fixed or random. The order can also be dynamically determined in a greedy manner, by choosing which agent to update

based on what associated drop in the Maxent Lagrangian would ensue [34, 2, 9]. (These various ordering schemes are similar to the those used in the majorization and block relaxation techniques in optimization statistics.) Aside from potential effects from Monte Carlo estimation error, each step of serial Brouwer updating is guaranteed not to increase the associated (Maxent) Lagrangian.

In **parallel** Brouwer updating, this updating procedure is followed simultaneously by all agents. Accordingly, when Monte Carlo sampling is used, parallel Brouwer updating can be viewed as a type of learning in games (see [20, 35, 36]). Now in general, when any q_j changes, for every $i \neq j$, what distribution q_i minimizes i's Lagrangian will change, in accord with Eq. 7. This suggests that a step of parallel Brouwer updating may "thrash", and have each agent change in a way that confounds the other agents' changes. (See [20, 36] and references therein for analysis of related issues in ficticious play.) In such a case the update may not actually decrease the associated (Maxent) Lagrangian, unlike with serial Brouwer.

There are many possible ways of addressing this problem. One is to mix parallel and serial Brouwer updating, so that only subsets of the agents perform parallel updates at any given time. These can be viewed as management hierarchies, akin to those in human organizations. Often such hierarchies can also be determined dynamically, during the updating process. Other ways to address the potential thrashing problem with parallel Brouwer have each agent i not use the current value $\mathbb{E}_{q^t_{-i}}(G|x_i)$ alone to update $q^t_i(x_i)$, but rather use a weighted average of all values $\mathbb{E}_{q^{t'}_{-i}}(G|x_i)$ for $t' \leq t$, with the weights shrinking the further into the past one goes. This introduces an inertia effect which helps to stabilize the updating. (Indeed, in the continuum-time limit, this weighting becomes the replicator dynamics [2].)

A similar idea is to have agent i use the current $\mathbb{E}_{q^t_{-i}}(G|x_i)$ alone, but have it only move part of the way the parallel Brouwer update recommends. Whether one moves all the way or only part-way, what agent i is interested in is what distribution will be optimal for i *for the next distributions of the other agents*. Accordingly, it makes sense to have agent i predict, using standard time-series tools, what those future distributions will be. This amounts to predicting what the next vector of values of $\mathbb{E}_{q^t_{-i}}(G|x_i)$ will be, based on seeing how that vector has evolved in the recent past. See [20] for related ideas.

2.5 Nearest Newton

Care must be taken when using second order methods to descend the Maxent Lagrangian. One approach starts by making a quadratic approximation (over the space of all distributions p, not just all product distributions q) to the Lagrangian, $\mathscr{L}(p)$ based on the current point p^t. Newton's method then specifies a p^{t+1} that minimizes that quadratic approximation. We can then find the product distribution

that is nearest (in pq KL distance) to p^{t+1} and move to that product distribution. The resultant update rule for the Maxent Lagrangian is called **Nearest Newton** descent [8]:

$$\frac{q_i^{t+1}(j)}{q_i^t(j)} = 1 \quad - \quad S(q_i^t) - \ln(q_i^t(j)) \\ - \quad \beta[\mathbb{E}_{q^t}(G \mid x_i = j) - \mathbb{E}_{q^t}(G)] \tag{8}$$

where q^t is the current (assumed to be product) distribution. The conditional expectations in Nearest Newton are the same as those in gradient descent. Accordingly, they too can be estimated via Monte Carlo sampling, if need be.

In [2] it is shown that in the continuum time limit, Nearest Newton updating for modifying the probability distribution of any particular agent becomes a variant of replicator dynamics (with the different strategies of replicator dynamics identified with the different possible moves of the agent performing the Nearest Newton update). That paper also shows that that when the terms in that continuum limit version of Nearest Newton are Monte-Carlo estimated, Nearest Newton becomes a version of ficticious play. More precisely, it becomes identical to a "data-aged" continuum time limit of parallel Brouwer updating. The stepsize of the Nearest Newton procedure is identical to the constant in the exponent of the data-aging.

2.6 Private Utilities

Performing the updates given above involves a separate conditional expected utility for each agent. Since accurate estimates usually require extensive sampling, the world utility G occurring in each agent i's update rule can be replaced with a private utility g_i chosen to ensure that the Monte Carlo estimation of $\mathbb{E}(g_i|x_i)$ has both low bias, with respect to estimating $\mathbb{E}(G|x_i)$, and low variance [37]. Bias represents the alignment between the private utility and world utility. With zero bias, updates that reduce the private utility are guaranteed also to reduce the world utility. It is also desirable for an agent to distinguish its contribution from that of the other agents: variance measures this sensitivity. With low variance, the agents can perform the individual optimizations accurately without a large number of Monte-Carlo samples. The earlier COIN research explored the general concept of private utilities [38], providing several useful types. PC theory formalizes many of these concepts and provides a framework for developing problem specific utilities.

Two private utilities are typically used with the solution method: **Team Game** (TG) and **Wonderful Life Utility** (WLU). These utilities are defined as:

$$g_{TG_i}(x_i, x_{(i)}) = G(x_i, x_{(i)})$$

$$g_{WLU_i}(x_i, x_{(i)}) = G(x_i, x_{(i)}) - G(CL_i, x_{(i)}).$$

For TG, the private utility is simply the world utility. For WLU, the private utility is the world utility minus the world utility with the agent action "clamped" to the value CL_i. The choice of clamping value can strongly affect the performance [38], although clamping to the lowest probability action has been shown to be minimum variance [4]. Both of these utilities have zero bias. However, due to the subtracted term, WLU has much lower variance than TG. Other private utilities have also been explored. In particular, the **Aristocrat Utility** (AU) has been shown to be superior to WLU [12], but is often difficult to evaluate.

Often however, there is some known structure to the objective or constraint functions that can be exploited with a private utility. One example is one of the canonical problems in computer science, the k-Sat satisfiability problem [39]. Other examples are problems with objective functions that are the sum of multiple contributions. A problem with this type of structure is considered later in this work.

3 Delayed Sampling Experiments

3.1 Basic Solution Algorithm

The basic optimization algorithm consists of three major components, all of which can be performed in a distributed manner: Monte-Carlo sampling to obtain agent actions given their probability distributions, private utility evaluation for the agents for each joint sample, and update of the probability distributions. The latter step includes the regression to obtain the expected utility for each agent across its actions.

More precisely, the general algorithm proceeds as follows (detailed description of some of these steps are presented in the following subsections):

1. Initialize.
   - **(a)** Set the parameters: temperature T, probability update step size α, and data aging rate γ. For constrained problems set the Lagrange multiplier step size η.
   - **(b)** Assign the annealing schedule for T as a function of iteration number.
   - **(c)** Specify the number of Monte-Carlo samples m for each iteration.
   - **(d)** Assign agents to the variables in the problem, with their actions representing choices for values. Set the starting probabilities for each agent to uniform over its possible actions.

(e) Set the termination criteria. A number of possibilities exist, including: the number of iterations or convergence of the probabilities, objective values, or actions.

2. Minimize the Lagrangian.

   (a) Monte-Carlo sample the agent probabilities to obtain m IID (identically and independently distributed) joint actions.

   (b) For each sample,

   - Evaluate the objective function.
   - For each agent, compute the private utility.

   (c) Estimate the move-conditioned expected utilities for the agents for each of their possible moves using a regression. (An illustration is given below)

   (d) Update the probability distributions of each agent using these estimates according to Brouwer updating or Nearest Newton. (We ensure non-negative probabilities by setting all values that are smaller than 10^{-6} to 10^{-6} and then re-normalizing.

   (f) Update the parameters of the product distributions. Implement the assigned annealing schedule for T to update T. For continuous problems, update the regression window width τ.

   (g) Evaluate the termination criteria. If not satisfied, return to step 2(a), otherwise proceed to 3.

3. Final Evaluation.

   (a) Determine the highest probability action for each agent.

   (b) Evaluate the objective function with this set of actions.

3.1.1 Modifications for Continuous Variables

Many of the basic elements are easily extended to the continuous domain, such as the Monte-Carlo sampling, the use of private utilities, and the basic solution algorithm. Two aspects of the solution algorithm must, however, be modified. First, the integrals of the Maxent Lagrangian are approximated using a representation for the probability density across the domain of the continuous variable. Appropriate choice of the representation maintains the favorable properties of the update rules. Several possible representations have been considered [40].

The preferred approach is to represent the magnitudes at equally spaced locations across the variable domain using a trapezoidal approximation. Second,

since the sampling occurs for only a scattering of points across the range of the variables, a regression is necessary. In the current work, a simple regression based upon exponentially weighted averaging across the sampled values is used. The specific formulation is described in detail in Section 3.1.3.

3.1.2 Data Aging

A useful technique in the Monte-Carlo sampling is to allow previous samples to be re-used. This is accomplished by first "aging" the older data to reflect the fact that it was formed under a different q. For the discrete problems this is simple averaging with data aging controlled by the parameter γ. One can replace the empirical average for the most recent block k by

$$\mathrm{E}(g_i|x_i = j) = \frac{N_{ij}^{(k)}}{D_{ij}^{(k)}} = \frac{\sum_m g_i(x_i = j, x_{(i)})\delta(x_i = j) + \gamma N_{ij}^{(k-1)}}{\sum_m \delta(x_i = j) + \gamma D_{ij}^{(k-1)}},$$

where $\delta(x_i = j)$ equals 1 when $x_i = j$ and 0 otherwise and m is the number of Monte-Carlo samples in the block. To accomplish this requires storage of the matrices N_{ij} and D_{ij}, which have dimensions number of agents by number of moves. The resulting matrix of estimated expectations has these same dimensions.

Aging typically allows the number of samples per block m to be reduced, improving the performance of the minimization. For small m though, the most recent block may have no samples of some move $x_i = j$. This is only a concern if the move has not yet been sampled because D_{ij} would be undefined. One way to avoid this issue is to force the sampling of each move at the initial iteration. Care must be taken during the forced sampling to have the $x_{(i)}$ formed by sampling $q_{(i)}$. Another alternative is to average over just those k for which $G_{i,j}(k)$ exists.

3.1.3 Regressions

For the discrete case, as just discussed, the regression for each agent consists of averaging the utility values observed for each of its actions. Data aging can then be applied to re-use information from previous samples. In the case of continuous variables, however, the sampling occurs at only a scattering of points across the range of the variables. As a result, a more complex regression is necessary.

Given m sample pairs at iteration k, $\{x_j^k, g^k(x_j{}^k)\}$, $j = 1..m$, the regression provides estimates of $\mathrm{E}[g|x_i]$ for the x_i that are the locations parameterizing the distribution. In the current work, a simple regression based on exponentially weighted

averaging across the sampled values is used. The regression is given by,

$$\mathrm{E}[g^k|x_i] = \frac{N_i^k}{D_i^k} = \frac{\sum_m g^k(x_j{}^k) exp[-(x_i - x_j^k)^2/(2\tau^2)]}{\sum_m exp[-(x_i - x_j^k)^2/(2\tau^2)]}. \tag{9}$$

This formulation still allows data aging to be easily incorporated,

$$\mathrm{E}[g^k|x_i] = \frac{N_i^k + \gamma N_i^{k-1}}{D_i^k + \gamma D_i^{k-1}}, \tag{10}$$

with the parameter γ setting the aging rate.

One refinement to Eqn. 9 is to adapt the parameter τ automatically as the optimization proceeds. Since each agent i knows the probability distribution $q_i(x_i)$ from which its x_i are drawn, it is natural to set the regression window based upon some measure associated with the distribution. For the current work, the following procedure is applied to each of the variables:

- Given the number of samples m, determine their expected locations. To do this invert the cumulative probability density function at m evenly spaced intervals. To accomplish this, use the same subroutine that generates the Monte-Carlo samples.
- Determine the median spacing between adjacent expected locations.
- Set the value of τ as

  $$\tau = W_0 \, m \, \overline{\Delta x}, \tag{11}$$

  where $\overline{\Delta x}$ is the median spacing and W_0 is a scalar multiple, typically set to 0.2. This value proved appropriate for all the problems studied during this work. For problems involving higher numbers of variables, however, the parameter may need to be modified.

3.1.4 Annealing

The procedure for updating the temperature, referred to as the annealing schedule, plays an important role in the efficiency and reliability of the approach. If the temperature is reduced too quickly, the approach is more likely to find a local minimum. Too slowly, and a larger number of iterations, and therefore samples if Monte-Carlo sampling is used, are required. In the current work, a geometric schedule is typically applied. This involves multiplying the temperature by some fixed factor every few iterations. For the various applications within this work, the specific annealing schedule is provided with each. A minimum temperature is often also employed to insure that the solutions do not become delta functions.

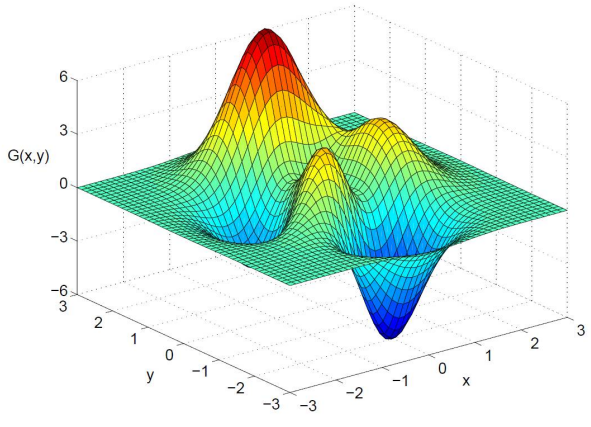


Figure 1: Peaks function.

An advantage of casting the problem directly in terms of the Maxent Lagrangian is that the explicit annealing schedule can be avoided. For example, the saddle point of the Lagrangian can be found by performing steepest ascent of $\mathcal{L}$ in the Lagrange parameter T while performing a descent in q. This type of approach was implemented in [39]. The opportunity to include other techniques from gradient based optimization [41] is also possible.

3.2 Example for a Continuous Problem

A simple two variable continuous problem is solved to show the various aspects of the continuous domain solution method. The function is shown in Figure 1, which consisted of several Gaussian peaks. The objective function was given by,

$$\begin{aligned} G(x,y) = {} & 3(1-x)^2 e^{-x^2-(y+1)^2} \\ & - 10(x/5 - x^3 - y^5)e^{-x^2-y^2} \\ & - 1/3e^{-(x+1)^2-y^2}. \end{aligned} \tag{12}$$

This is also known as the *Peaks* function in MATLAB. Note that, for uniform initial probability distributions, variable x can not see the global minimum until variable y changes its distribution away from uniform. As a result, cooperation was required between the variables to find the optimum.

Nearest Newton was used form this problem, with the parameter settings as listed in Table 1. While these values resulted in consistent convergence, they may not be the most efficient. For example, other values may result in fewer total function calls. The probability densities were represented using the trapezoidal approach with 200 points.

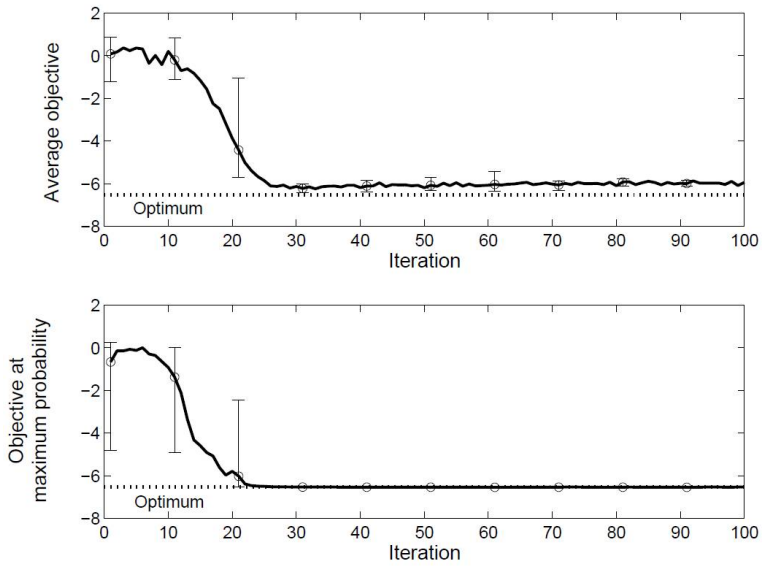


Figure 2: Convergence history of the objective for the *Peaks* function.

The overall convergence is consistent, as indicated by Figure 2. The two plots show the convergence of the sample average and the objective value at the maximum probability. The histories over 10 runs were averaged with error bars showing the minimum and maximum deviations. The convergence of the variables is shown in Figure 3. The results indicated that the technique quickly and consistently found the minimum, typically within 30 iterations. Although a small test problem, the quick convergence here with only 10 Monte-Carlo samples per iteration illustrated the potential of the approach.

The converged sample average was typically slightly above the optimum in Figure 2 because of the moderate lower bound on the temperature. A nonzero final temperature resulted in probability distributions that were centered about the optimum with some probability mass at less optimum values. As a result, when

Samples per iteration	10
Initial temperature, T_o	1
Final temperature, T_f	$5 \times 10^{-2} \times E[G]$
Annealing rate	0.25 every 10 iterations
Step size, α	0.1
Data aging rate, γ	0.5
Variable ranges	[-3, 3]

Table 1: Parameter settings for *Peaks* optimization.

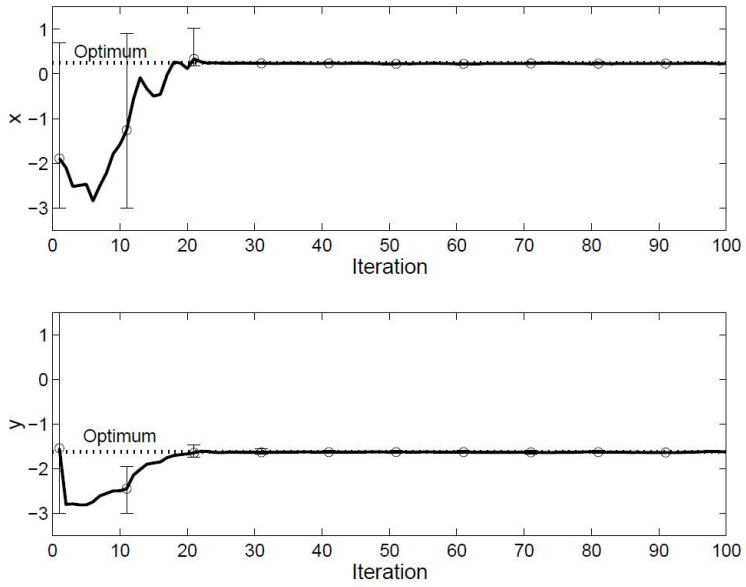


Figure 3: Convergence history of the variables for the *Peaks* function.

sampled, sometimes values slightly away from the optimum were obtained, reducing the overall average relative to the optimum. This is clear from Figure 4, which shows the converged probability distributions. This figure also illustrates another advantage of the approach, the sensitivity of the objective to each of the variables. Since the converged probabilities are related to the expected utilities through Eq. 7, given the probabilities $q_x(x)$, the expected utility, $\mathbb{E}(g_x|x)$ is obtained.

The performance of the approach was dependent upon the updating of the temperature T and the regression window widths τ_x and τ_y. The corresponding parameter histories for the *Peaks* optimizations are shown in Figure 5. While the temperature primarily followed its assigned schedule, the regression widths were updated automatically. As a result, smooth updates were obtained that closely reflected the convergence of the optimization.

3.3 Summary of Performance on Larger-Scale Problems

The approach has also been implemented on a number of larger-scale problems. In Bieniawski [9], the Probability Collectives approach using Nearest Newton was applied to three different problem domains: discrete (El Farol Bar Problem), discrete constrained (Bin-Packing), and constraint satisfaction (N-Queens). The performance was compared to a distributed reinforcement learning algorithm iden-

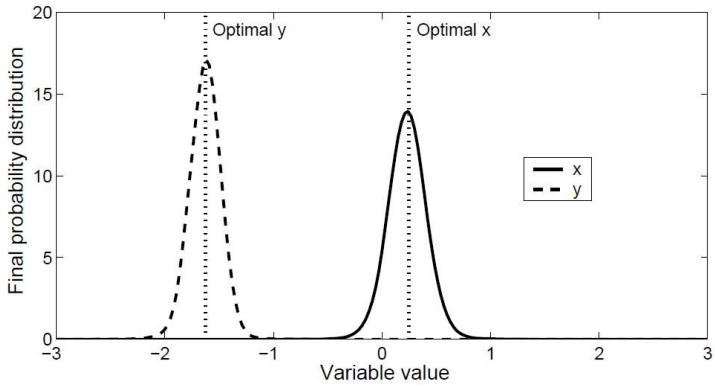


Figure 4: Final probability distributions for *Peaks* function.

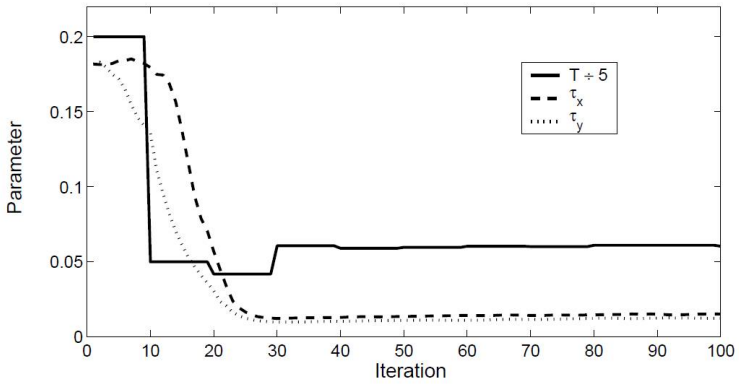


Figure 5: Parameter history for the *Peaks* function.

tical to Parallel Brouwer updating. This approach was at the core of the earlier COIN work, and the comparisons illustrated the improved performance, continued benefits of private utilities, and the extensions to constrained problems obtained with Probability Collectives. Antoine [11] applied the approach to aircraft routing, a constrained integer programming problem. The approach solved instances involving 129 variables and 184 constraints. Bieniawski [42] compared the performance of the different updating schemes, Serial Brouwer, Parallel Brouwer, and Nearest Newton on many instances of the Bin Packing Problem. Solutions within 1 bin of optimal were found for perfect packing problems involving 60 agents and 60 possible moves. Larger variants were also examined. Several variants of Serial Brouwer were investigated, including ones that attempted to reduce the cost of private utility evaluation. Nearest Newton proved to have the best performance of the approaches considered. A subset of these results are repeated later in this section. Bieniawski [10] investigated several aerospace related optimization problems, including a discrete structural sizing problem and a trajectory optimization. The latter results are repeated later in this section. The former addressed the 10-bar truss, a canonical discrete structural sizing problem. Macready [39] applied the solution approach to the canonical constraint satisfaction problem, k-Sat. Private utilities exploited the sparsity structure of the problem and enabled a completely distributed, analytical solution. Hard variants with 100 variables and 430 constraints were solved.

3.4 Discrete Optimization: Bin Packing Problem

The proposed approach was applied to several variants of the bin packing problem in order to explore its application to discrete constrained optimization problems. The bin packing problem consists of assigning N items of differing sizes into the smallest number of bins each with capacity c. For the current study instances were chosen which have a designed minimum number of bins [43] and were obtained from the OR-Library [44]. The instances consisted of 60 items to be packed in groups of three into 20 bins each of capacity 100. Since in general the minimum number of bins is not known, the move space of the agents was set to the number of items. The objective function for the optimization was selected as:

$$G = \begin{cases} \sum_{i=1}^{N} |x_i| & \text{if } x_i \leq c \\ \sum_{i=1}^{N} |x_i - c| & \text{if } x_i > c \end{cases} \tag{13}$$

where x_i is the total size of the items in bin i. Two approaches were explored to enforce the constraint on the number of bins: i) adding a quadratic penalty to the objective or ii) using the constraint augmented approach. For the latter constraints were added on the bin levels, $x_i \leq c$ for all i.

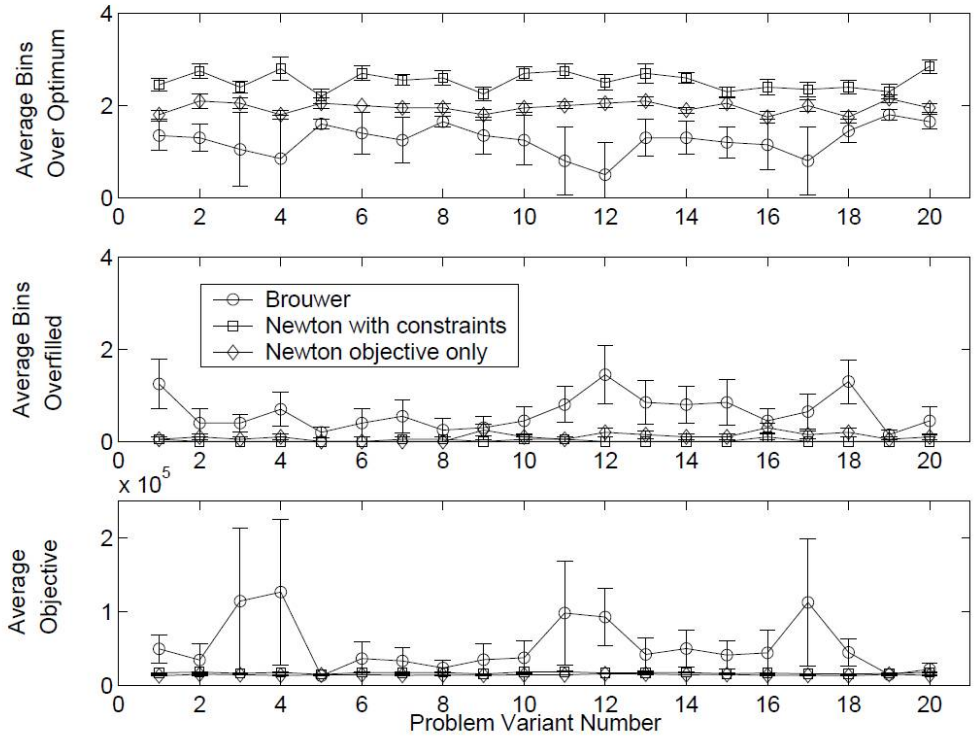


Figure 6: Comparing RL and PD theory methods for the bin packing problem.

Figure 6 shows the comparison between all three results. The first plot shows the average number of bins over the optimum, the second the number of bins over capacity, and the third plot the average objective function. Note that the unique optimum is no bins over the optimum and no bins over capacity. While Brouwer results in fewer bins it also results in more overfilled bins. In addition, the usage of the overfilled bins is much poorer than the Newton based schemes, as indicated by the average objective (lower is better). Both Newton schemes obtain similar average objectives, with the constrained version enforcing the capacity constraints by using extra bins.

3.5 Collectives for Control: Application to Stochastic Collocation

Collocation is an optimization approach for systems that include a time-domain simulation, such as control synthesis. In contrast to integrating the equations of motion at each iteration for a set of the design variables, in collocation the opti-

mization iteratively solves the simulation along with the design variables. The result is a much larger optimization problem, as the numerical integration in the simulation is replaced by constraints in the optimization. The advantage is a smooth, well-posed, and, most importantly, sparse optimization, even for problems that involve significant nonlinearities. This approach has typically found application in spacecraft and rocket trajectory problems [45], but it has also been applied to coupled design of an aircraft and its control laws [46].

For example, control design can be formulated as the following optimization problem,

$$\begin{aligned} \text{minimize:} \quad & h(x), \\ \text{with respect to:} \quad & x \text{ and } K, \\ \text{subject to:} \quad & \dot{x} = f(x, u), \\ \text{where:} \quad & u = g(x, K). \end{aligned}$$

The objective function h, control policy g, and system dynamics f can be arbitrary. Numerous techniques are used to approximate the system dynamics to obtain constraint equations in a form compatible with an optimizer [46]. The key advantage is that, although the problem size is now much larger since it includes the system states x, it is sparse. The collectives approach can solve this optimization efficiently by exploiting the sparse nature of the problem using a private utility. There is also the potential for collectives to extend collocation methods to account for noise and disturbances.

To illustrate the collectives approach applied to collocation, a classical calculus of variations problem, the Brachistochrone [47, 48], is solved. The objective is to find the minimum-time trajectory between two points for an object moving only

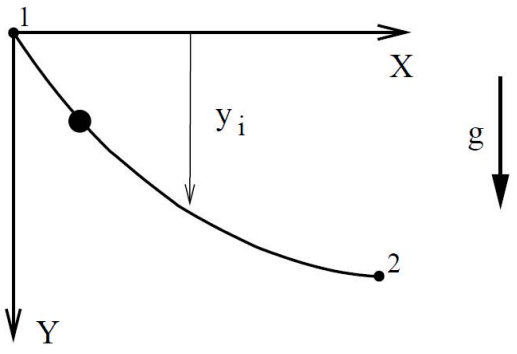


Figure 7: The Brachistochrone problem.

under the influence of gravity, Figure 7. Following [47] the objective function is:

$$G = t_{12} = \int_{x_1,y_1}^{x_2,y_2} f \, dx,$$

where

$$f = (1 + (dy/dx)^2)^{1/2} \, (2gy)^{1/2}.$$

Although this problem can be solved analytically, it provides a useful benchmark for evaluating optimization methods. To apply optimization techniques, variables are defined as the vertical location of points along the trajectory. The integral and the derivatives in the objective function are then approximated using these locations. A trapezoidal approximation is made to the integral at N points and a central finite difference is used for the derivative. This results in the following optimization problem,

$$\min_{\vec{y}} G = \frac{\Delta x}{2} \left[f_0 + 2f_1 + ... + 2f_N - 1 + 2f_N \right],$$

where, for the interior points,

$$f_i = (1 + [\frac{1}{2\Delta x}(y_{i+1} - y_{i-1}]^2)^{1/2} \, (2gy_i)^{1/2}. \tag{14}$$

For the boundary points, f_0 and f_N, forward or backward approximations are used for the derivatives.

This optimization was solved by a commercially available gradient based optimizer [49] and by the collectives approach. Collectives are particularly applicable to objectives such as Eqn. 14 due to sparse interactions between the agents representing the variables. The objective was a sum of contributions, a structure for which a private utility can be developed. Since the individual contributions to the objective f_i were only functions of a single variable and its neighbors, a suitable private utility was

$$\begin{aligned} g_i(y_{i-2}, y_{i-1}, y_i, y_{i+1}, y_{i+2}) = \frac{\Delta x}{2} [& 2f_{i-1}(y_{i-2}, y_{i-1}, y_i) \\ & + 2f_i(y_{i-1}, y_i, y_{i+1}) \\ & + 2f_{i+1}(y_i, y_{i+1}, y_{i+2})]. \end{aligned} \tag{15}$$

Eqn. 15 was used for the interior agents, while similar private utilities were obtained for the first and last agents. Note that this private utility has no bias since it includes all the dependencies of the objective upon agent i. This type of private utility scales well with increasing numbers of agents since the number of dependencies for each agent remains constant. Since the dependencies were known,

only the other agents' probabilities, indicated in Eqn. 15, were required to evaluate the expectations. This suggested two possible variants of the collectives approach. The only difference between the variants is the calculation of the expected utilities. First, sampling could be used as before, but with the agents passing their moves directly to one another so they can be evaluated according to Eqn. 15. This is also equivalent to having all the agents pass their moves to a centralized entity that computes the various contributions to the objective f_i. Each agent then knows that it contributes to its own f_i and to its neighbors, f_{i-1} and f_{i+1}, using the sum of these three as its private utility. For the agents located on the edges, appropriate modifications are made. Second, the agents could pass their probability distributions to one another, which can then be integrated along with Eqn. 15 to obtain the expected private utilities. This variant is now deterministic, since no sampling is involved. Results for the second approach are shown here. See [40, 10] for the results for the first approach.

If the agents share their probability distributions each agent now computes its conditional expected utility exactly. Figure 8 compares the iteration history for the different solution approaches with this implementation. The left plot shows the objective function, while the right shows the convergence criteria. The criteria used here is the 2-norm of the change in the probability distributions between iterations. The convergence of the various approaches look very similar, and are very rapid. Serial Brouwer converged the fastest, with Parallel Brouwer finding a better optimum slightly faster than Nearest Newton. Serial Brouwer performed better since the agents performed their updates in sequence, rather than all at once as with Parallel Brouwer or Nearest Newton. As a result, the agent being updated received the most recent information from its just updated upstream neighbor. This can be implemented at no extra cost since, with no sampling, they were computationally equivalent.

The optimal trajectories for this variant are compared with the gradient-based and analytical solutions in Figure 9. There is no appreciable difference between the discretized trajectories. The converged probability distributions for the agents are shown in Figure 10. The figure shows the probability distributions across possible Y values for the agents versus their position X. The distributions provide sensitivity information through their relationship to the objective via the Boltzmann distribution, Eqn. 7. This was obtained without additional computational cost. The figure illustrates the higher importance of the agents closest to the starting and ending points. This partially explains differences seen between the optimal trajectories for the sampled variant. From Figure 10 it is evident that the objective was less sensitive to the positions in the middle of the trajectory. As a result, the optimizer was slower to obtain the optimal values for these positions.

Another advantage of this approach, which still remains to be explored, is the ability to treat stochastic boundary conditions. Since these types of bound-

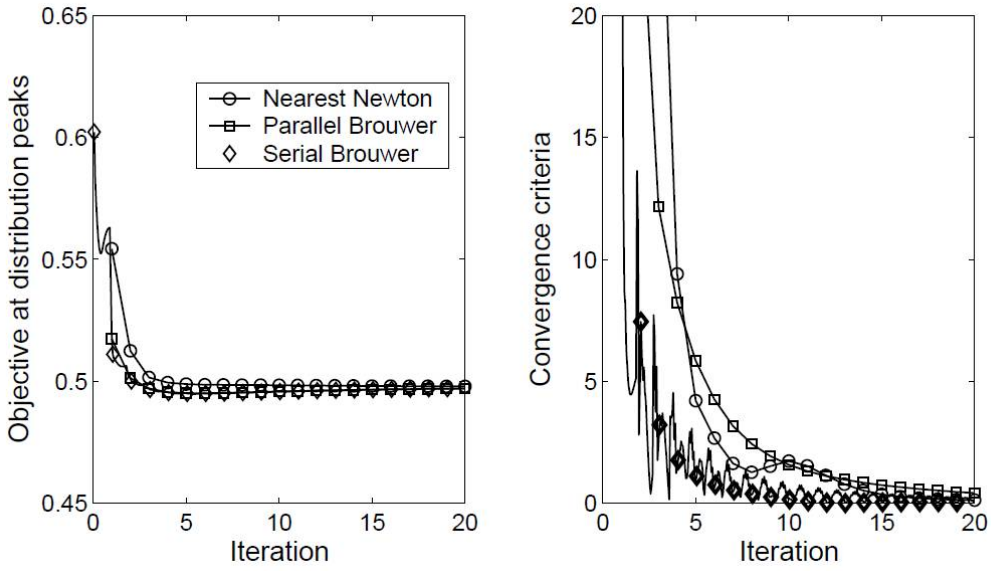


Figure 8: Convergence history for Brachistochrone without sampling.

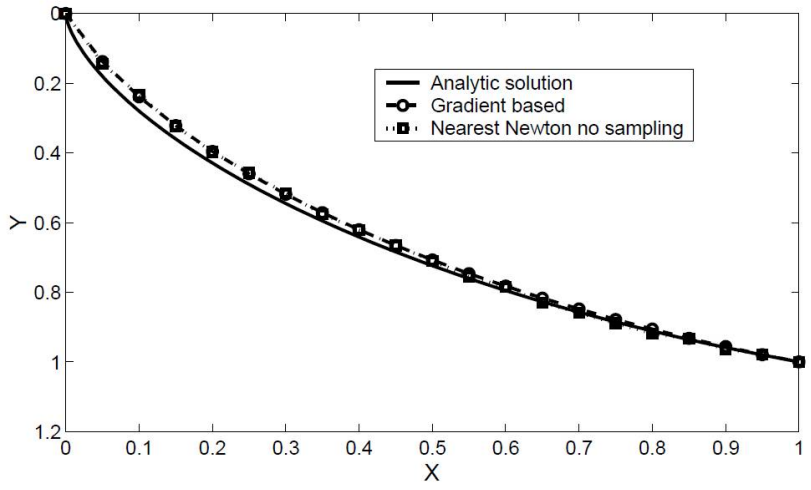


Figure 9: Final trajectories for the Brachistochrone without sampling.

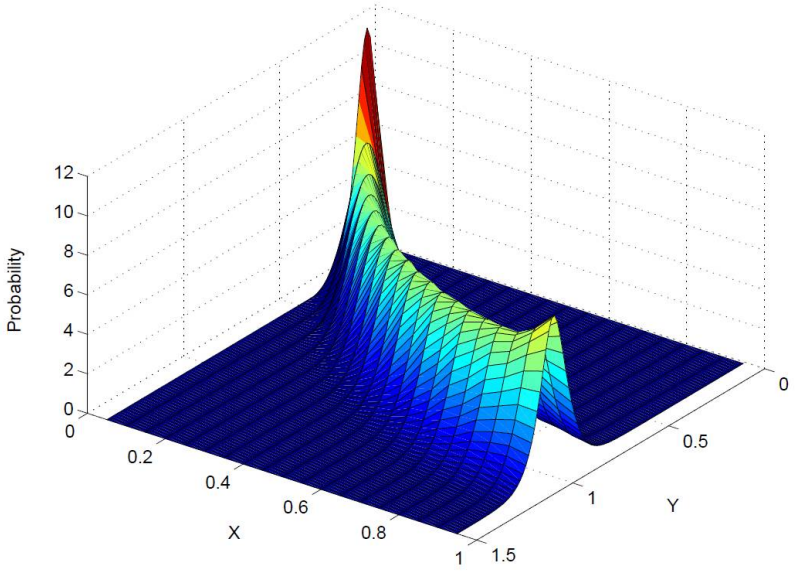


Figure 10: Converged probability distributions for the Brachistochrone.

ary conditions are typically represented as probability distributions, they can be incorporated as additional agents whose probabilities are not updated. These distributions could even be provided as needed by other agents to remove the need for sampling.

4 Immediate Sampling Theory

We now move on to immediate sampling PC. Like delayed sampling PC, it involves updating a probability distribution based on samples of that distribution. But unlike delayed sampling PC, in immediate sampling there are no restrictions on the class of probability distributions being considered.

4.1 Monte Carlo Optimization

Consider the problem

$$\text{(P1)}: \qquad \text{argmin}_{\phi\in\Phi} \int dw\ U(w,\phi).$$

where even if we know the analytic form of $U(.,.)$, we cannot analytically perform the minimization. Monte Carlo Optimization (MCO) [50] is a computational way

to search for the solution of (P1). In MCO we use importance sampling to rewrite the integral in (P1) as

$$\begin{aligned} \int dw\ U(w,\phi) &= \int dw\ v(w)\frac{U(w,\phi)}{v(w)}. \\ &\triangleq \int dw\ v(w) r_{v,U,w}(\phi), \end{aligned} \tag{16}$$

for some sampling distribution v, where it is assumed that we can evaluate $v(w)$ for any w, up to an overall normalization constant.. Following the usual importance sampling procedure, we then IID sample v to form a sample set $\{U(w^i,.) : i = 1,\ldots N\}$. These specify a set of N sample functions from $\phi \rightarrow \mathbb{R}$:

$$r^i(\phi) \triangleq r_{v,U,w^i}(\phi).$$

In MCO, these N functions are used in combination with any prior information to estimate the solution to (P1). Conventionally, this is done by approximating the solution to (P1) with the solution to the problem

$$\text{(P2)}: \qquad \text{argmin}_\phi \sum_i r^i(\phi)$$

which can often be carried out computationally. We will use the term **naive MCO** to refer to solving (P2) this way.

Note that even though $\sum_i r^i(\phi)$ is an unbiased estimate of $\int dw\ U(w,\phi)$ for any *fixed* ϕ, its minimizer over all ϕ is not an unbiased estimate of $\min_{\phi\in\Phi} \int dw\ U(w,\phi)$. This is because searching for the minimizing ϕ introduces bias; the minimizer over ϕ of $\sum_i r^i(\phi)$ is less than $\min_{\phi\in\Phi} \int dw\ U(w,\phi)$. Accordingly $\text{argmin}_\phi \sum_i r^i(\phi)$ is typically a poor estimate of $\text{argmin}_{\phi\in\Phi} \int dw\ U(w,\phi)$. This suggest that one should use some other technique than naive MCO to estimate $\text{argmin}_{\phi\in\Phi} \int dw\ U(w,\phi)$. In particular, one might want to introduce some variance in the estimator if doing so sufficiently reduces the bias.

4.2 Naive Immediate Sampling

We now show how to cast the immediate sampling optimization problem as an instance of MCO. Use importance sampling to write the integral transform con-

sidered in immediate sampling, Eq. (1), as

$$\begin{aligned}
\mathscr{F}_{\mathscr{G}}(q_\theta) &\triangleq \int dx dg\ P(g \mid x, \mathscr{G}) F(g, q_\theta(x)) \\
&= \int dx\ h(x) \frac{\int dg\ P(g \mid x, \mathscr{G}) F(g, q_\theta(x))}{h(x)} \\
&= \int dx dg\ h(x) P(g \mid x, \mathscr{G}) \frac{F(g, q_\theta(x))}{h(x)} \qquad (17) \\
&\triangleq \int dx\ h(x) r_{P(g|x,\mathscr{G}),h}(\theta). \qquad (18)
\end{aligned}$$

where we call $h(x)$ the **sampling distribution**.

We form a **sample set** of N pairs $D^1 \equiv \{x^i, g^i\}$ by IID sampling the distribution $h(x)P(g \mid x, \mathscr{G})$ in the integrand of Eq. 17. (That sampling is the 'immediate' Monte Carlo process.) Note that D^1 is equivalent to a set of N **sample functions** from θ to $\mathbb{R}$:

$$r_h^i(x^i, \theta) \triangleq \frac{F(g^i, q_\theta(x^i))}{h(x^i)} \ : \ i = 1, \ldots N.$$

In the simplest version of immediate sampling, we use the functions $r_h^i(x^i, \theta)$, together with our prior knowledge (if any), to estimate the θ that minimizes $\mathscr{F}_{\mathscr{G}}(q_\theta)$. As an example, not using any prior knowledge, we could estimate $\mathscr{F}_{\mathscr{G}}(q_\theta)$ for any θ as

$$\int dx\ h(x) r_{P(g|x,\mathscr{G}),h}(\theta) \approx \frac{\sum_i r_h^i(x^i, \theta)}{N}. \qquad (19)$$

For any θ, this estimate is both an unbiased estimate of $\mathscr{F}_{\mathscr{G}}(q_\theta)$ and the maximum likelihood estimate of $\mathscr{F}_{\mathscr{G}}(q_\theta)$. Moreover, in the usual way with importance sampling, for a well-chosen sampling distribution this estimate will be quite accurate. Accordingly, to estimate the θ that minimizes $\mathscr{F}_{\mathscr{G}}(q_\theta)$ we could simply search for the θ that minimizes $\sum_i r_h^i(x^i, \theta)$.[5] However this procedure is essentially an example of naive MCO. So in particular, it would be expected to suffer from the bias problems discussed in Sec. 4.1.

[5] In practice, it may make sense to divide the sum in Eq. 19 by $\sum_i 1/h(x^i)$, to reduce the variance of our estimate, and then find the θ that optimizes that ratio of sums, rather than the θ that just optimizes the numerator term [see 51]. For example, this can be helpful when one uses cross-validation to set β, as described below.

4.3 Parametric Machine Learning

Parametric Machine Learning (PL) is a well-developed field with techniques for both supervised and unsupervised parametric learning. Broadly speaking, it seeks the solution to problems of the form

$$\text{(P3)}: \qquad \text{argmin}_{\xi \in \Xi} \int dx\ P(x) R_x(\xi).$$

for some appropriate space x, parameter set Ξ, and function R.

For example in supervised learning, $x = (x_1, x_2)$ where x_1 is the "input variable" and x_2 is the "output variable", a "test set" is formed by sampling $P(x)$, there is a real-valued "loss function" $L(x_2, x_2')$, and we are interested in finding the parameter ξ that minimizes

$$\int dx\ P(x_1) P(x_2 \mid x_1) L(f_\xi(x_1), x_2) \Big] \tag{20}$$

over some parameterized set of "data fits", $\{f_\xi\} : x_1 \to x_2$. This is of the form (P3) if we define $R_x(\xi) \triangleq L(f_\xi(x_1), x_2)$.

To perform the minimization in (P3), in PL we have a "training set" comprising N values. For example, in standard supervised learning, the training set is N pairs (x_1^i, x_2^i) formed by IID sampling $P(x)$. Although not usually viewed this way, the training set can be re-expressed as a set of N functions from $\Xi \to \mathbb{R}$, $D \equiv \{R_{x^i}(\xi)\}$.

One approach to this minimization would first make the maximum likelihood estimate

$$\begin{aligned} \int dx\ P(x) R_x(\xi) &\approx \frac{1}{N} \sum_i R_{x^i}(\xi), \\ &\triangleq \frac{1}{N} \sum_i R^i(\xi). \end{aligned} \tag{21}$$

One would then solve for the ξ minimizing the sum, and use this as an approximation to the solution to P3. In practice, though, this procedure is seldom used directly: although the approximation in Eq. 21 is unbiased for any fixed ξ, $\min_\xi \sum_i R^i(\xi)$ is not an unbiased estimate of $\min_\xi \int dx\ P(x) R_x(\xi)$. (This phenomenon is called "over-fitting".) Therefore in PL this approximation is modified, to incorporate bias-reduction techniques.[6]

[6]Note that most sampling theory analysis of PL does not directly consider the biases and variances of the separate Monte Carlo estimators for each ξ. Rather, it considers a different type of bias and variance — the bias and variance of an entire algorithm that chooses a ξ based on associated MC estimates of expected loss. See [52] for a discussion of the relation between the two types of bias and variance.

MCO	PL
w	x
ϕ	ξ
$v(w)$	$P(x)$
$r_{v,w}(\phi)$	$R_x(\xi)$
$r^i_v(\phi)$	$R^i(\xi)$

Table 2: Correspondence between variables in PL and MCO.

4.4 PL Equals MCO

The preceding analysis suggests that the general MCO problem of extrapolation from a sample set of empirical functions to minimize the integral of Eq. 16 is identical to the general PL problem of extrapolation from a training set of empirical functions to minimize the integral in (P3). To establish this formally, make the identifications in Table 2, which result in the integrals in Eq. 16 and (P3) becoming identical.

This correspondence formally establishes that just as direct minimization of Eq. 21 is biased, naive MCO will be biased. However in PL many techniques have been found for addressing this problem of bias, and more generally for finding the optimal fit. These include in particular regularization (to reduce bias), cross-validation (to set hyperparameters of the regularization), bagging, and stacking.

The correspondence of Table 2 shows how to apply these PL techniques to MCO rather than PL, and thereby improve on naive MCO. Since immediate sampling is just a special case of MCO, we can also apply such PL techniques to improve naive immediate sampling PC. Many experiments have validated the fast optimization of such immediate sampling. In the next subsection we summarize a few of them.

5 Immediate Sampling Experiments

In this section, we demonstrate the application of PL to immediate-sampling PC for unconstrained optimization of both deterministic and nondeterministic functions $P(g \mid x, \mathscr{G})$ over real-valued x's. We first describe our choice of $\mathscr{F}_{\mathscr{G}}$, which will be pq KL distance. Next, as an illustrative example, we apply immediate sampling for this choice to an extremely simple optimization problems, where the objective $P(g \mid x, \mathscr{G})$ is deterministic, simply being a 2-D quadratic function. Subsequently, we apply it to deterministic and stochastic versions of two well-known unconstrained optimization benchmarks, the Rosenbrock function and the Woods function.

In going through these examples we highlight the use of PL techniques to enhance optimizer performance. In particular, we show that cross-validation for regularization yields a performance improvement of an order of magnitude. We then show that cross-validation for model-selection results in improved performance, especially in the early stages of the algorithm. We also show that bagging can yield significant improvements in performance.

5.1 Minimizing pq KL Distance

As described in Sec. 2.1, a good choice of objective function for delayed sampling PC is qp KL distance where p is the Boltzmann distribution over G values, $p(x) \propto exp(-G(x)/T)$. This qp KL distance distance can also be formulated as the objective $\mathscr{F}_{\mathscr{G}}$ of immediate sampling, by choosing

$$F(g, q_\theta(x)) \triangleq g q_\theta(x) + q_\theta(x) \ln[q(x)]/T$$

(cf. Eq. (1).) However in immediate sampling, rather than use the qp KL distance for this choice of Boltzmann p, it is typically better to use the pq KL distance,

$$\mathrm{KL}(p||q) = \int dx\, p(x) \ln\left(\frac{p(x)}{q(x)}\right)$$

as discussed in [26]. (The immediate sampling objective function can be formulated as this KL distance by choosing $F(g, q(x)) \triangleq exp(-g/T)\ln[q(x)]$, up to an irrelevant q-independent overall multiplicative constant and an irrelevant q-independent additive constant.)

It is easy to show that when there are no restrictions on q, pq KL distance is minimized (to 0) by taking $p = q$. However, because we have a finite amount of data, we want to restrict q to some subset of all distributions. We then parametrize elements of that class by θ, and write elements of the class as q_θ. So minimizing pq KL distance is equivalent to the following cross-entropy minimization problem:

$$\begin{array}{ll} \text{minimize over } \theta & -\int dx\, p(x) \ln\left(q_\theta(x)\right), \\ \text{where} & \int dx\, q_\theta(x) = 1, \\ & q_\theta(x) \geq 0 \quad \forall x. \end{array} \tag{22}$$

5.1.1 Gaussian Densities

If q_θ is log-concave in its parameters θ, the minimization problem (22) is a convex optimization problem. In particular, consider the case where $X = \mathbb{R}^n$, and q_θ is a multivariate Gaussian density, with mean μ and covariance Σ, parametrized as follows,

$$q_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right),$$

then the optimal parameters are given by matching first and second moments of p and q_θ.

$$\begin{aligned} \mu^\star &= \int dx\, x\, p(x), \\ \Sigma^\star &= \int dx\, (x-\mu^\star)(x-\mu^\star)^T p(x). \end{aligned}$$

It is easy to generalize this to the case where $X \subset \mathbb{R}^n$, by making a suitable modification to the definition of p. This is described in Sec. 5.2.1.

5.1.2 Immediate Sampling with a Single Gaussian

Using importance sampling, we can convert the cross-entropy integral in Eq. 22 to a sum over data points, as follows.

$$\frac{1}{N} \sum_D \frac{p(x^i)}{h(x^i)} \ln\left(q_\theta(x^i)\right),$$

where D is the data set $\{(x^i, g^i)\}, i = 1, \ldots, N$. This sets up the minimization problem for immediate sampling for pq KL distance.

$$\text{minimize} \qquad -\sum_D \frac{p(x^i)}{h(x^i)} \ln\left(q_\theta(x^i)\right). \tag{23}$$

Denote the likelihood ratios by $s^i = p(x^i)/h(x^i)$. Differentiating Eq. 23 with respect to the parameters μ and Σ^{-1} and setting them to zero yields the following formulas:[7],[8]

$$\begin{aligned} \mu^\star &= \frac{\sum_D s^i x^i}{\sum_D s^i} \\ \Sigma^\star &= \frac{\sum_D s^i (x^i-\mu^\star)(x^i-\mu^\star)^T}{\sum_D s^i} \end{aligned}$$

5.1.3 Mixture Models

The single Gaussian is a fairly restrictive class of models. Mixture models can significantly improve flexibility, but at the cost of convexity of the KL distance minimization problem. However, a plethora of techniques from supervized learning, in particular the Expectation Maximization (EM) algorithm, can be applied with minor modifications.

[7] As one would expect, in the infinite-data limit these formulas are identical to the moment-matching results for the full-blown integral case.

[8] Note that these formulas resemble those for MAP density estimation, often used in supervised learning to find the MAP parameters of a distribution from a set of samples. The difference in this case is that each sample point is weighted by the likelihood ratio s^i, and is equivalent to 'converting' samples from h to samples from p.

Suppose q_θ is a mixture of M Gaussians, that is, $\theta = (\mu, \Sigma, \phi)$ where ϕ is the mixing p.m.f, we can view the problem as one where a hidden variable z decides which mixture component each sample is drawn from. We then have the optimization problem

$$\text{minimize } -\sum_D \frac{p(x^i)}{h(x^i)} \ln\left(q_\theta(x^i, z^i)\right).$$

Following the standard EM procedure, we multiply and divide the quantity inside the logarithm by some $Q_i(z^i)$, where Q_i is a distribution over the possible values of z^i. As before, let s^i be the likelihood ratio of the i'th sample.

$$\text{minimize } -\sum_D s^i \ln\left(\sum_{z^i} Q_i(z^i) \frac{q_\theta(x^i, z^i)}{Q_i(z^i)}\right)$$

Then using Jensen's inequality, we can take Q_i outside the logarithm to get a lower bound. To make this lower bound tight, choose $Q_i(z^i)$ to be the constant $p(z^i|x^i)$. Finally, differentiating with respect to μ_j, Σ_j^{-1} and ϕ_j gives us the EM-like algorithm:

$$\begin{aligned}
\text{E-step: For each i, set} \quad Q_i(z^i) &= p(z^i|x^i), \\
\text{that is,} \quad w_j^i &= q_{\mu,\Sigma,\phi}(z^i = j|x^i), \quad j = 1, \ldots, M. \\
\text{M-step: Set} \quad \mu_j &= \frac{\sum_D w_j^i s^i \, x^i}{\sum_D w_j^i s^i}, \\
\Sigma_j &= \frac{\sum_D w_j^i s^i \, (x^i - \mu_j)(x^i - \mu_j)^T}{\sum_D w_j^i s^i}, \\
\phi_j &= \frac{\sum_D w_j^i s^i}{\sum_D s^i},
\end{aligned}$$

Since this is a nonconvex problem, one typically runs the algorithm multiple times with random initializations of the parameters.

5.2 Implementation Details

In this section we describe the implementation details of an iterative immediate-sampling PC algorithm that uses the Gaussian mixture models described in the previous section to minimize pq KL distance to a Boltzmann target parametrized by $\beta \equiv 1/T$. We also describe the modification of a variety of techniques from parmetric learning that significantly improve performance of this algorithm. An overview of the procedure is presented in Algorithm 1.

5.2.1 Example: Quadratic $G(x)$

Algorithm 1 Overview of *pq* minimization using Gaussian mixtures

1: Draw uniform random samples on X
2: Initialize regularization parameter β
3: Compute $G(x)$ values for those samples
4: **repeat**
5: Find a mixture distribution q_θ to minimize sampled *pq* KL distance
6: Sample from q_θ
7: Compute $G(x)$ for those samples
8: Update β
9: **until** Termination
10: Sample final q_θ to get solution(s).

Consider the 2-D box $X = \{x \in \mathbb{R}^2 \quad | \quad \|x\|_\infty < 1\}$. Consider a simple quadratic on X,

$$G_Q(x) = x_1^2 + x_2^2 + x_1 x_2, \quad x \in X.$$

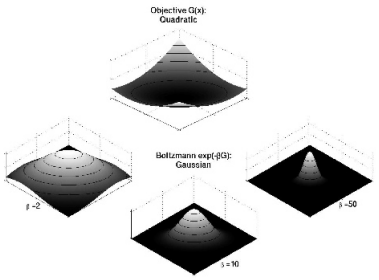


Figure 11: Quadratic $G(x)$ and associated Gaussian targets

The surface and contours of this simple quadratic on X are shown in Fig. 11. Also shown are the corresponding Boltzmann target distributions p^β on X, for $\beta = 2, 10$ and 50. As can be seen, as β increases, p^β places increasing probability mass near the optimum of $G(x)$, leading to progressively lower $\mathbb{E}_{p^\beta} G(x)$. Also note that since $G(x)$ is a quadratic, $p^\beta(x) \propto \exp(-\beta G(x))$ is a Gaussian, restricted to X and renormalized. We now 'fit' a Gaussian density q_θ to the Boltzmann p^β by minimizing $\mathrm{KL}(p^\beta \| q_\theta)$, for a sequence of increasing values of β. Note that q_θ is a distribution over $\mathbb{R}^2$, and G_Q is not defined everywhere in $\mathbb{R}^2$. Therefore, we extend the definition of G_Q to all of $\mathbb{R}^2$ as follows.

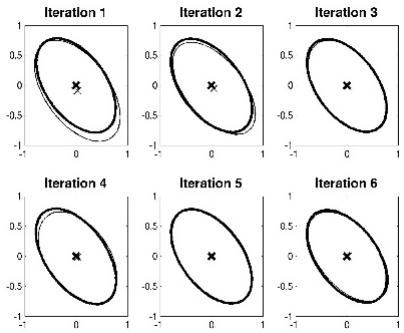
$$G_Q(x) = \begin{cases} x_1^2 + x_2^2 + x_1 x_2, & x \in X. \\ \infty & \text{otherwise.} \end{cases}$$

Now $p^\beta = 0$ for all $x \notin X$, and the integral for KL distance can be reduced to an integral over X. This means that samples outside X are not considered in our computations.

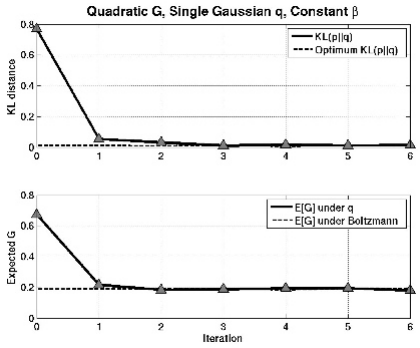
5.2.2 Constant β

First, we fix $\beta = 5$, and run a few iterations of the PC algorithm. To start with, we draw $N_j = 30$ samples from the uniform distribution on X. The best-fit Gaussian

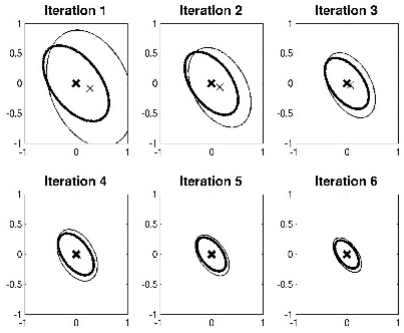
is computed using the immediate sampling procedure outlined in the preceding section. At each successive iteration, $N_j = 30$ more samples are drawn from the current q_θ and the algorithm proceeds. A total of 6 such iterations are performed. The 90% confidence ellipsoids corresponding to p^β (heavy line) and the iterates of q_θ (thin line) are shown in Fig. 12. Also shown are the corresponding values of $\mathbb{E}_{q_\theta} G(x)$, computed using the sample mean of $G_Q(x)$ for 1000 samples of x drawn from each q_θ, and $\mathrm{KL}(p^\beta \| q_\theta)$, computed as the sample mean of $\ln(p^\beta(x)/q_\theta(x))$ for 1000 samples of x drawn according to p^β.



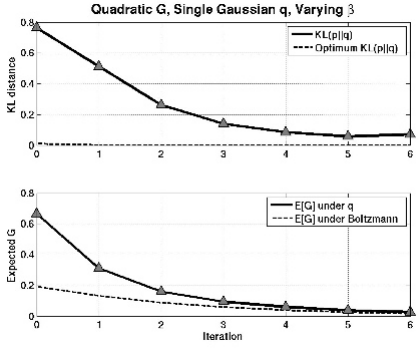
(a) Constant β: Confidence ellipsoids



(b) Constant β: KL distance and expected G



(c) Varying β: Confidence ellipsoids



(d) Varying β: KL distance and expected G

Figure 12: PC iterations for quadratic $G(x)$.

5.2.3 Varying β

Next, we change β between iterations, in the 'update β' step shown in algorithm(1). With the same algorithm parameters, we start with $\beta = 10$, and at each iteration, we use a multiplicative update rule $\beta \leftarrow k_\beta \beta$, for some constant $k_\beta > 1$, in this case, 1.5. As the algorithm progresses, the increasing β causes the target density p^β to place increasing probability mass on regions with low $G(x)$,

as shown in Fig. 11. Since the distributions q_θ are best-fits to p, successive iterations will generate lower $\mathbb{E}_{q_\theta} G(x)$. The 90% confidence ellipsoids and evolution of $\mathbb{E}_{q_\theta} G(x)$ and KL distance are shown in Fig. 12.

5.2.4 Cross-validation to Schedule β

In more complex problems, it may be difficult to find a good value for the β update ratio k_β. However, we note that the objective $\mathrm{KL}(p^\beta \| q_\theta)$ can be viewed as a regularized version of the original objective, $\mathbb{E}_{q_\theta}[G(x)]$. Therefore, we use the standard PL technique of cross-validation to pick the regularization parameter β from some set $\{\beta\}$.

More precisely, at each iteration, we partition the data set D of all samples $\{(x^i, g^i\}$ found so far, into training and test data sets, indicated by D_T and D_V respectively. Then for each β in some candidate set $\{\beta\}$, we find the optimal parameters $\theta^\star(\beta)$ using only the training data D_T. Next, we test the associated $q_{\theta^\star(\beta)}$ on the test data D_V using the following performance measure.

$$\widehat{g}(\theta) = \frac{\sum_{D_V} \dfrac{q_\theta(x^i) G(x^i)}{h(x^i)}}{\sum_{D_V} \dfrac{q_\theta(x^i)}{h(x^i)}}, \tag{24}$$

The objective $\widehat{g}(\theta)$ is an estimate[9] of the unregularized objective $\mathbb{E}_{q_\theta}[G(x)]$. Finally, we set $\beta^\star = \arg\min_{\beta \in \{\beta\}} \widehat{g}(\theta^\star(\beta))$, and compute $\theta^\star(\beta^\star)$ using *all* the data D. Note that the whole cross-validation procedure is carried out without any more calls to the oracle $\mathscr{G}$.

We demonstrate the functioning of cross-validation on the well-known Rosenbrock problem in two dimensions, given by

$$G_R(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,$$

over the region $X = \{x \in \mathbb{R}^2 \mid \|x\|_\infty < 4\}$. The optimum value of 0 is achieved at $x = (1, 1)$. The details of the cross-validation algorithm used are presented in Algorithm 2. For this experiment, we choose

$$\begin{array}{rrrrrr} \texttt{maxExtIter} = & 4, & k_1 = & 0.5, & k_2 = & 2, \\ N_j = & 10, & n_\beta = & 5, & K = & 10. \end{array}$$

[9]The reason for dividing by the sum of $q(x^i)/h(x^i)$ is as follows. If the training data is such that no probability mass is placed on the test data, the numerator of $\widehat{g}_{q_\theta}$ is 0, regardless of the parameters of q_θ. In order to avoid this peculiar problem, we divide by the sum of $q(x^i)/h(x^i)$, as desribed by **(author?)** [51].

Algorithm 2 Cross-validation for β.

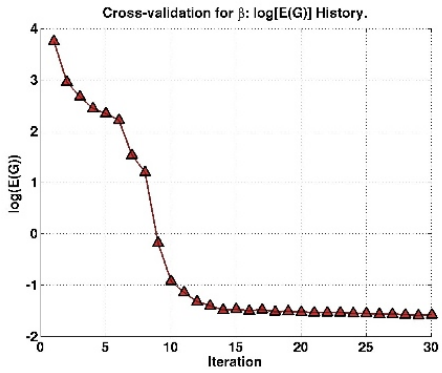
Initialize interval extension count `extIter` = 0, and `maxExtIter` and β_0..
repeat
  At $\beta = \beta_0$, consider the interval $\Delta\beta = [k_1\beta_0, k_2\beta_0]$.
  Choose $\{\beta\}$ be a set of n_β equally-spaced points in $\Delta\beta$.
  Partition the data into K random disjoint subsets.
  for each fold k, **do**
    Training data is the union of all but the k^{th} data partitions.
    Test data is the k^{th} partition.
    for β_i in $\{\beta\}$, **do**
      Use training data to compute optimal parameters $\theta^\star(\beta_i, D_{T_k})$.
      Use test data to compute held-out performance $\widehat{g}(\theta^\star(\beta_i, D_{V_k}))$, from Eq. 24.
    end for
  end for
  Compute average held-out performance, $\overline{g}(\beta_i)$, of $\widehat{g}(\theta^\star(\beta_i, D_{V_k}))$.
  Fit a quadratic $Q(\beta)$ in a least-squares sense to the data $(\beta_i, \overline{g}(\beta_i))$.
  if Q is convex **then**
    Set optimum regularization parameter $\beta^\star = \arg\min_{\beta\in\Delta\beta} Q(\beta)$.
  else
    Fit a line $L(\beta)$ in a least-squares sense to the data $(\beta_i, \overline{g}(\beta_i))$.
    Choose $\beta^\star = \arg\min_{\beta\in\Delta\beta} L(\beta)$.
  end if
  Increment `extIter`
  Update $\beta_0 \leftarrow \beta^\star$
until `extIter` > `maxExtIter` or Q is convex.

The histories of $\mathbb{E}_q G(x)$ and β are shown in Fig. 13. Also shown are plots of the fitted $Q(\beta)$ at iterations 8 and 15. As can be seen, the value of β sometimes *decreases* from one iteration to the next, which can never happen in any fixed multiplicative update scheme.

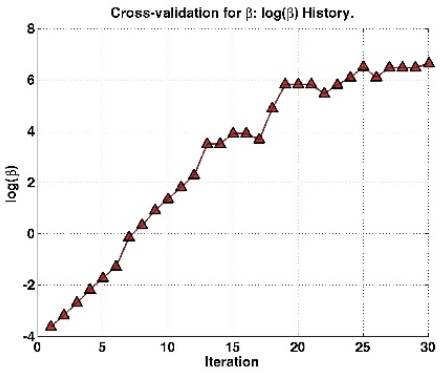
We now demonstrate the need for an automated regularization scheme, on another well-known test problem in $\mathbb{R}^4$, the Woods problem, given by

$$\begin{aligned} G_{\text{woods}}(x) &= 100(x_2 - x_1)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ &\quad +10.1[(1 - x_2)^2 + (1 - x_4)^2] + 19.8(1 - x_2)(1 - x_4). \end{aligned}$$

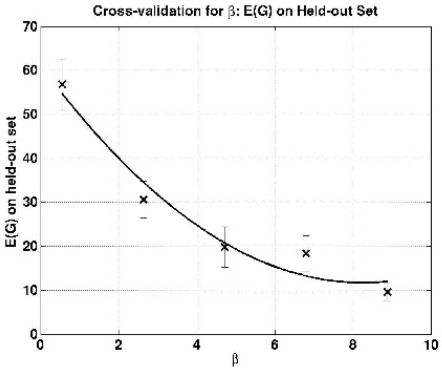
The optimum value of 0 is achieved at $x = (1, 1, 1, 1)$. We run the PC algorithm 50 times with cross-validation for regularization. For this experiment, we used a single Gaussian q, and set



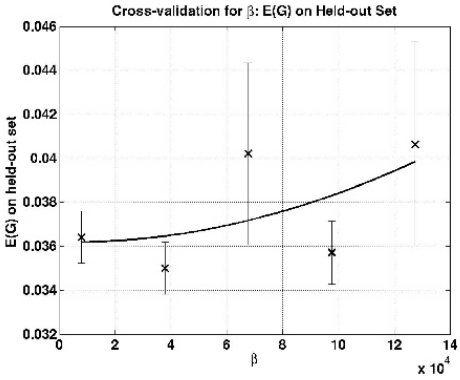
(a) $\mathbb{E}_q G(x)$ history.



(b) β history.



(c) Fit $Q(\beta)$, iteration 8.

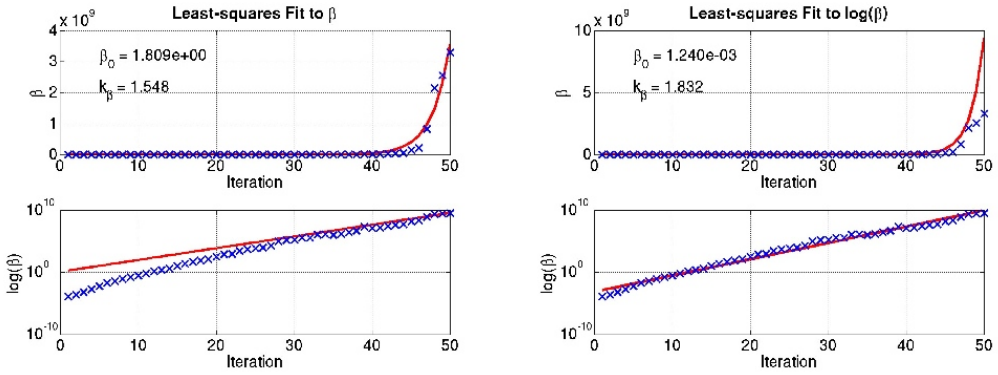


(d) Fit $Q(\beta)$, iteration 15.

Figure 13: Cross-validation for β: 2-D Rosenbrock $G(x)$.

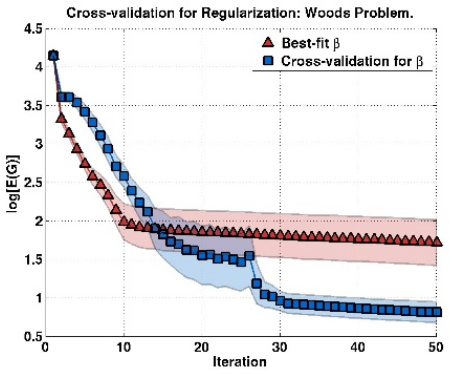
$$\texttt{maxExtIter} = 4, \quad k_1 = 0.5, \quad k_2 = 3,$$
$$N_j = 20, \quad n_\beta = 5, \quad K = 10.$$

From these results, we then attempt to find the best-fit multiplicative update rule for β, only to find that the average β variation is not at all well-approximated by *any* fixed update $\beta \leftarrow k_\beta \beta$. This poor fit is shown in Fig. 14, where we show a least-squares fit to both β and $\log(\beta)$. In the fit to $\log(\beta)$ the final β is off by over 100%, and in the fit to β, the initial β is off by several orders of magnitude. We then compare the performance of cross-validation to that of PC algorithm using the fixed update rule derived from the best least-squares fit to $\log(\beta)$. From a comparison over 50 runs, we see that using this best-fit update rule performs extremely poorly - cross-validation yields an improvement in final $\mathbb{E}_{q_\theta} G(x)$ by over an order of magnitude, as shown in Fig. 15.



(a) Least-squares fit to β: (b) Least-squares fit to $\log(\beta)$:

Figure 14: Best-fit β update rule.



(a) $\log(\mathbb{E}_q G)$ history.

Figure 15: Cross-validation beats best-fit fixed β update: 4-D Woods $G(x)$.

5.2.5 Bagging

While regularization is a method to decrease bias, bagging is a well-known variance-reducing technique. Bagging is easily incorporated in our algorithm. Suppose, at some stage in the algorithm, we have N samples (x^i, g^i), we resample our existing data set exactly N times with replacement. This gives us a different set of data set D', which also contains some duplicates. We compute optimal parameters $\theta^\star(D')$. We repeat this resampling process k_b times and uniformly average the resulting optimal densities $q_{\theta^\star(D'_k)}, k = 1, \ldots, k_b$.

We demonstrate this procedure, using the Rosenbrock function and a single Gaussian q_θ. In this experiment, we also demonstrate the ability of PC to handle non-deterministic oracles by adding uniform random noise to every function evaluation, that is, $(g \mid x, G) \sim \mathcal{U}[-0.25, 0.25]$. For this experiment, $N_j = 20, k_b = 5$.

The β update is performed using the same cross-validation algorithm described above. Fig. 16 shows the results of 50 runs of the PC algorithm with and without bagging. We see that bagging finds better solutions, and moreover, it reduces the

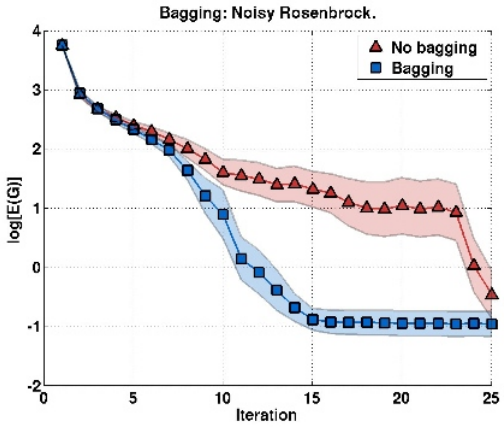


Figure 16: Bagging improves performance: Noisy 2-D Rosenbrock.

variance between runs. Note that the way we use bagging, we are only assured of improved variance for a single MC estimation at a given θ, and not over the whole MCO process of searching over θ.

5.2.6 Cross-validation for Regularization and Model Selection

In many problems like the Rosenbrock, a single Gaussian is a poor fit to p^β for many values of β. In these cases, we can use a mixture of Gaussians to obtain a better fit to p^β. This raises the question of how to choose the number of components in the mixture model. To address this, we use the fact that cross-validation can be used more broadly than just picking the regularization parameter of the training algorithm. It can be used to pick *any* set of "hyperparameters" of the training algorithm, including the hyperparameter of what model to use.

We ran a series of experiments testing this, using a greedy algorithm based on cross-validation to search over both a set of candidate β's and a set of candidate models:

1. We first pick the regularization parameter β, using Algorithm 2.

2. For that β, we use Algorithm 3 to pick the number of mixture components.

The details of our experiments are the same as the preceding section, but without bagging. The set of models $\{\{\phi\}\}$ is the set of Gaussian mixtures with one,

Algorithm 3 Cross-validation for model selection.

Initialize the set $\{\{\phi\}\}$ of model classes $\{\phi\}$ to search over.
Partition the data into K disjoint subsets.
for each fold k, **do**
  Training data is all but the k^{th} data partitions.
  Test data is the k^{th} data partition.
  for $\{\phi_i\}$ in $\{\{\phi\}\}$ **do**
    Compute the optimal parameter set $\theta^\star(D_{T_k}) \in \{\phi_i\}$
    Compute held-out performance $\widehat{g}(\theta^\star(D_{V_k}))$
  end for
  Compute the sample held-out performance, $\overline{g}(\{\phi_i\})$, from Eq. 24.
end for
Choose best model class $\{\phi^\star\} = \arg\min_{\phi_i} \overline{g}(\{\phi_i\})$.

two or three mixing components. Fig. 17 shows the variation of $\mathbb{E}_q(G)$ vs. iteration. The mixture model is much quicker to yield lower expected G, because the Boltzmann at many values of β is better approximated by a mixture of Gaussians. However, note that the mixture models performs poorly towards the end of the run. The reason for this is as follows: No shape regularization was used during the EM procedure. This means that the algorithm often samples from nearly degenerate Gaussians. These 'strange' sample sets hurt the subsequent performance of importance sampling, and hence of the associated MCO problem. This can be alleviated by using some form of shape regularization in the EM algorithm.

5.3 Other applications of PL techniques to immediate sampling

There are numerous other PL techniques that have been successfully applied to immediate sampling. For example, cross-validation is a "winner-takes-all" technique; it chooses the single value of a "hyperparameter" like β that optimizes in-sample vs. out-of-sample behavior and then uses that value to produce a single fit to some data set. As an alternative, one can use that same in-sample vs. out-of-sample behavior to determine how best to *combine* the (fits generated using the) hyperparameters under consideration. This technique is known as **stacking** [53, 54, 55], and has been extremely successful in supervised machine learning. In [26, 27] stacking was applied for immediate sampling, and resulted in a substantial gain in performance.

Another set of techniques is related to the use of surrogate models in conventional optimization. The general idea behind these techniques is to form a single-valued fit to the data set D, $L(x)$ (e.g., using regression is x is a real-valued

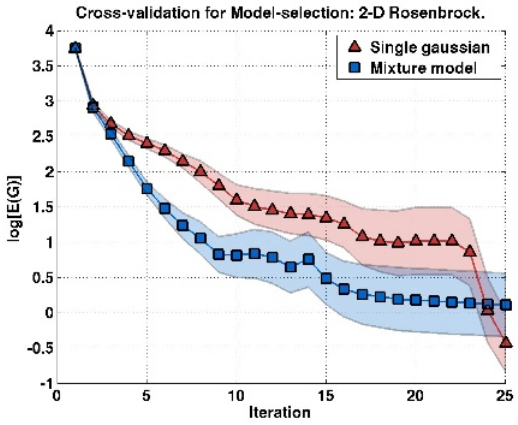


Figure 17: Cross-validation for regularization and model-selection: 2-D penalty function $G(x)$.

vector), or more generally use D to form a fit $L(g \mid x)$ to the actual conditional distribution $P(g \mid x, \mathscr{G})$ (e.g., by using Gaussian processes). One could then use L to guide the setting of q_θ rather than D. The PL field of active learning in general is closely related to this, and in particular the recent work on sampling of bandits, e.g., with the UCB algorithm [56], is closely related, as are the technique of using surrogate models in optimization.

Note though that L will not be a perfectly accurate model of the oracle in general. This implies that we should not use just L to form our next q_θ, in analogy to the work on sampling of bandits, but instead form our next q_θ after combining D and L in a way that gives more credence to D. Techniques for doing this are called **fit-based MCO / immediate sampling**. As an example, note that while samples of the actual oracle $\mathscr{G}$ are expensive, samples of L are quite cheap. We can exploit this is to combine D and L: pick some integer k and form the union of k copies of the original training set D with a set of $k|D|$ (cheap) samples of L, and use that union rather than D as the training set for forming q_θ.[10]

Finally, there is ongoing work exploring what the best choice for F is. For example, arguably one is not interested in $\mathbb{E}_{q_\theta}(G) = \int dx\ q_\theta(x)G(x)$, but rather in **expected improvement**, which is defined as $\int_a dx\ q_\theta(x)G(x)$, where a is the best value g found in the current data set D. This suggests several changes to the use of qp and/or pq KL distances. Extensions of this idea are concerned not with the probability of the possible g values produced by sampling $q_\theta(x)$ once, but rather with the probability of the best g value produced by sampling $q_\theta(x)$ a total of m

[10]Note that if we have knowledge concerning the gradient of $G(x)$ at some x's in addition to having D's samples of $G(x)$, then we can use that extra knowledge to help set L. This is a way to incorporate gradient-based information concerning the oracle into immediate sampling.

times. (The difference is between considering mean behavior under q_θ and tail behavior under q_θ.)

Even more ambitiously, one might try to account for the fact that the next samples of q_θ will be used to update q_θ itself. After all, it is that updated version of q_θ that will subsequently be sampled, and therefore it is the samples of that updated version of q_θ that we are truly concerned with. For example, one could imagine doing this by "double-cross-validation". (This more ambitious type of immediate sampling is also related to both the active learning and bandits literatures.) In this approach D is divided into *three* data sets, D_1, D_2 and D_3. One then trains on D_1 to form a q_{θ^1} using a particular value of some hyperparameter like β. Next one uses importance sampling corrections to resample D_2 to get D_2^1, a set of unbiased samples of q_{θ^1}. After this one trains on $D_1 \cup D_2^1$ to form q_{θ^2}. Finally, one uses the empirical average of $F(g, q_{\theta^2}$ across D_3 to judge the quality of the initial hyperparameter.

6 Conclusion

This article concerns "blackbox optimization" algorithms in which one iterates the following procedure: Choose a value $x \in X$, getting statistical information about an associated value $G(x)$, then use the set of all pairs $\{(x, G(x))\}$ found so far to choose a next x value at which to sample G, the goal being to find x's with as small $G(x)$ as possible, and to do so as fast as possible. Examples of conventional blackbox optimization algorithms are genetic algorithms, simulated annealing, etc. These conventional algorithms work directly with values x, stochastically mapping the set $\{(x, G(x))\}$ to the next x. The distribution over new x's that gets sampled is never explicitly optimized. In contrast, in the Probability Collectives (PC) approach, one explicitly uses the set $\{(x, G(x))\}$ to optimize the probability distribution over x that will be sampled.

There are two general types of PC, "delayed sampling", and "immediate sampling". In delayed sampling one restricts the distribution to be a product distribution. This allows a lot of the calculation of the optimal next distribution to be done in closed form. Delayed sampling PC is particularly well-suited to distributed optimization / control problems, where a product distribution is a natural way to describe the problem.

In immediate sampling one instead reformulates the blackbox optimization problem in a way that is very similar to Monte Carlo optimization. One advantage of reformulating PC this way is that it allows us to use an arbitrary class of probability distributions, rather than being restricted to product distributions as in delayed sampling PC. Another advantage is that the reformulation can be done in a way that makes PC's core issue, of how best to estimate a distribution based

on samples, formally identical to the general problem of inductive inference considered in parametric machine learning. This formal identity allows us to apply the many very powerful techniques that have been developed in machine learning to the problem of blackbox optimization, thereby substantially improving performance.

This article reviewed some of the theory behind both kinds of PC. It then presented some of the many experiments that have demonstrated the power of both kinds of PC.

References

[1] D. H. Wolpert, C. E. M. Strauss, and D. Rajnayaran. Advances in distributed optimization using probability collectives. *Advances in Complex Systems*, 2006. in press.

[2] D. H. Wolpert. What Information theory says about best response, binding contracts, and Collective Intelligence. In A. Namatame, editor, *Proceedings of WEHIA04*. Springer Verlag, 2004.

[3] D. H. Wolpert and S. Bieniawski. Adaptive distributed control: beyond single-instant categorical variables. In A. Skowron, editor, *Proceedings of MSRAS04*, pages 1562–1567. Springer Verlag, 2004.

[4] D. H. Wolpert. Information theory - the bridge connecting bounded rational game theory and statistical physics. In D. Braha, A. Minai, and Y. Bar-Yam, editors, *Complex Engineered Systems: Science meets technology*, pages 262–290. Springer, 2004.

[5] S. Bieniawski, I. Kroo, and D. H. Wolpert. Flight Control with Distributed Effectors. In *Proceedings of 2005 AIAA Guidance, Navigation, and Control Conference, San Francisco, CA*, 2005. AIAA Paper 2005-6074.

[6] W. Macready, S. Bieniawski, and D.H. Wolpert. Adaptive multi-agent systems for constrained optimization. Technical report IC-04-123, 2004.

[7] C. Fan Lee and D. H. Wolpert. Product distribution theory and semi-coordinate transformations. *Proceedings of the Third Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 522–529, 2004. Columbia University.

[8] D. H. Wolpert and S. Bieniawski. Distributed control by lagrangian steepest descent. In *Proceedings of the 2004 IEEE Control and Decision Conference*, pages 1562–1567, 2004.

[9] S. Bieniawski and D. H. Wolpert. Adaptive, distributed control of constrained multi-agent systems. In *Proceedings of the Third Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 1230–1231, 2004.

[10] S. Bieniawski, D. H. Wolpert, and I. Kroo. Discrete, continuous, and constrained optimization using collectives. In *Proceedings of 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York*, 2004.

[11] N. Antoine, S. Bieniawski, I. Kroo, and D. H. Wolpert. Fleet assignment using Collective Intelligence. In *Proceedings of 42nd Aerospace Sciences Meeting*, 2004. AIAA-2004-0622.

[12] D. H. Wolpert and C. F. Lee. Adaptive Metropolis-Hastings sampling using product distributions. 2004. cond-mat/0504163.

[13] D. H. Wolpert, K. Tumer, and J. Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems - 11*, pages 952–958. MIT Press, 1999.

[14] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

[15] D. H. Wolpert and K. Tumer. Collective Intelligence, Data Routing and Braess' Paradox. *Journal of Artificial Intelligence Research*, 16:359–387, 2002.

[16] D. H. Wolpert. Theory of Collective Intelligence. In K. Tumer and D. H. Wolpert, editors, *Collectives and the Design of Complex Systems*, New York, 2003. Springer.

[17] W. Macready and D. H. Wolpert. Distributed constrained optimization with semicoordinate transformations. submitted to Journal of Operations Research, 2005.

[18] J. R. Meginniss. A new class of symmetric utility rules for gambles, subjective marginal probability functions, and a generalized Bayes' rule. *Proceedings of the American Statistical Association, Business and Economics Statistics Section*, pages 471 – 476, 1976.

[19] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. MIT Press, Cambridge, MA, 1998.

[20] J.S. Shamma and G. Arslan. Dynamic fictitious play, dynamic gradient play, and distributed convergence to nash equilibria. *IEEE Trans. on Automatic Control*, 50(3):312–327, 2004.

[21] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620, 1957.

[22] J.S. De Bonet, C.L. Isbell Jr., and P. Viola. Mimic: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems - 9*. MIT Press, 1997.

[23] R. Rubinstein and D. Kroese. *The Cross-Entropy Method*. Springer, 2004.

[24] P. Sabes and M. Jordan. Reinforcement learning by probability matching. In *Advances in Neural Information Processing Systems - 8*. MIT Press, 1995.

[25] D. H. Wolpert, K. Tumer, and E. Bandari. Improving search by using intelligent coordinates. *Physical Review E*, 69(017701), 2004.

[26] D. Rajnarayan and David H. Wolpert. Bias-variancee trade-offs: Novel applications. In C. Sammut and G. Webb, editors, *Encyclopedia of Machine Learning*, 2011.

[27] D. Rajnarayan and David H. Wolpert. Exploiting parametric learning to improve black-box optimization. In J. Jost, editor, *Proceedings of ECCS 2007*, 2007.

[28] D. Rajnarayan and David H. Wolpert. Bias-variance techniques for monte carlo optimization: Cross-validation for the ce method. arXiv:0810.0877v1, 2008.

[29] D.H. Wolpert and D. Rajnarayan. Parametric learning and monte carlo optimization. http://lanl.arxiv.org/abs/0704.1274, 2007.

[30] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

[31] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, 1991.

[32] D.J.C. Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[33] D. Koller and N. Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

[34] D. H. Wolpert. Product distribution field theory. 2003. preprint cond-mat/0307630.

[35] D. Fudenberg and D. Kreps. Learning mixed equilibria. *Games and Economic Behavior*, 5:320–367, 1993.

[36] D. Fudenberg and D. K. Levine. Steady state learning and Nash equilibrium. *Econometrica*, 61(3):547–573, 1993.

[37] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd ed.)*. Wiley and Sons, 2000.

[38] D. H. Wolpert, K. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. *Europhysics Letters*, 49(6):708–714, March 2000.

[39] W. Macready and D. H. Wolpert. Distributed optimization. In *Proceedings of ICCS 04*, 2004.

[40] S. Bieniawski. *Distributed Optimization and Flight Control Using Collectives*. PhD thesis, Stanford University, 2005.

[41] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical optimization*. Academic Press, 1981.

[42] S. Bieniawski and D. H. Wolpert. Product distributions for distributed optimization. In *International Conference on Complex Systems*, Boston, Mass., May 2004.

[43] E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. 1994. Working paper CRIF Industrial Management and Automation, CP 106 - P4, 50 av. F.D.Roosevelt, B-1050 Brussels, Belgium, email: efalkena@ulb.ac.be.

[44] J.E.Beasley. Or-library:distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41 (11):1069–1072, 1990. http://mscmga.ms.ic.ac.uk/info.html.

[45] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4), July-August 1987.

[46] M. Holden. *Optimization of Dynamic Systems Using Collocation Methods*. PhD thesis, Stanford University, 1999.

[47] Eric W. Weisstein. Brachistochrone problem. From *MathWorld* – A Wolfram Web Resource. http://mathworld.wolfram.com/BrachistochroneProblem.html.

[48] H. Ashley. *Engineering Analysis of Flight Vehicles*. Dover Publications, New York, 1974.

[49] The MathWorks. *MATLAB Optimization Toolbox User's Guide*, 2000.

[50] Y. M. Ermoliev and V. I. Norkin. Monte carlo optimization and path dependent nonstationary laws of large numbers. Technical Report IR-98-009, International Institute for Applied Systems Analysis, March 1998.

[51] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, New York, 2004.

[52] D. H. Wolpert. On bias plus variance. *Neural Computation*, 9:1211–1244, 1997.

[53] L. Breiman. Stacked regression. *Machine Learning*, 24, 1996.

[54] B. Clarke. Bayes model averaging and stacking when model approximation error cannot be ignored. *Journal of Machine Learning Research*, pages 683–712, 2003.

[55] J. Sill, G. Takacs, Mackey L., and Lin D. Feature-weighted linear stacking. unpublished: arXiv:0911.0460, 2009.

[56] P. Auer, Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.