Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

On the approximate domain optimization of deterministic and expected value criteria

A. Lecchini-Visintini, J. Lygeros and J. Maciejowski

Abstract— **We define the concept of approximate domain optimizer for deterministic and expected value optimization criteria. Roughly speaking, a candidate optimizer is an approximate domain optimizer if only a small fraction of the optimization domain is more than a little better than it. We show how this concept relates to commonly used approximate optimizer notions for the case of Lipschitz criteria. We then show how random extractions from an appropriate probability distribution can generate approximate domain optimizers with high confidence. Finally, we discuss how such random extractions can be performed using Markov Chain Monte Carlo methods.**

I. Introduction

For continuous domains, most of the popular optimization methods perform a local gradient-based search and in general converge to local optimizers; in some cases, for example convex optimization problems, it may also be possible to guarantee convergence to a global optimizer [1]. By contrast, random search methods such as simulated annealing perform a global search and can therefore be considered as a powerful complement to local search methods. Moreover, under certain assumptions it can be shown that simulated annealing algorithms converge to a global optimizer as the number of steps grows to infinity even for continuous optimization domains [2], [3], [4], [5]. Little is known about the rate of convergence and the finite-time properties of these algorithms, however, unless the domain is finite [6], [7], [8], [9].

To overcome this shortcoming, in this paper we introduce an approximate optimizer concept, motivated by the concept of finite-time learning with known accuracy and confidence used in statistical learning theory [10], [11]. In general, optimization algorithms for problems defined on continuous variables can only find approximate solutions in finite time [12]. Here we consider three levels of approximation: Domain approximation (what fraction of the optimization domain is better than our candidate optimizer), value approximation (how much better are these points), and confidence (if the candidate optimizer is random, how confident are we about its domain and value properties). We show that under very weak assumptions on the optimization domain and optimization criterion this three way approximation can be used to obtain finite-time performance guarantees for simulated annealing algorithms over continuous domains. Moreover, under some additional (but still quite weak) regularity conditions we demonstrate how our approximate optimizer concept can be related to more standard approximate optimizers considered in the stochastic programming literature. This link provides theoretical support for the use of simulated annealing and Markov chain Monte Carlo methods for that have been proposed in, for example, [13], [14], [15] for solving stochastic programming problems.

In Section II we introduce the approximate domain optimizer concept and show how it relates to approximate optimizer concepts in the stochastic programming literature (Theorem 1). In Section III we define an appropriate sense in which random variables can be thought of as approximate domain optimizers and establish a family of probability distributions that can be used to generate approximate domain optimizers with high confidence (Theorem 2). Then in Section IV we present a method for performing extractions from this family of probability distributions using Markov Chain Monte Carlo (MCMC) methods. Proofs are omitted in the interest of space. Detailed proofs of all facts can be found in [16]; a proof of Theorem 2 was reported in [17].

II. Approximate domain optimizers

Let λ denote the Lebesgue measure on $\mathbb{R}^n$. The 2-norm will be used throughout for $\mathbb{R}^n$ and the total variation norm will be used for measures. Probability densities are given with respect to the Lebesgue measure. Various Greek letters will be used to denote approximation parameters. To avoid repeatedly stating the range in which they take values, from now on we assume that $\alpha \in (0,1)$, $\gamma \in (0,1)$, $\delta > 0$, $\epsilon \in (0,1)$, $\rho \in (0,1)$, and $\sigma \in (1/2, 1)$. We also assume that $J \geq 0$.

Consider the optimization criterion $U : \Theta \to \mathbb{R}$ with $\Theta \subseteq \mathbb{R}^n$. Let

$$U^* = \sup_{\theta \in \Theta} U(\theta). \tag{1}$$

U can be either a deterministic or an expected value optimization criterion. In the former case for every $\theta \in \Theta$ the value of $U(\theta)$ can be computed directly. In this case the optimization problem (1) is a standard programming problem. In the latter case, $U(\theta)$ is given as an expected value over a random variable whose probability distribution may also depend on θ. More formally, we assume the existence of a random variable x taking values in a set X, according to

The authors would like to thank M. Vidyasagar for helpful discussions. Research supported by EPSRC (UK) under grant EP/C014006/1 and by the European Commission under the projects HYGEIA, FP6-NEST-004995 and iFly, FP6-TREN-037180.

A. Lecchini-Visintini is with the Department of Engineering, University of Leicester, alvl@leicester.ac.uk, J. Lygeros is with the Automatic Control Laboratory, ETH Zurich, lygeros@control.ee.ethz.ch, and J. Maciejowski is with the Department of Engineering, University of Cambridge jmm@eng.cam.ac.uk.

a family of probability measures $P(dx;\theta)$ parameterized by $\theta \in \Theta$. We further assume that $U(\theta)$ for each value of $\theta \in \Theta$ can be written as an expected value over the corresponding probability measure

$$U(\theta) = \int_{x \in X} u(x,\theta) P(dx;\theta)$$

for some function $u : X \times \Theta \to \mathbb{R}$. In this case the optimization problem (1) is a stochastic programming problem. We assume throughout that $u(x,\theta)$ and $P(dx;\theta)$ are such that the expected value $U(\theta)$ is well defined for all $\theta \in \Theta$. In such problems it is usually not possible to compute $U(\theta)$ directly, either because evaluation of the integral requires too much computation, or because an analytical expression for $P(dx;\theta)$ is not available; for example, [18], [19] present applications in air traffic management and systems biology where $P(dx;\theta)$ can only be sampled by simulation. In the particular case that $P(dx;\theta)$ does not depend on θ, the optimization task is often called "empirical risk minimization", and is studied extensively in statistical learning theory [10], [11]. The results of this paper apply equally to the optimization of deterministic and expected-value criteria.

We will consider the optimization problem (1) under two separate sets of assumptions.

Assumption 1: Θ is Lebesgue measurable and bounded (i.e. $\lambda(\Theta) < \infty$). U is Lebesgue measurable and bounded.
This very weak assumption will be a standing assumption for all results. The approximate domain optimizer results will hold vacuously if $\lambda(\Theta) = 0$ so we will mostly be interested in the case $\lambda(\Theta) > 0$. Without loss of generality we will assume that $U(\theta) \in [0,1]$ for all $\theta \in \Theta$.

For some results a somewhat stronger assumption will be needed.

Assumption 2: Θ is compact and Lebesgue measurable. U is $L-$Lipschitz continuous.
Conditions on $u(x,\theta)$ and $P(dx;\theta)$ to ensure that $U(\theta)$ is Lipschitz can be found in [20]. It is easy to see that Assumption 2 implies Assumption 1 and also the existence of a global optimizer.

*Definition 1 (*Approximate domain optimality): $\hat{\theta} \in \Theta$ is called an approximate domain optimizer (with respect to the Lebesgue measure) with value imprecision ϵ and residual domain α if and only if

$$\lambda\{\theta \in \Theta \mid U(\theta) > U(\hat{\theta}) + \epsilon\} \leq \alpha\lambda(\Theta).$$

In words, this optimality concept provides a bound on the Lebesgue measure of the set of points that are ϵ better than our candidate as a fraction of the Lebesgue measure of the entire domain Θ. This notion of optimality is motivated by the work of Vidyasagar on statistical learning theory [10], [11] and especially its application to automatic control problems [21], [22]. The main difference is that in the work of Vidyasagar measures other than the Lebesgue measure are allowed (typically probability measures used to sample the space Θ). We will only consider the Lebesgue measure here, hence will drop the phrase "with respect to the Lebesgue measure" whenever we refer to this optimality concept. We will use

$$\Theta(\epsilon,\alpha) = \{\hat{\theta} \in \Theta \mid \lambda\{\theta \in \Theta \mid U(\theta) > U(\hat{\theta}) + \epsilon\} \leq \alpha\lambda(\Theta)\}$$

to denote the set of approximate domain optimizers with value imprecision ϵ and residual domain α. Note that for all ϵ and all α, if $\Theta^* \neq \emptyset$ then $\Theta^* \subseteq \Theta(\epsilon,\alpha)$; moreover, $\Theta(\epsilon,\alpha) \neq \emptyset$ even if $\Theta^* = \emptyset$.

A more common notion of approximate optimizer is the following.

*Definition 2 (*Approximate value optimality): A point $\hat{\theta} \in \Theta$ is called an approximate value optimizer with imprecision ϵ if and only if $U(\theta) \leq U(\hat{\theta}) + \epsilon$ for all $\theta \in \Theta$.
This notion is commonly used in the stochastic programming literature (see, for example, [20], [23]) and provides a direct bound on the optimal value. We will use

$$\Theta^*(\epsilon) = \{\hat{\theta} \in \Theta \mid \forall\theta \in \Theta,\ U(\theta) \leq U(\hat{\theta}) + \epsilon\}$$

to denote the set of approximate value optimizers with imprecision ϵ. Once again, for all $\epsilon > 0$, if $\Theta^* \neq \emptyset$ then $\Theta^* \subseteq \Theta^*(\epsilon)$, but $\Theta^*(\epsilon) \neq \emptyset$ even if $\Theta^* = \emptyset$.

One can see that approximate value optimality is a stronger concept than approximate domain optimality, in the sense that for all ϵ and α, $\Theta^*(\epsilon) \subseteq \Theta(\epsilon,\alpha)$. Conversely, however, given an approximate domain optimizer it is in general not possible to draw any conclusions about the approximate value optimizers. A relation between domain and value approximate optimality can, however, be established under Assumption 2. Let Γ denote the gamma function.

Theorem 1: Under Assumption 2, if $\hat{\theta}$ is an approximate domain optimizer with value imprecision ϵ and residual domain α then it is also an approximate value optimizer with imprecision $\epsilon + \frac{L}{\sqrt{\pi}}\left[\frac{n}{2}\Gamma\left(\frac{n}{2}\right)\right]^{\frac{1}{n}}[\alpha\lambda(\Theta)]^{\frac{1}{n}}$.

III. Random domain optimizers

We now examine how random $\hat{\theta} \in \Theta$ extracted according to some probability distribution $\hat{\theta} \sim \pi(d\theta)$ can be thought of as domain optimizers. The natural interpretation is to try to ensure that the probability with which the extracted $\hat{\theta}$ are approximate domain optimizers with value imprecision ϵ and residual domain α is high enough; in other words, establish a lower bound on the measure of the set $\Theta(\epsilon,\alpha)$ with respect to the probability measure $\pi(d\theta)$.

A. Proposed distribution family

We consider a family of probability distributions $\pi(d\theta; J,\delta)$ parameterized $J \geq 0$ and $\delta > 0$.

Theorem 2: Consider the distribution $\pi(d\theta; J,\delta) \propto [U(\theta) + \delta]^J \lambda(d\theta)$ with $J \in \mathbb{N}$ and $\delta > 0$. For all ϵ, α we have that

$$\pi(\Theta(\epsilon,\alpha); J,\delta) \geq \frac{1}{1 + \left[\frac{1+\delta}{\epsilon+1+\delta}\right]^J \left[\frac{1}{\alpha}\frac{1+\delta}{\epsilon+\delta} - 1\right]\frac{1+\delta}{\delta}}$$

For a proof of the theorem and a discussion of its implications see [17]. Notice that, by construction, $\pi(d\theta; J,\delta)$ is absolutely continuous with respect to the Lebesgue measure with density $p(\theta; J,\delta) \propto [U(\theta) + \delta]^J$. Intuitively, parameter J determines how concentrated the distribution $\pi(d\theta; J,\delta)$

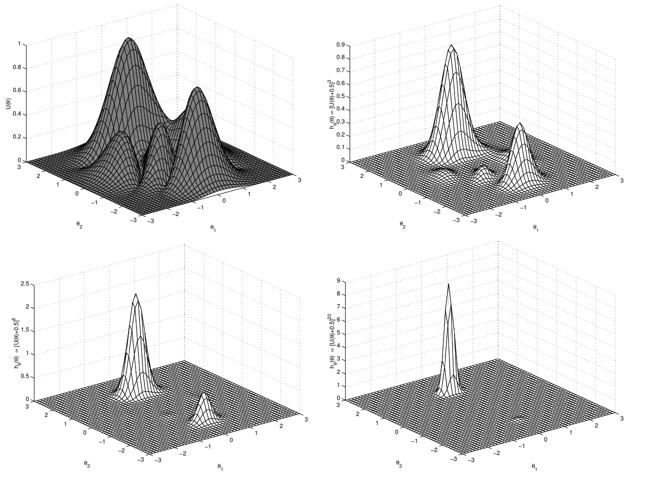

Fig. 1. Example of the effect of exponent J. The top left plot shows $U(\theta)$. The remaining plots show $[U(\theta)+\delta]^J$ for $\delta = 0.5$ and $J = 3$ (top right) 6 (bottom left) and 20 (bottom right) respectively.

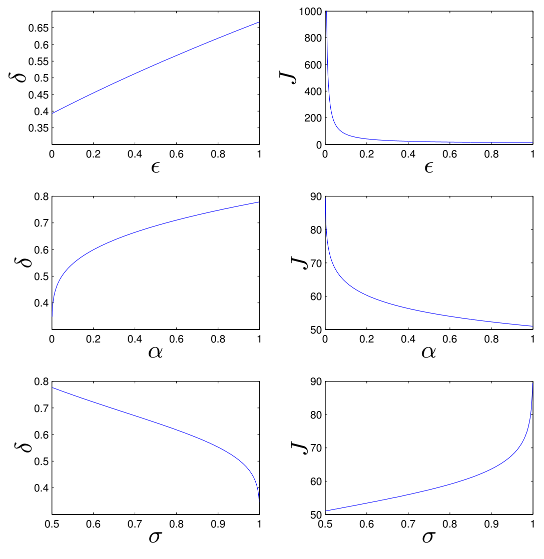

Fig. 2. Variation of the "optimal" δ and the corresponding J with respect to ϵ, α, and σ. Two of the three parameters ϵ, α, σ are kept constant for each row, set to $\epsilon = 0.1$, $\alpha = 0.01$, $\sigma = 0.99$. Notice the rapid growth of J as $\epsilon \longrightarrow 0$ and the mild growth as $\alpha \longrightarrow 0$ and $\sigma \longrightarrow 1$.

will be around the global optima of $U(\theta)$ (see Figure 1). The parameter δ can be thought of as the amount of Lebesgue measure (uniform distribution) one adds to $U(\theta)$ before sampling. The presence of at least some uniform distribution is required for technical reasons in the proof of Theorem 2. Exactly how much is up to the designer, however. To obtain some insight on this choice it is instructive to turn the bound of Theorem 2 around, to provide a lower bound on J to ensure that $\hat{\theta} \in \Theta(\epsilon, \alpha)$ with high enough probability.

Corollary 1: Consider $\sigma \in (1/2, 1)$. $\pi(\Theta(\epsilon,\alpha); J, \delta) \geq \sigma$ as long as

$$J \geq \frac{1+\epsilon+\delta}{\epsilon}\left[\log\frac{\sigma}{1-\sigma} + \log\frac{1}{\alpha} + 2\log\frac{1+\delta}{\delta}\right].$$

It turns out that the bound given for J in Corollary 1 admits a unique minimum for δ as a function of α, σ and ϵ [16]. Figure 2 shows this value of δ and the value of the corresponding bound.

If the optimization criterion is Lipschitz continuous, Corollary 1 can be used together with Theorem 1 to derive bounds on the probability that a $\hat{\theta}$ extracted according to $\pi(d\theta; J, \delta)$ is an approximate value optimizer with a given imprecision.

B. Related methods

M. Vidyasagar [11], [21] proposed a fully randomized algorithm for obtaining approximate domain optimizers with respect to a general probability measure $\pi(d\theta)$ (not just the Lebesgue measure considered here). The idea is to extract N independent samples $\theta_1, \ldots, \theta_N$ according to $\pi(d\theta)$ and K independent samples $x_1, \ldots, x_K$ according to $P(dx)$ and set

$$\hat{\theta}_V = \arg\min_{i=1,\ldots,N} \frac{1}{K}\sum_{j=1}^{K} u(x_j, \theta_i). \tag{2}$$

Notice that $\hat{\theta}_V$ is again a random variable, due to the random extractions for θ and x. Under minimal assumptions (close to our Assumption 1) it can be shown that if

$$N \geq \frac{\log\frac{2}{1-\sigma}}{\log\frac{1}{1-\alpha}} \text{ and } K \geq \frac{1}{2\epsilon^2}\ln\frac{4N}{1-\sigma}, \tag{3}$$

then

$$\pi\{\theta \in \Theta \mid U(\theta) > U(\hat{\theta}) + \epsilon\} \leq \alpha \tag{4}$$

with probability at least σ. Here "probability" is taken with respect to the product measure $\pi^N \times P^K$ for the independent extractions $\theta_1, \ldots, \theta_N, x_1, \ldots, x_K$. The total computation necessary to obtain $\hat{\theta}_V$ is proportional to $N + K$ plus the amount of time needed to solve the optimization problem (2), which is linear in NK. Similar (potentially tighter) bounds can be obtained if the family of functions $\{u(\cdot, \theta) \mid \theta \in \Theta\}$ has the Uniform Convergence of Empirical Means property.

This approach is clearly related to the approach proposed here, but it is much simpler to implement and much easier to analyze from a computational complexity point of view. Notice, however, that the measure of the set of points which are ϵ better than the candidate optimizer is taken with respect to π in (4) as opposed to the Lebesgue measure in Definition 1. Since the search distribution π is arbitrary (design choice), this makes it difficult to prove results like Theorem 1, linking approximate domain optimizers with approximate value optimizers, even in the case of Lipschitz criteria. More importantly, because $\pi(d\theta)$ is used both to sample from Θ and to assess the measure of the set of points that are better than the obtained solution, one runs the risk of (4) becoming a self fulfilling prophecy. A myopic search strategy that samples only a small part of the space may end up looking good from the point of view of equation (4), but would leave large parts of Θ with potentially high values of $U(\theta)$ unexplored.

In terms of the necessary computation, the bound in Corollary 1 scales more slowly than (3) with respect to ϵ ($1/\epsilon$ as opposed to $1/\epsilon^2$). The growth with respect to σ and α is comparable. Note, however, that the bounds of (3) effectively include all the necessary computations, since the cost of solving (2) is small. By contrast, the bounds on J only tell part of the story, since the cost of extracting a sample from the distribution $\pi(d\theta; J, \delta)$ is not included. We address this issue in the next section.

In the stochastic programming literature, several other methods have been proposed for obtaining approximate value optimizers. Most of these methods [24], [20], [25], [23] rely on random extractions from $P(dx;\theta)$ which are used to estimate either U or its gradients, followed by a structured search over the space Θ. Drawing a direct comparison between these methods and methods based on randomly sampling the parameter space is not straightforward, however, since some methods require solving an additional optimization problem [24], [20], others require computing sub-gradients [25], [23], others require extracting from complex distributions, etc. The above discussion can lead to some insights about the relative merits of the different approaches [16]. See also [26] for a different class of randomized optimization methods, geared primarily toward deterministic, convex problems.

In terms of applicability, the approach based on extractions from $\pi(d\theta; J, \delta)$ and the approach of Vidyasagar apply to very general optimization criteria, provided one is willing to live with approximate domain optimizers. The approach of [24], [20] applies only to Lipschitz criteria. It does, however, also require one to solve an additional optimization problem constructed through Monte Carlo extractions, which is likely to be generally feasible only under convexity assumptions. The same is roughly true for the approach of [25], [23]. These approaches (as well as the approach of [11], [21]) also appear to be limited to the case where $P(dx;\theta)$ does not depend on θ, though in some cases the extension to more general $P(dx;\theta)$ is straightforward, at least conceptually. Our approach can deal with the general case $P(dx;\theta)$ by construction, as we will see in the next section; being able to deal with this case is crucial in certain applications, including those presented in [18], [19]. Another subtle difference is that the approach proposed here and the approach of Vidyasagar require one to sample both $\theta \in \Theta$ and $x \in X$, whereas the stochastic programming approaches of [24], [20], [25], [23] only require sampling $x \in X$. This may cause additional complication for the former methods, depending on the shape of the set Θ and the distribution used to sample from it. However, similar problems may be encountered when solving the follow-on optimization problem with the approach of [24], [20] or when computing sub-gradients with the approach of [25], [23].

Algorithm 1 (MCMC for deterministic criteria)

```
    initialization
        Select $\theta_0 \in \Theta$
        Set $k = 0$
    repeat
        Extract $\tilde{\theta} \sim g(\theta|\theta_k)$
        Set $\rho = \min\left\{1, \frac{g(\theta_k|\tilde{\theta})[U(\tilde{\theta})+\delta]^J}{g(\tilde{\theta}|\theta_k)[U(\theta_k)+\delta]^J}\right\}$
        Set $\theta_{k+1} = \begin{cases} \tilde{\theta} & \text{w.p. } \rho \\ \theta_k & \text{w.p. } 1-\rho \end{cases}$
        Set $k = k + 1$
    until True
```

TABLE I

IV. Generation of random domain optimizers using MCMC

Now let us turn to the question of how to generate random extractions according to the distribution required by Theorem 2. This can be done using Markov Chain Monte Carlo (MCMC) methods, by coding the desired distribution as the stationary distribution of a Markov chain. Algorithms I and II allow one to generate samples from the resulting chain for deterministic and expected value criteria respectively; notice that (unlike Theorem 2) Algorithm II only applies to the case where $J \geq 1$ and integer. Much more efficient algorithms for sampling the desired distributions are of course available, Algorithms I and II are listed here only to illustrate how this can be done and because they lead to simple statements of the subsequent results.

Algorithm I is the Metropolis-Hastings algorithm for generating a Markov chain with stationary distribution whose density function is proportional to $[U(\theta)+\delta]^J$. The same property for Algorithm II was shown for expected value criteria in [13], [14], [15]. Both algorithms maintain as (part of the) state a value $\theta_k \in \Theta$; in addition, Algorithm II also keeps track of an estimate, U_k, of the value of U at θ_k. At each loop they make use of a proposal distribution with density $g(\theta|\theta_k)$ to extract a new candidate $\tilde{\theta} \in \Theta$. The candidate is then accepted or rejected with a probability that relates to the relative value of $U(\tilde{\theta})$ and $U(\theta_k)$ and the relative likelihood of $\tilde{\theta}$ and θ_k with respect to $g(\cdot|\cdot)$. Notice that for implementation purposes, densities need to be known only up to a constant.

Let $\pi_k(d\theta)$ denote the probability distribution of the k^{th} step of the Markov chain and recall that the stationary distribution of the chain has density $p(\theta; J, \delta) \propto [U(\theta)+\delta]^J$. The following fact [17] is immediate from the definition of the total variation norm.

Theorem 3: Let θ_k with distribution $\pi_k(d\theta)$ be the state of the chain generated by Algorithm I or II and assume J respects the bound of Corollary 1. Then the statement "θ_k is an approximate domain optimizer of U with value imprecision ϵ and residual domain α" holds with probability at least $\sigma - \|\pi_k(d\theta) - p(\theta; J, \delta)\|$.

Standard results in the theory of Markov chains [27], [28], [29], [30], [31] allow one to establish conditions under which a Markov chain converges to its stationary distribution. In

```
Algorithm 2 (MCMC for expected value criteria)
    initialization
        Select $\theta_0 \in \Theta$
        Extract independent $x_i \sim P(dx, \theta_0)$, $i = 1, 2, \ldots, J$
        Set $U_0 = \prod_{i=1}^{J} [u(x_i, \theta_0) + \delta]$
        Set $k = 0$
    repeat
        Extract $\tilde{\theta} \sim g(\theta|\theta_k)$
        Extract independent $x_i \sim P(dx, \tilde{\theta})$, $i = 1, 2, \ldots, J$
        Set $\tilde{U} = \prod_{i=1}^{J} [u(x_i, \tilde{\theta}) + \delta]$
        Set $\rho = \min\left\{1, \frac{g(\theta_k|\tilde{\theta})\tilde{U}}{g(\tilde{\theta}|\theta_k)U_k}\right\}$
        Set $(\theta_{k+1}, U_{k+1}) = \begin{cases} (\tilde{\theta}, \tilde{U}) & \text{w.p. } \rho \\ (\theta_k, U_k) & \text{w.p. } 1-\rho \end{cases}$
        Set $k = k + 1$
    until True
```

TABLE II

this case, the last term in Theorem 3 will tend to 0 as k tends to infinity. Therefore, after an initial burn in period, Algorithms I and II will approximately generate approximate optimizers. In some cases it is also possible to determine the rate of convergence. The simplest such case is when the proposal distribution, $g(\theta|\theta_k)$, used in the algorithms is independent of the current state θ_k; in this case we simply denote its probability density function by $g(\theta)$. The following is taken from [32].

Theorem 4: Assume that there exists $M > 1$ such that for all $\theta \in \Theta$, $p(\theta; J, \delta) > 0$, $g(\theta) > 0$, and $p(\theta; J, \delta) \leq Mg(\theta)$. Then $\|\pi(\cdot; J, \delta) - \pi_k(\cdot)\| \leq \left(1 - \frac{1}{M}\right)^k$.
The following fact follows immediately.

Theorem 5: Assume that there exists $M > 1$ such that for all $\theta \in \Theta$, $p(\theta; J, \delta) > 0$, $g(\theta) > 0$, and $p(\theta; J, \delta) \leq Mg(\theta)$. Then

$$\pi_k[\Theta(\epsilon, \alpha)] \geq \frac{1}{1 + \left[\frac{1+\delta}{\epsilon+1+\delta}\right]^J \left[\frac{1}{\alpha}\frac{1+\delta}{\epsilon+\delta} - 1\right] \frac{1+\delta}{\delta}} - \left(1 - \frac{1}{M}\right)^k$$

The conditions of Theorem 4 are easy to meet in our case. We can select the proposal distribution such that $g(\theta) > 0$ and, by construction, we also have that $p(\theta; J, \delta) > 0$. Moreover, because U is assumed to be bounded, if we select g to be the uniform distribution over Θ, then the condition of Theorem 4 will be met for an appropriately large M. This suggests a simple strategy for obtaining an approximate domain optimizer with a given value imprecision ϵ, a given residual domain α with a high enough probability ρ:

1) Select σ and γ such that $(1 - \gamma)\sigma \geq \rho$.
2) Select δ and J using Corollary 1.
3) Run the Markov chain with g the uniform distribution over Θ for $k \geq \frac{\log(\gamma\sigma)}{\log\left(1-\frac{1}{M}\right)}$ steps.

Then the subsequent states of the Markov chain will be approximate domain optimizers with value imprecision ϵ and residual domain α with probability at least $(1 - \gamma)\sigma \geq \rho$. Notice that we have introduced one more design parameter, γ, to determine how much of the probability of obtaining an approximate domain optimizer is attributed to the selection of J (σ above) and how much to the convergence of the Markov chain ($\gamma\sigma$ above).

In some cases this strategy can produce approximate domain optimizers very efficiently. One such case is when the optimization criterion $U(\theta)$ has a "flat top".

Proposition 1: Assume that $\lambda(\Theta^*) \geq \beta\lambda(\Theta) > 0$ for some $\beta \in (0, 1)$. If $K \geq \frac{\log(\gamma\sigma)}{\log(1-\beta)}$ and

$$J \geq \frac{1 + \epsilon + \delta}{\epsilon}\left[\log\frac{\sigma}{1 - \sigma} + \log\frac{1}{\alpha} + 2\log\frac{1 + \delta}{\delta}\right]$$

then $\pi_k(\Theta(\epsilon, \alpha)) \geq (1 - \gamma)\sigma$ for all $k \geq K$.
It is interesting to note that the "flat top" assumption also ensures that, under weak regularity conditions on U (effectively Assumption 1 above), the distribution generated by simulated annealing algorithms ($\pi(\cdot; J, \delta)$ in our case) converges asymptotically (as $J \to \infty$ in our case) to the uniform distribution over the set of global optimizers Θ^* [2].

With the bound of Proposition 1 in place we can do a rudimentary complexity analysis for the computational effort necessary to obtain approximate domain optimizers using Algorithms I and II. In particular we will count the number of random extractions necessary. In this sense, generating the next state of the Markov chain for Algorithm I is roughly the same cost for all J (two random extractions necessary, one for $\tilde{\theta}$ and one for the accept-reject step). Therefore, the total amount of work one needs to obtain an approximate domain optimizer for a deterministic criterion is proportional to K, and hence logarithmic with respect to σ and β. The cost appears to be independent of ϵ and α, but these two parameters will enter in the power J to which we need to raise $U(\tilde{\theta})$ in the calculations.

For an expected value criterion, the computational cost for generating the next state of the Markov chain in Algorithm II is proportional to $J+2$ (one extraction for θ, J for the x, and one for the accept-reject step). Therefore the total amount of computation needed is of the order of $K(J+2)$, which scales rather well (the worst growth rate is $1/\epsilon$ with respect to the value imprecision).

If the "flat top" condition is not met, however, it can be easily seen that the above strategy based on the uniform proposal distribution $g(\theta)$ can lead to very slow convergence of the Markov chain as the following proposition suggests.

Proposition 2: Let π_k denote the distribution of the Markov chain at step k. If

$$k \geq \left(\frac{1 + \delta}{\delta}\right)^J \log\frac{1}{\gamma\sigma}$$

then $\|\pi(\cdot; J, \delta) - \pi_k(\cdot)\| \leq \gamma\sigma$.
Even though this is only a sufficient condition it suggests that the number of steps that the Markov chain needs to converge to within $\gamma\sigma$ of the desired stationary distribution grows exponentially in J. In particular, if we use the value of J established in Corollary 1 and assume that all logarithms are taken with base $(1 + \delta)/\delta$ to simplify the formula, the sufficient condition becomes

$$k \geq \left(\frac{\sigma}{\alpha(1 - \sigma)}\left(\frac{1 + \delta}{\delta}\right)^2\right)^{\frac{1+\epsilon+\delta}{\epsilon}} \log\frac{1}{\gamma\sigma}.$$

The problem here is the implicit dependence of the convergence rate on the exponent J. This is due to the fact that if $\lambda(\Theta^*) = 0$ then the stationary distribution of the chain not only becomes "sharper" as J increases, but the peak of its density function also goes to infinity. Hence the mismatch (encoded by M above) between the stationary distribution of the chain and the uniform proposal distribution used for searching the domain Θ also goes to infinity. It is also interesting to note that in this case the asymptotic convergence ($J \to \infty$) of simulated annealing algorithms can be proven only under additional regularity assumptions (typically differentiability) on U [2], [3], [4], [5].

V. Concluding remarks

We presented a series of results aiming to provide theoretical support for the use of randomized methods for optimization, both in a deterministic and a stochastic setting. Our main results allow us to provide finite sample guarantees for simulated annealing type algorithms for optimization over continuous domains (Theorem 2) and link these bounds to the computation of approximate optimizers in a stochastic programming context (Theorem 1). We also discussed how extractions from the necessary distributions can be generated using MCMC methods (Theorem 5).

For certain combinatorial optimization problems (where the optimization domain is finite) it has been shown that allowing the value of J to increase along the computation (simulated annealing) leads to better performance than keeping it constant [33]. In our case this would imply designing an "annealing schedule" to change J_k as the computation progresses. Current work concentrates on the development of such annealing schedules. In addition to general purpose schedules that can operate under weak assumptions (like Assumptions 1 and 2 above) we are also looking to exploit any available structure (for example differentiability or convexity) to improve the computational performance of the methods. The extension of the results to unbounded optimization domains is another area of current research.

References

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[2] H. Haario and E. Saksman, "Simulated annealing process in general state space," *Adv. Appl. Prob.*, vol. 23, pp. 866–893, 1991.

[3] S. B. Gelfand and S. K. Mitter, "Simulated annealing type algorithms for multivariate optimization," *Algorithmica*, vol. 6, pp. 419–436, 1991.

[4] C. Tsallis and D.A. Stariolo, "Generalized simulated annealing," *Physica A*, vol. 233, pp. 395–406, 1996.

[5] M. Locatelli, "Simulated Annealing Algorithms for Continuous Global Optimization: Convergence Conditions," *J. Optimiz. Theory App.*, vol. 104, no. 1, pp. 121–133, 2000.

[6] P.J. van Laarhoven and E.H.L. Arts, *Simulated annealing: Theory and applications*, Reidel, Amsterdam, 1987.

[7] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli, "Convergence and finite-time behavior of simulated annealing," *Adv. Appl. Prob.*, vol. 18, pp. 747–771, 1986.

[8] B. Hajek, "Cooling schedules for optimal annealing," *Mathematics of Operations Research*, vol. 13, pp. 311–329, 1988.

[9] J. Hannig, E. K. P. Chong, and S. R. Kulkarni, "Relative Frequencies of Generalized Simulated Annealing," *Math. Oper. Res.*, vol. 31, no. 1, pp. 199–216, 2006.

[10] V.N. Vapnik, *The Nature of Statistical Learning Theory*, Cambridge University Press, Springer, New York, US, 1995.

[11] M. Vidyasagar, *A theory of learning and generalization: with applications to neural networks and control systems*, Springer-Verlag, London, second edition, 2003.

[12] L. Blum, C. Cucker, M. Shub, and S. Smale, *Complexity and Real Computation*, Springer-Verlag, New York, 1998.

[13] P. Müller, "Simulation based optimal design," in *Bayesian Statistics 6: proceedings of the Sixth Valencia International Meeting*, J. O. Berger, J. M. Bernardo, A. P. Dawid, and A. F. M. Smith, Eds. 1999, pp. 459–474, Oxford: Clarendon Press.

[14] A. Doucet, , S.J. Godsill, and P. Robert, "Marginal maximum a posteriori estimation using Markov chain Monte Carlo," *Statistics and Computing*, vol. 12, pp. 77–84, 2002.

[15] P. Müller, B. Sansó, and M. De Iorio, "Optimal Bayesian design by inhomogeneous Markov chain simulation," *Journal of the American Statistical Association*, vol. 99, no. 467, pp. 788–798, 2004.

[16] A. Lecchini Visintini, J. Lygeros, and Jan M. Maciejowski, "Approximate domain optimization for deterministic and expected value criteria," Tech. Rep. AUT08-03, Automatic Control Laboratory, ETH Zurich, March 2008.

[17] A. Lecchini-Visintini, J. Lygeros, and J. Maciejowski, "Simulated annealing: Rigorous finite-time guarantees for optimization on continuous domains," in *Neural Information Processing Systems (NIPS07)*, Vancouver, B.C., Canada, December 3-6 2007.

[18] A. Lecchini, W. Glover, J. Lygeros, and J. Maciejowski, "Monte carlo optimization for conflict resolution in air traffic control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 470–482, December 2006.

[19] K. Koutroumpas, E. Cinquemani, P. Kouretas, and J. Lygeros, "Parameter identification for stochastic hybrid systems using randomized optimization: A case study on subtilin production by *Bacillus Subtilis*," *Nonlinear Analysis: Hybrid Systems*, vol. 2, no. 3, pp. 786–802, August 2008.

[20] A. Shapiro, "Stochastic programming approach to optimization under uncertainty," *Mathematical Programming, Series B*, vol. 112, pp. 183–220, 2008.

[21] M. Vidyasagar, "Randomized algorithms for robust controller synthesis using statistical learning theory," *Automatica*, vol. 37, pp. 1515–1528, 2000.

[22] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*, Springer-Verlag, London, 2005.

[23] Yu. Nesterov, "Primal-dual subgradient methods for convex problems," Tech. Rep. 2005/67, CORE, 2005.

[24] J. Linderoth, A. Shapiro, and S. Wright, "The empirical behavior of sampling methods for stochastic programming," *Annals of Operations Research*, vol. 142, pp. 215–241, 2006.

[25] J.-Ph. Vial and Yu. Nesterov, "Confidence level solutions for stochastic programming," Tech. Rep. 2000/13, CORE, 2000.

[26] Z.B. Zabinsky, *Stochastic adaptive search for global optimization*, Kluwer Academic Publishers, Norwell, 2003.

[27] P. Diaconis and D. Strook, "Geometric bounds for eigenvalues of Markov chains," *The Annals of Applied Probability*, vol. 1, no. 1, pp. 36–61, 1991.

[28] S.P. Meyn and R.L. Tweedie, *Markov Chains and Stochastic Stability*, Online edition: http://probability.ca/MT/, 2005, First edition: Springer, London, 1993.

[29] J.S. Rosenthal, "Minorization conditions and convergence rates for Markov chain Monte Carlo," *Journal of the American Statistical Association*, vol. 90, no. 430, pp. 558–566, 1995.

[30] R. Douc, E. Moulines, and J.S. Rosenthal, "Quantitative bounds on convergence of time-inhomogeneous Markov chains," *The Annals of Applied Probability*, vol. 14, no. 4, pp. 1643–1665, 2004.

[31] G.O. Roberts and J.S. Rosenthal, "General state space Markov chains and MCMC algorithms," *Probability Surveys*, vol. 1, pp. 20–71, 2004.

[32] K.L. Mengersen and R.L. Tweedie, "Rates of convergence of the Hastings and Metropolis algorithm," *The annals of Statistics*, vol. 24, no. 1, pp. 101–121, 1996.

[33] I. Wegener, "Simulated annealing beats Metropolis in combinatorial optimization," in *ICALP 2005*, L. Caires, G.F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds., number 3580 in LNCS, pp. 589–601. Springer-Verlag, Heidelberg, 2005.