14

Randomized Algorithms for Control and Optimization

Rudolf Kulhavý

Honeywell Technology Center, Prague and
Institute of Information Theory and Automation, Acad. Sci. CR
182 08 Prague, Czech Republic

14.1 Introduction

A number of problems that industry and businesses are facing today are *hard* to deal with using analytical, calculus-based tools. Such problems include

- estimation of parameters in nonlinear or otherwise complex models,
- design of feedback control law in nonlinear control problems,
- engineering design of complex and not completely understood manufacturing processes,
- design of scheduling rules in ATM switches,
- buffer allocation in production line,
- estimation of rare event probability in high reliability systems,
- routing policy in communication networks,
- logistic system and policy design in global transportation network.

The common characteristics of such problems are lack of structure (resulting from combination of combinatorial, discrete, and symbolic variables), inherent presence of uncertainties (requiring time-consuming averaging), and huge search space (not easy to parameterize and prone to combinatorial explosion).

Hard problems can be solved only approximately. The purpose of this paper is to show that randomized algorithms based on statistical simulation of variant models and designs yield a consistent and systematic framework for design and analysis of *approximate* solutions to such problems.

14.2 Statistical Simulation

If we cannot solve the problem *exactly*, we must settle for solving it *approximately*. If we cannot solve *all* instances of a problem, we must settle for solving *almost all* of them. This is a recurring idea of many of the recent approaches to solving hard problems.

14.2.1 Sampling from population

One does not need to ask the whole electorate to learn the share of votes for presidential candidates. A sufficiently precise answer can be obtained by addressing much a smaller sample of voters. The trick is that the *sample* copies the structure of the whole *population* in all its essential characteristics (age, sex, education, religion, ethnic origin, etc.).

The fact that we can learn all essential about the population by analyzing its sample is used now routinely in opinion polls, retail surveys, environmental monitoring, and statistical process control, to

name but a few applications. Substitution of a collection of samples for the whole population effectively breaks down the size of the original problem.

14.2.2 Monte Carlo simulation

Repetitive computer simulation of a system model – starting from various initial conditions and running with different sets of parameters – has become one of the engineer's major tools. Systematic exploration of the parameter space involves, however, prohibitive computations, even for problems of moderate complexity. We can alleviate this constraint if we replace the 'population' of all possible parameter values by a 'sample' of their representative values.

The prerequisite is to know the *probability distribution* of key quantities (such as parameters or states) of the system under study. Once the distribution is known, the simulation proceeds by randomly sampling from it. The desired result is taken as an average of the simulation outcomes over the number of simulation runs. Furthermore, we can predict the statistical error in this average result, and hence estimate the number of trials needed to achieve a prescribed error limit.

The above forms the essence of the Monte Carlo method whose origin dates back to the 1940s. The Monte Carlo simulation has been used since then in applications as diverse as nuclear reactor design, quantum chromodynamics, radiation cancer therapy, traffic flow, stellar evolution, econometrics, Dow-Jones forecasting, oil well exploration, and VLSI design.

14.2.3 Stochastic models

The models entering the Monte Carlo simulation must be *stochastic* – to define the probability distribution of the variables of interest. This may seem a restrictive and artificial assumption, especially as most of the first-principle models of physical and chemical processes are represented by a set of *deterministic* differential equations. A closer look shows, however, that data always exhibit some kind of random behavior. This may be due to a physical noise affecting the measurements but, more typically, it is a consequence of the incomplete knowledge of all the relevant variables. The 'hidden' variables affect the system behavior in a way that cannot be fully predicted.

Consider, for instance, a continuous stirred tank reactor where the temperature and concentration inside the reactor are measured but the feed temperature and concentration are not monitored, being supposed to be constant. Any variation in the feed parameters produces a deviation from the ideal model. As a result, we have two sources of randomness – the imprecision of sensors and the imprecision of the model. Both can be modeled as external noises corrupting the process states and the sensor measurements.

14.2.4 MC computation engine

The Monte Carlo simulation can be applied even in cases that have no apparent stochastic content, such as the evaluation of a definite integral or the inversion of a system of linear equations. One can still pose the desired solution in terms of probability distributions. Even though the transformation may seem artificial, this step allows the system to be treated as a stochastic process for the purpose of simulation and hence Monte Carlo methods can be applied to simulate the system.

This is crucially important for implementation of statistical methods that involve operations with probabilities, such as fitting of a probability distribution to given data and computation of marginal and conditional distributions of selected variables from a given joint distribution. The probabilistic operations entail computation with functions (infinitely dimensional objects) and multivariate integration (extremely difficult in high dimensions). All these operations can be approximated using sample-based computations; for instance, expectation of a given function can be replaced with its sample average.

14.3 Managing Complex Models

Consideration of increasingly complex models stresses the importance of consistent handling of uncertainty in estimation. Although the use of complex models generally decreases the approximation error between the actual and model behaviors, the estimation uncertainty typically increases – we are left with more degrees of freedom but the same amount of data.

Choosing a compromise between the approximation error and estimation uncertainty (known as bias-variance trade-off in the mean squared error estimation) is an inevitable step in any statistical modeling. In the past decades, engineers usually preferred making the estimation uncertainty small and negligible, even at the cost of adopting simplifying model assumptions, such as linear dynamics and Gaussian stochastics.

This approach can hardly be followed nowadays when trying to solve some of the hard problems mentioned in the introduction. Rather we must get used to the fact that estimation will leave us with significant residual uncertainty about the resulting model. This situation boosts the need to have a systematic framework for describing and propagating the uncertainty (as the new data come). It is increasingly accepted that 'probability' and 'probability calculus laws' are ideally suited for the job.

14.3.1 Bayesian paradigm

It sounds very natural to describe the random behavior of data through its probability distribution. One can describe, however, in terms of probability even the uncertainty of assigning such a distribution. Probability thus appears in a twofold role – as a relative frequency of possible outcomes of a certain experiment in a long run, and as a measure of uncertainty of unknown (not necessarily stochastic) quantities.

Jacob Bernoulli, Thomas Bayes, and Pierre Simon Laplace came with the idea of 'inverse' probability as early as in the 18th century. Much later, in this century, it was shown that the rules of consistent reasoning under uncertainty coincide essentially with the laws of probability calculus.

The symmetric view of probability, usually ascribed to T. Bayes, has turned out useful in many areas. It has brought some unique features into statistics, creating its self-contained branch. It has put artificial intelligence on a firm basis and set up a reference solution for design of expert systems. It has been applied successfully in physics, engineering, and statistics to ill-posed (underdetermined) problems, which commonly appear in system identification, fault detection, econometrics, control, medical diagnosis, geophysical exploration, image processing, synthesis of electrical filters or optical systems.

The pros and cons of the Bayesian paradigm can be well illustrated on the case of stochastic adaptive control. Suppose we are to control a dynamic system that depends on some unknown parameter. We have two options then. Either we satisfy with a point estimate of the parameter, or we take its uncertainty into account. In the latter case, the expectation in a cost function applies to both the stochastic behavior of the system and the uncertainty of the parameter. This converts the original problem into a hyperproblem that has no more unknowns. Its information state is formed by the probability density function of the original state *and* the parameter, conditional on the observed data.

The beauty of the result is that it looks as if all uncertainty vanished. As soon as the prior density is set, the state evolves in a definite way, governed by the laws of probability theory. The appeal of the solution is paid, however, by the immense dimension of the information state. Unless the problem has a finite-dimensional statistic, there is no feasible way of updating the full information state, and an approximate posterior density has to be evaluated numerically.

Here comes the opportunity for randomized algorithms. Their bottom line is simple: the posterior distribution is approximated with a long enough sequence of samples, and probabilistic calculations are replaced with the corresponding sample averages. Their implementation can, however, become a nightmare in high dimensions.

14.3.2 Why Monte Carlo?

We have already indicated that the Monte Carlo method is a fairly old invention. The recent wave of renewed interest may thus look a bit of surprise, though there are good reasons for such a comeback.

First, the steady increase in the computing power accompanied with relative decrease in computer prices have created the situation when one can afford routine implementation of the Monte Carlo simulation on fairly standard PCs.

Second, the methods based on linear models have approached their performance limits. It is not that these methods should disappear from our toolboxes, we just cannot expect any more surprises and significant improvements compared with the state of the art. This has created a serious incentive for new paradigms to be tried.

Third, new powerful algorithms of sampling have appeared. The Monte Carlo efficiency is determined by the complexity of generating sample values from a given distribution. While in one-dimensional case the problem represents a textbook exercise with numerous possible solutions, in higher dimensions it was for long time practically infeasible. The breakthrough in this area has been marked with the advent of Markov Chain Monte Carlo (MCMC) methods.

14.3.3 Classical Monte Carlo

Before we proceed to MCMC algorithms, we recall two traditional techniques of sampling from complex probability distributions.

14.3.3.1 Importance sampling

The trick behind the importance sampling (Tierney, 1994) is drawing of samples from a *proposal* distribution that is similar to the target one, but considerably simpler to sample from. After a collection of samples from the proposal distribution is drawn, the sample set is processed so as to become a set of samples from the target distribution. The idea of such a post-processing is extremely simple – the target samples are obtained by resampling the intermediate samples with probabilities proportional to the ratio of the target and proposal probabilities.

The technique itself is a simple consequence of elementary probability calculus laws. It appeared explicitly as *sampling/importance resampling* in connection with drawing missing data patterns (Rubin, 1988). Soon after it was applied – under name *weighted bootstrap* – to estimation of the posterior density in Bayesian estimation (Smith and Gelfand, 1992; cf. Efron, 1982).

Importance sampling works efficiently if the auxiliary distribution is close enough to the target distribution. In the opposite case, the result of importance sampling can be a degenerate set of samples – repeating just a couple of representative points whereas the majority of the intermediate samples never get a chance. This behavior is only stressed in high dimensions.

Importance sampling is ideally suited for recursive Bayesian parameter estimation where the prior distribution is a natural candidate for the proposal distribution. It can be extended even to recursive state estimation, where the samples are additionally propagated through the state equations.

14.3.3.2 Rejection sampling

Rejection sampling (Ripley, 1987) also starts by drawing samples from a simpler proposal distribution. But, in contrast to importance sampling, the samples are immediately accepted or rejected – using a random mechanism based on comparison of the proposal and target probabilities.

Once again, rejection sampling works well if the proposal distribution is similar to the target distribution. The acceptance rate decreases to zero exponentially fast as the dimension of the underlying space increases. Hence, rejection sampling cannot be recommended for generating samples from high-dimensional distributions, but it is a very useful method for one-dimensional (sub)problems.

14.3.4 Markov Chain Monte Carlo

The first algorithm of the MCMC type appeared in statistical physics as early as in the 1950s (Metropolis *et al*., 1953). It took, however, three decades before the MCMC algorithms were reinvented in global optimization and in Bayesian image restoration. In the early 1990s (Gelfand and Smith, 1990), the MCMC algorithms penetrated Bayesian statistics. Since then, the MCMC methods have become a *de facto* standard tool for approximation of probabilistic computations.

All the MCMC methods are based on a very simple idea. Suppose you need to draw a sample from a complex multivariate probability distribution. Instead of direct sampling from the distribution, you

1. design a Markov chain (stochastic process that is fully described by probabilities of all possible state-to-state transitions) whose stationary distribution coincides with the target distribution,
2. simulate the Markov chain in your computer,
3. take the simulated values as samples from the target distribution (neglecting perhaps samples coming from the initial, 'burn-in' period).

As there are many possible ways of building such a Markov chain, various MCMC algorithms have been proposed in the course of time. From the multitude of methods and algorithms published in the last decade, four major classes of MCMC algorithms can be distinguished.

14.3.4.1 Metropolis sampler

We have seen that the importance and rejection sampling are efficient enough only if the proposal distribution is similar to the target distribution. In high dimensions, the design of an appropriate proposal distribution becomes a truly challenging task. The Metropolis algorithm, instead of looking for a simpler approximate distribution, makes use of a proposal distribution that depends on just the current state.

More specifically, the algorithm simulates an auxiliary stochastic process that suggests possible candidates for the samples. At every step the candidate samples are either *accepted* or *rejected* with probabilities determined by the stochastic process and the target distribution. Two most frequently met clones of Metropolis sampler (Tierney, 1994) are the *random walk chains*, which use a random walk as a sample-proposing stochastic process, and *independence chains*, which take the candidate samples from a fixed probability distribution.

The Metropolis algorithm was originally introduced by Metropolis *et al*. (1953) for computing properties of substances composed of interacting individual molecules. Hastings (1970) relaxed the symmetric Markov chain assumption, which made it possible to consider the option of sampling from a fixed distribution. Since then, numerous modifications to the basic algorithm have been proposed, which make use of proposal distributions that give faster movement through the state space.

Example 14.1. A simple method of signal validation is to model the distribution of signal differences through a convex combination of three normal distributions with close-to-zero, moderate, and large variances corresponding to sensor failure, normal operation, and gross error, respectively. Assuming for simplicity that the variances can be fixed beforehand, the model has two unknowns—the probabilities of sensor failure and gross error. Estimates of these two probabilities can serve as an indicator of the sensor health.

Figure 14.1 shows the posterior probability density function of the unknown probabilities, evaluated over a very dense grid of the parameter values. Note that many points in the parameter space are assigned zero probability. Evaluation at these points is avoided if only a sample from the posterior density is simulated using a Metropolis sampler. Given the sample, the posterior density can be approximated, as illustrated in Figure 14.2.

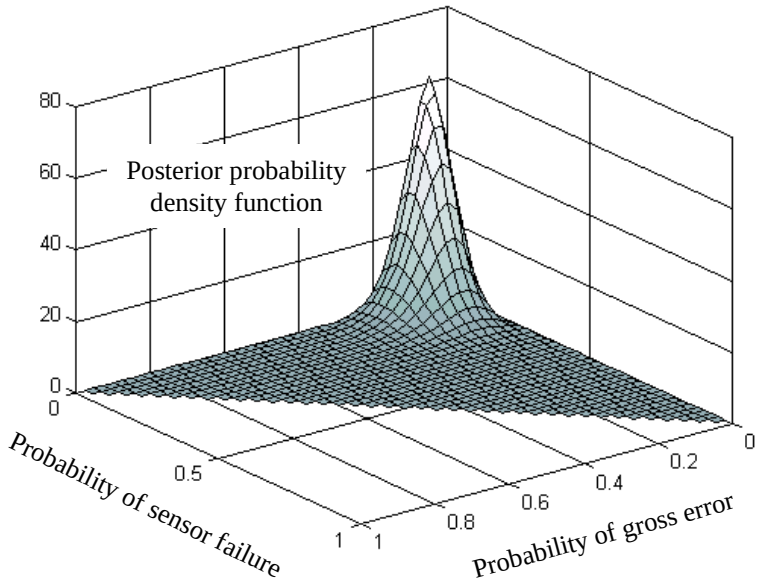

Figure 14.1 An example of the posterior probability density function for the unknown parameter being composed of the probabilities of sensor failure and gross error.

14.3.4.2 Gibbs sampler

The algorithm samples component-by-component from the full *conditional* distributions rather than directly from the *joint* distribution. The conditional distributions are typically one-dimensional and thus much easier to sample from. The Gibbs algorithm can also be extended to the case of a parameter vector composed of several blocks; sampling then goes block-by-block.

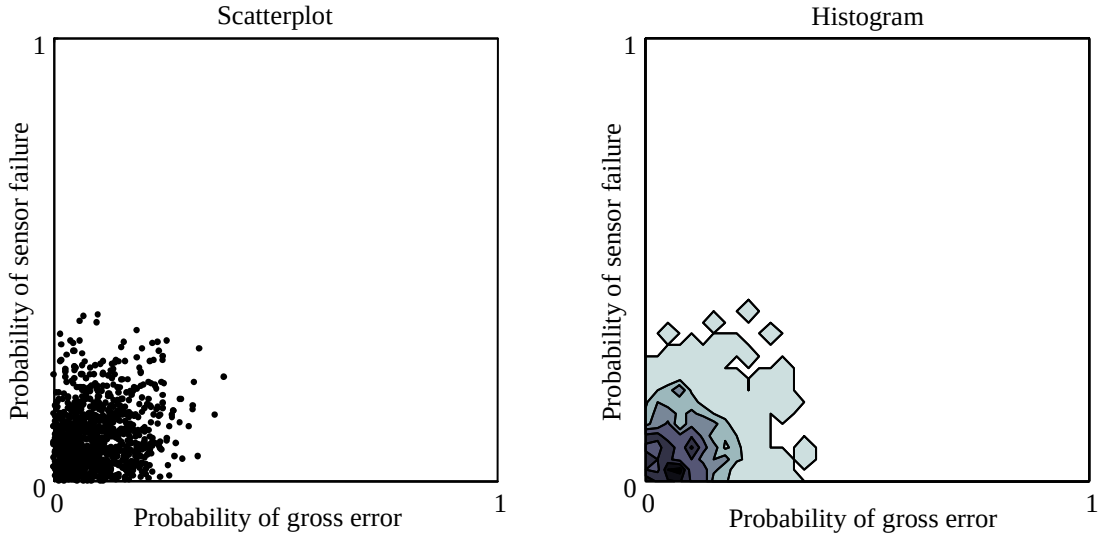

Figure 14.2 Sample probabilities of sensor failure and gross error drawn from the posterior probability density function using a Metropolis sampler.

The Gibbs sampler is a natural option in estimation of complex hierarchic or structured models. Consider, e.g., the problem of joint estimation of state and parameter values. The Gibbs sampler suggests basically switching between

1. sampling of states given the previous parameter values, and
2. sampling of parameters given the previous state values.

Similarly, estimation of change time for a signal proceeds by taking samples of

1. the initial level given the terminal level and change time,
2. the terminal level given the initial level and change time, and
3. the change time given the initial and terminal levels.

The Gibbs sampler appeared first in the image processing literature (Geman and Geman, 1984). In the early 1990s, it was presented to the Bayesian community (Gelfand and Smith, 1990) where it has quickly become a tool of choice.

Note that to sample from the one-dimensional full conditionals, one can use the Metropolis algorithm where the candidate sample is taken from a kernel-smoothed estimate of the target density based on the previously accepted samples. This is an example of so-called *hybrid* algorithms, which try to combine the advantages of different sampling algorithms.

An idea similar to the Gibbs sampler is used in the *hit-and-run algorithm* (Schmeiser and Chen, 1991) where sampling is made in randomly chosen directions rather than dimension-by-dimension.

14.3.4.3 Langevin sampler

The algorithm is based on simulation of a Langevin stochastic differential equation, driven by the gradient of logarithm of the target density and perturbed by a standard Brownian motion. The solution to the Langevin equation is known to be asymptotically distributed according to the target distribution (Rossky, Doll and Friedman, 1978). The Langevin sampler is an example of a molecular dynamics method, which has been used originally in statistical physics to compute phase space trajectories of a collection of molecules, which individually obey classical laws of motion (Heermann, 1990).

The Langevin equation needs to be discretized before it can be implemented in digital computer (Amit, Grenander and Piccioni, 1991). The choice of the discretization period crucially affects the sampler performance; a too long period results in a significant deviation of the sample distribution from the target one, a too short period calls for unnecessarily many samples. The major advantage of the Langevin sampler is the simplicity of the algorithm as the gradient of logarithm of the target density can often be precomputed analytically.

14.3.4.4 User's choices

The application of MCMC algorithms requires solving a number of practical issues such as:

- What kind of MCMC algorithm fits best the problem under study?
- How long simulation is 'long enough'?
- How long the initial 'burn-in' period should be?
- Is it better to run one long simulation or rather a number of shorter ones?

- How to choose the starting point (if required)?
- Can another parameterization be of help?

No simple answers exist to these questions. The truth is that the MCMC technology provides a set of tools that need to be used thoughtfully and with care. There is no free-lunch solution to the general sampling problem. On the other hand, no other technology can compete currently with the MCMC in the complexity of problems resolved.

14.4 Managing Complex Designs

Many human-made system problems, such as manufacturing automation, communication networks, computer performances, and resource allocation problems, involve combinatorics rather than real analysis. The objective of optimization is here proposing an ideal configuration rather than tuning the design parameters. Search for optima of functions of discrete variables is known as *combinatorial optimization*.

Even today, many of large-scale combinatorial optimization problems can only be solved approximately, which is closely related to the fact that these problems have been proved NP-hard. Such problems are not solvable by an amount of computation effort that is bounded by a *polynomial* function of the size of the problem. When the optimization algorithm yields a globally optimal solution in a prohibitive amount of computation time, an option is to use an approximation algorithm, which yields an approximate solution only, but in an acceptable amount of computation time.

Simulated Annealing

Simulated annealing is a modification of the Metropolis algorithm where sampling goes from a probability distribution parameterized by a scalar parameter, called temperature or control parameter. The temperature is decreased in between subsequent Markov chains so that it is approaching asymptotically zero. The sampling distribution is related to the optimized function in such a way that it concentrates for decreasing values of the temperature over the function's local optima. If the cooling scenario is sufficiently slow, simulated annealing is able to locate even the global optimum.

Metropolis *et al.* (1953) proposed their sampling algorithm originally for an efficient simulation of the evolution of a solid to thermal equilibrium. It took about thirty years before Kirkpatrick, Gelatt and Vecchi (1983) and, independently, Cerny (1985) realized there exists an analogy between 'minimizing the cost function of a combinatorial optimization problem' and 'slow cooling of a solid until it reaches its low energy ground state', and that the optimization process can be realized by applying the Metropolis algorithm.

By substituting cost for energy and by executing the Metropolis algorithm at a sequence of slowly decreasing temperature values, Kirkpatrick, Gelatt and Vecchi (1983) obtained a combinatorial optimization algorithm, which they called simulated annealing. The algorithm is known now also as Monte Carlo annealing, statistical cooling, probabilistic hill climbing, stochastic relaxation, or probabilistic exchange algorithm.

Solutions, obtained by simulated annealing, do not depend on the initial configuration and have a cost usually close to the minimum costs. Furthermore, it is possible to give a polynomial upper bound for the computation time for some implementations of the algorithm. The general applicability of simulated annealing is sometimes undone by the computation effort. A number of modifications and improvements in the MCMC vein have been suggested since the pioneering 1983 paper.

User's choices

The practical implementation of simulated annealing requires to define an annealing schedule, which involves the choice of initial temperature, decision how many iterations are performed at each temperature (in other words, how long the Markov chain is to be), and how much the temperature is decremented at each step as cooling proceeds. The cooling scenario required for the algorithm to end up (with probability one) in a global optimum is overly slow for most optimization problems. Much faster scenarios are thus used in practice, such as a geometric one proposed first by Kirkpatrick, Gelatt and Vecchi (1983).

The choice of a particular cooling schedule affects the final cost achieved with simulated annealing. Randelman and Grest (1986) showed empirically that the expected final cost for a traveling salesman problem depends logarithmically on the *cooling rate*, defined by the ratio of a temperature decrease and Markov chain length, both being supposed fixed for simplicity. It implies that simulated annealing

leaves no room for cheating; short Markov chains call for slow cooling, fast cooling requires longer Markov chains, i.e., more samples to be generated by the Metropolis algorithm. The cooling rate defines then how close one can get on average to the global optimum.

Simulated annealing has been used in various combinatorial optimization problems in such diverse areas as computer-aided design of integrated circuits, analytical chemistry, molecular modeling, image processing, code design, neural network theory, scheduling problems, etc. (Laarhoven and Aarts, 1987).

Example 14.2. Figure 14.3 illustrates application of simulated annealing to batch recipe optimization. The batch process was pyrrole synthesis subject to hard constraints on minimum product amount, maximum by-product concentration, and maximum feedstock concentration. The objective of optimization was to minimize the makespan indicated in Figure 14.3 through a contour plot. The actual makespan was not known util the recipe was applied. The area outside the apparent "valley" corresponds to infeasible batch recipes. The globally optimal recipe is marked with a diamond. The simulated annealing used a Metropolis sampler with independence chains drawn from a kernel-smoothened density estimate based on the previous batch runs. The global behavior of the optimizer depends on the bandwidth used for kernel smoothing. The left-hand plot shows the optimizer behavior for a bandwidth that yields a good trade-off between the speed of search and the number of off-spec batches. The right-hand plot illustrates the same optimizer behavior for a very small bandwidth resulting in "cautious" search with no lost production.

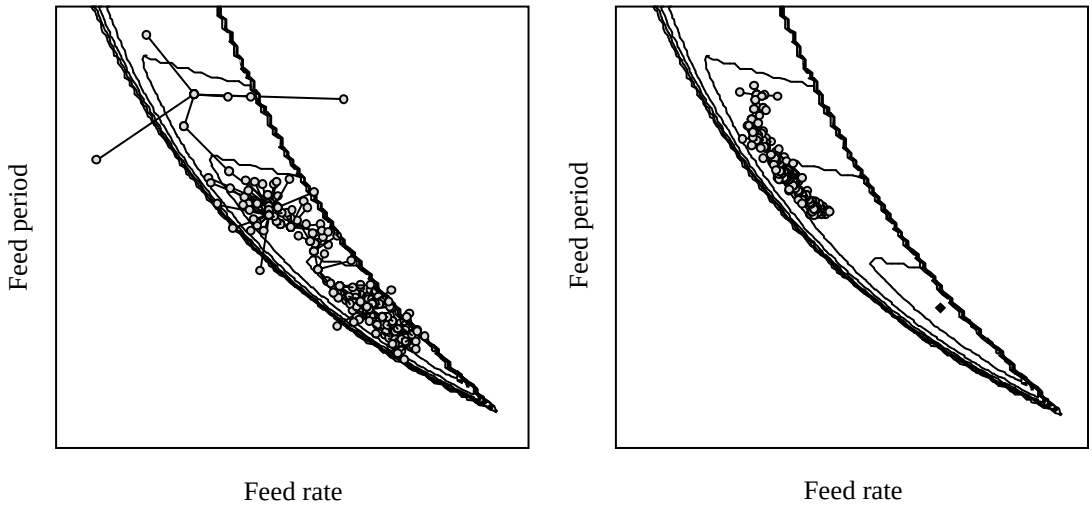

Figure 14.3 Application of simulated annealing to batch recipe optimization. Circles denote recipes proposed, lines connect past recipes with their descendants obtained through random perturbation resulting from kernel smoothing.

Evolutionary Algorithms

Evolutionary algorithm is an umbrella term for any computer-based solution using evolution as a key element in its design and implementation. In the optimization context, evolutionary algorithms maintain a population of candidate solutions that evolve according to rules of selection and reproduction. Each candidate solution in the population is assigned a measure of its fitness. The algorithm proceeds by choosing a sub-population of high-fitness candidates for offspring production achieved typically through recombination and mutation of selected candidates followed by the survivor selection. The final selection gives rise to a new population (generation) of candidate solutions. The procedure repeats until a stopping condition (time, fitness) is satisfied.

Two typical representatives of evolutionary algorithms are *evolutionary programming* (Fogel, Owens and Walsh, 1966) and *genetic algorithms* (Koza, 1992). The typical genetic algorithm approach involves encoding the problem solutions as a string of representative tokens, the genome. In evolutionary programming, the representation follows from the problem. As a result, evolutionary programming is well suited for optimization of continuous functions whereas genetic algorithms are better adapted for combinatorial optimization. In evolutionary programming, the only source of

diversity is mutation. In genetic algorithms, diversity is guaranteed primarily through recombination, i.e., by exchanging genes between two "parents" to produce two "children".

Evolution and randomness

The selection, recombination and mutation involves always a certain element of randomness. It is present in the random selection of individual solutions from the population, the random choice of crossover points, the random perturbation (i.e., mutation) of the population, the random selection of survivors to form a new generation. Evolutionary algorithms often use a stochastic tournament concept based on sampling randomly several solutions from the population and then selecting one with the highest fitness. The use of a random mechanism give a chance (though not equal one) to each solution in the population. As a result, evolutionary algorithms do not tend to get stuck on a locally optimal solution and can often find a globally optimal solution.

Viewed as randomized algorithms, evolutionary algorithms and simulated annealing have much in common. Simulated annealing using Metropolis sampler with independence chains or with *parallel* random walk chains can be regarded as an example of evolutionary programming with very specific definitions of the fitness function and the selection and mutation operators. Note that the statistical theory behind MCMC algorithms gives significantly more guidance in defining the key parameters of stochastic optimization. On the other hand, a very loose framework of evolutionary algorithms provides the user with more freedom to take the advantage of application-specific features of the optimization problem.

Ordinal Optimization

In many real-life problems, the performance evaluations are corrupted with large "noises", which reflect the inherent random nature of the optimized system. The performance measure is then defined as *expectation* of the underlying utility or cost function. For such problems, even statistical simulation can appear a prohibitively expensive and time-consuming solution.

In *stochastic* optimization, one faces – in addition to the combinatorial explosion or NP-hard limitation – the problem of efficient calculation of the expected performance. In complex cases, the expectation can be evaluated only numerically, typically by sampling. Uncertainty (confidence interval) of such an estimate decreases with square root of the number of samples. In other words, for each order of magnitude increase in the accuracy, two orders of magnitude increase in computing cost must be paid. This may become a prohibitive burden.

The idea of ordinal optimization is based on the observation that if we

- take enough samples of the performance measure,
- order the performances, and
- take the best N designs under this ordering,

we find among those designs – *with high probability* – designs that are *good enough*, i.e., belonging to the top designs for the original problem. The above holds even if the performance values are quite simple (noisy) estimates of the actual performance. The designs found this way can serve as initial points for subsequent, more precise analysis.

Why it works

Technically speaking, the ordinal optimization is based upon two facts. First, 'order' converges exponentially fast while 'value' converges much slower (with square root of the number of sampled designs). That is, being interested primarily in ordering, we do not need to spend so much time on the performance evaluation. The fact that it is much easier to compare values than to determine the actual difference is what distinguishes the 'ordinal' optimization from the traditional, 'cardinal' one.

Second, the probability of *not* finding a good enough design by randomly searching the set of possible designs decreases exponentially fast with the size of the good-enough set. The probability that *none* of 1,000 uniformly sampled designs belongs to the top 5% of the design space is of an order 10^{-23}. The probability that *less than five* of 1,000 uniformly sampled designs belongs to the top 5% of the design space is still of an order 10^{-17}. These figures, which apply regardless of the size of the design set (Ho and Lau, 1997), confirm our common sense: the more we soften the goal, the more probable (easier) is to find a solution.

In a nutshell, the ordinal optimization replaces the *best for sure* with the *good enough with high probability*. It works even with a blind search, i.e., uniform sampling of the search space. Naturally, by

taking advantage of a specific structure of the optimization problem, the search behavior can be significantly improved.

Ordinal optimization was introduced by Ho, Sreenivas and Vakili (1992) in analysis of discrete event dynamic systems. The actual benefit of ordinal optimization depends on the shape of the performance value distribution, which may be skewed in a particular problem towards bad designs. As a result, getting within 1% of the best in order is not necessarily within 1% of the optimal value. This is critical if the performance measure quantifies the actual financial performance. The absolute size of the design set needs to be taken into account as well. Getting within 1% of a search space of 10^{10} points can still be 10^8 away from the best.

Apparently, the ordinal optimization can serve as a fast route to 'good enough' starting points for subsequent analysis. Only this property is likely to give it a lot of use.

14.5 More Applications

Statistical simulation has been applied in the last 10–15 years successfully in many areas. Often the authors of the pioneering contributions were unaware of similar efforts going on in the other fields. Only recently, a bigger picture has started to appear, indicating that we are witnessing here a real shift of paradigm, which is likely to reshape dramatically our current view of modeling, control, and optimization.

Bayesian Estimation

The seminal paper by Gelfand and Smith (1990) has brought the Gibbs sampler to the attention of Bayesian statisticians. The impact of Gibbs and Metropolis samplers has been enormous and truly revolutionized the whole field. The Bayesian methodology interpreting everything unknown as random and uncertain was notorious for extremely difficult computations. Very few could be computed for models other than linear and Gaussian ones. After 1990, hundreds of long-resisting problems that were considered before infeasible have been successfully resolved. The MCMC methods have allowed the statisticians to focus their attention to careful modeling – considering complex hierarchical, non-linear, and non-Gaussian models.

Nonlinear Filtering

The state estimation for processes with highly non-linear and possibly non-Gaussian dynamics has been a challenge to control theory since the late 1960s. Randomized algorithms have been applied to the problem since the early 1990s, providing a conceptually simple solution, even though its practical implementation may call for additional tricks, in addition to lots of computation power. Gordon, Salmond and Smith (1993) applied *importance sampling* to a multiple target tracking problem, turning the estimation problem basically into parallel simulation of a sufficient number of state trajectories. O'Sullivan *et al*. (1993) proposed to solve the target tracking problem using the *Langevin sampler*, considering the option of massively parallel computation of the gradient of logarithm of the state density.

Bayesian Image Restoration

Bayesian estimation of the whole image is an extremely high-dimensional problem where the size of the unknown parameter vector, which is determined by the number of pixels to be restored, is typically in order of 10^5–10^6! Geman and Geman (1984) resolved the problem by introducing a two-dimensional version of the Gibbs sampler, taking the advantage of local correlation of data over a small enough neighborhood of the current pixel. The paper has influenced profoundly the Bayesian statistics, image processing, and neural networks communities.

Robust Control Design

In robust control design, given a family of plants and a family of controllers, the objective is to find a single fixed controller that performs reasonably well for almost all plants. The randomized algorithm

proposed by Vidyasagar (1997) to find a '*probably approximate near minimum*' of an objective function proceeds by taking a sufficient number of samples of plants and controllers, evaluating the sample average of the objective function for each controller, and finding the controller minimizing the sample average.

There have been many other attempts to use randomized algorithms in control and optimization, including a MCMC implementation of predictive control (Chen, Geisser and Geyer, 1993) and randomized implementation of dynamic programming (Rust, 1993).

Network Computing Models

Recently *Microsoft Research* has established a new fundamental research group that focuses on a statistical physics approach to discrete probability theory, combinatorics and theoretical computer science. As computer systems and networks become increasingly large and complex, statistical physics appears the appropriate language to describe their operation. Probabilistic methods can be employed to predict the average behavior of these systems, and to identify potential phase transitions. Combinatorial methods can be used to "count" configurations contributing to a given type of behavior.

14.6 Practical Aspects

The unprecedented power of stochastic simulation does not come for free. The theoretically optimum procedures can rarely be implemented in practice without making further compromises.

Probabilistic Behavior

Compared with closed-form analytic solutions or iterative calculus-based optimization schemes, which currently prevail in industry, stochastic simulation has a probabilistic behavior, which requires from the end user some caution in interpreting its results.
Time available for managing complex models or designs is rarely sufficient for approaching the global optimum. The search is often stopped well before that.

Due to its probabilistic nature, stochastic simulation may return to the same question a (more or less) different answer, depending on a particular random sequence drawn.

Stochastic simulation may even fail occasionally, being increasingly vulnerable to the quality of the (pseudo)random sequence generated by the computer.

The human check of results produced by stochastic simulation is welcome or even required. Otherwise, the algorithm needs to be complemented with additional, robustifying measures as suggested below.

The fact that the current manufacturing conditions and business environment are changing very quickly makes things only worse. Attempts to solve the modeling and optimization problems long enough, 'once forever' make little sense here. Rather we need a way of steadily moving in the "right direction" – improving the system performance and adapting to changes at every step.

Robust Simulation

As so often in engineering, the process knowledge and problem understanding makes life a bit easier than it may look through the general theory perspective.

Any prior information or structural assumptions on the problem under study can limit significantly the search space. Time spent on gathering and incorporating such information is almost always rewarded by faster and more reliable simulation.

Instead of trying to explain the behavior of a complex process using a single, global model, it is often sufficient to fit its response to the current operating conditions and current control scenario. The use of multiple or local models reduces radically the complexity of the original problem (cf. Figure 14.1).

Similarly, instead of trying to construct a global decision rule, it is often sufficient to search for a local decision rule working well for the current operating conditions. 'Local design' appears here as a natural counterpart to 'local modeling'.

The past data experience need not be ignored any more. The current performance of computer, database, and communication technology allows us to retrieve in real time any part of relevant process history, including controls applied in the past.

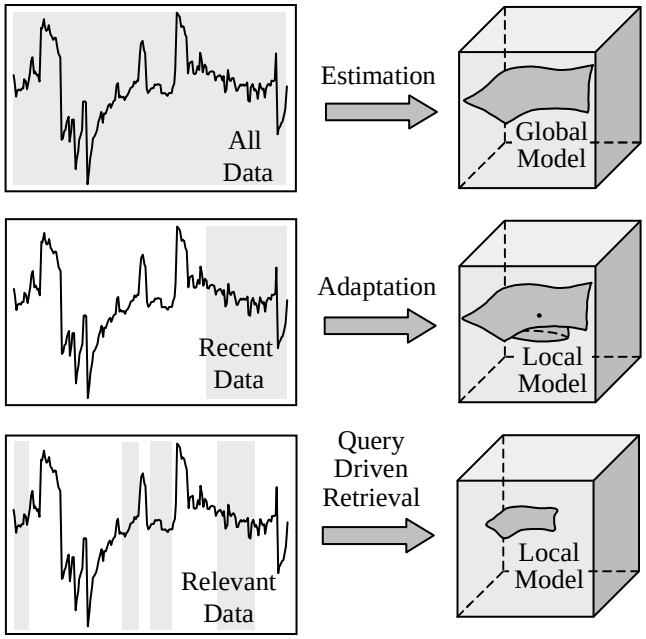

Figure 14.4 Comparison of global, local-in-time, and local-in-data modeling approaches. The plots show time series of data, the data cubes indicate the model-based data fits.

With process history available, optimization can start from the best design applied in the past under similar conditions. This option turns standard iterative optimization into optimization with iterations "spread in time". Searching around best practices eliminates repeating old mistakes while changes to the design are made only when likely (in a probabilistic sense) to improve the process performance.

Last but not least, models and designs that might result in lost or off-spec production, or equipment failures and breakdowns, have to be eliminated explicitly by force. Such a rule-based protection layer – always met in practical applications – can also be interpreted as a special kind of prior information about the problem.

Scope of Applications

Stochastic simulation has become already a powerful tool for off-line multidimensional data analysis, off-line estimation of complex, hierarchic or otherwise structured models, and advanced decision support. In these cases, human experience and intuition can easily be involved if the simulation results leave the end user with too much uncertainty.

Penetration of stochastic simulation into *automatic* control and decision-making is likely to be slower. Apart from purely technical issues, such as computer performance sufficient for real-time applications (and price for it), there are liability issues and psychological barriers here as well.

"No bridge is built to survive everything." Actually, no technical work is built this way; the engineering design has always been a compromise between functionality and price. Software, however, has been perceived for years as a domain where one is in 'full control'. This does not seem to be true any more, even if we put aside the vital issue of verification of increasingly complex computer programs. The problems met currently in industry and business are of such complexity that we cannot hope for their analytical solution. The end user must thus face the fact that software can fail occasionally just because feasible solutions have to be approximate, incomplete, and cost compromising.

14.7 Conclusion

The randomized algorithms are no panacea, but they offer a systematic and consistent approach to hard problems. The Monte Carlo computation engine is likely to turn our good old *error-free* computers, where the problem has been solved exhaustively or not solved at all, into *error-controlled* probabilistic machines, where the problem is solved (with high probability) always, only with larger or smaller error.

While the key algorithms are now ready to use, a number of practical issues remain to be resolved. For instance, in safety-critical applications, such as flight control or control of complex chemical reactions, the occasional drop in the algorithm performance (due to its random nature) needs to be detected and eliminated using a conventional stand-by controller. The convergence of the sample distribution to the target distribution has to be monitored. A supervisory logic needs to be developed so as to decide automatically about the initial conditions, length of simulation runs, algorithm stopping, resetting, etc.

14.8 References

Amit, Y., Grenander, U. and Piccioni, M. (1991) Structural image restoration through deformable templates, *Journal of the American Statistical Association*, **86**, 376–387.

Cerny, V. (1985) Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, **45**, 41–51.

Chen, L.-S., Geisser, S. and Geyer, C.J. (1993) Monte Carlo minimization for sequential control, Technical Report No. 591, School of Statistics, University of Minnesota.

Efron, B. (1982) The Bootstrap, Jacknife and other resampling plans, SIAM, Philadelphia.

Fogel, L.J., Owens, A.J. and Walsh, M.J. (1966) Artificial intelligence through simulated evolution, Wiley, New York.

Gelfand, A.E. and Smith, A.F.M. (1990) Sampling based approaches to calculating marginal densities, *Journal of the American Statistical Association*, **85**, 398–409.

Geman, S. and Geman, D. (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721–741.

Gordon, N.J., Salmond, D.J. and Smith, A.F.M. (1993) A novel approach to nonlinear/non-Gaussian Bayesian state estimation, *Proceedings of IEE-F*, **140**, 107–113.

Hastings, W.K. (1970) Monte Carlo sampling methods using Markov chains and their applications, *Biometrika*, **57**, 97–109.

Heerman, D.W. (1990) Computer simulation methods in theoretical physics, 2nd edition, Springer-Verlag, Berlin.

Ho, Y.C., Sreenivas, R.S. and Vakili, P. (1992) Ordinal optimization in DEDS, *Journal of Discrete Event Dynamic Systems*, **2**, 61–68.

Ho, Y.C. and Lau, T.W.E. (1997) Universal alignment probabilities and subset selection for universal probabilities, *Journal of Optimization Theory and Applications*, **93**, 455–490.

Kirkpatrick, S., Gelatt, Jr., C.D. and Vecchi, M.P. (1983). Optimization by simulated annealing, *Science*, **220**, 671–680.

Koza, J.R. (1992) Genetic programming: On the programming of computers by means of natural selection, MIT Press, Cambridge, MA.

Laarhoven, P.J.M. van and Aarts, E.H.L. (1987) Simulated annealing: Theory and applications, Kluwer, Dordrecht.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953) Equations of state calculations by fast computing machines. *J. Chem. Phys*., **21**, 1087–1091.

Randelman, R.E. and Grest, G.S. (1986) N-city traveling salesman problem: Optimization by simulated annealing, *Journal of Statistical Physics*, **45**, 885–890.

Ripley, B.D. (1987) Stochastic simulation, Wiley, New York.

Rossky, P.J., Doll, J.D. and Friedman, H.L. (1978) Brownian dynamics as smart Monte Carlo simulation. *Journal of Chemical Physics*, **69**, 4628–4633.

Rubin, D.B. (1988) Using the SIR algorithm to simulate posterior distributions (with discussion), in Bayesian statistics 3 (eds. J.M. Bernardo, M.H. DeGroot, D.V. Lindley and A.F.M. Smith), Oxford University Press, 395–402.

Rust, J. (1993) Using randomization to break the curse of dimensionality, Technical Report JR1, Department of Economics, Yale University.

Schmeiser, B. and Chen, M.-H. (1991) General hit-and-run Monte Carlo sampling for evaluating multidimensional integrals, Technical Report, School of Industrial Engineering, Purdue University.

Smith, A.F.M. and Gelfand, A.E. (1992) Bayesian statistics without tears: A sampling–resampling perspective, *American Statistician*, **46**, 84–88.

O'Sullivan, J.A., Miller, M.I., Srivastava, A. and Snyder D.L. (1993) Tracking using a random sampling algorithm, in: *Proceedings of the 12th IFAC World Congress*, Sydney, Vol. 5, 435–438.

Tierney, L. (1994) Markov chains for exploring posterior distributions. *The Annals of Statistics*, **22**, 1701–1762.

Vidyasagar, M. (1997) Statistical learning theory and its applications to randomized algorithms for robust controller synthesis, *Plenary Lectures and Minicourses*, *European Control Conference* (eds. G. Bastin and M. Gevers), 161–189.