Ministry of Education

And Science of the Russian federation

Ural Federal University

named after the first President of Russia B.N.Yeltsin

Ivan Savin

Andrey Pushkarev

MONTE CARLO METHODS IN ECONOMICS

electronic educational resource UrFU

The book is suited for students of the «Financial economy» and «Theoretical and experimental economy» master programs that are studying «Methods of Monte Carlo in economic research» course.

The aim of this book is to give an overview of what Methods of Monte Carlo are, present their scope of use, and provide useful guidelines for adopting these methods in applied research.

Yekaterinburg,

2015

CONTENTS

INTRODUCTION

The Monte Carlo methods are group of analyzing approaches that are based on the random number generation and usually employs computer algorithms. These methods are widely used in the computational sciences, but in the last decades found their way into social studies.

Monte Carlo methods have been existing for more than a 60 years by now, being one of the first computational approaches used with newly invented digital computers. These methods are usually used in the cases where traditional approaches fail to provide accurate results, whether complex functions are analyzed or markets with a big number of different actors are modeled. With development of computer technologies Monte Carlo methods have found their way into many disciplines. Nowadays these methods are more widespread than ever. However, in some cases now even more effective approaches are proposed. Yet, in many applications Monte Carlo cannot be surpassed in effectiveness and accuracy. For example, it is still dominant in its original application, which is simulating complex interactions in any area where quantitative models are possible.

The main aim of the book is to demonstrate what Monte Carlo methods are and how they can be used to solve problems in mathematics, economics, physics, biology, sociology, and a number of other sciences. This is done by reviewing different computational approaches that use Monte Carlo methods and providing different examples to illustrate how these methods could be used in practical research.

This book extensively uses mathematical language as one of the means of explanation, thus student should have certain level of mathematical skills, including matrix arithmetic and integration. Apart from that, this book provide algorithms for computer programs using MATLAB instruments as illustrations, therefore to understand them programming experience would be helpful, but not crucially needed, since all examples are commented and explained. A ‘computer inclined’ mind is more important in this case. Knowledge in statistics, probability theory, and econometrics would also be beneficent for understanding of this book.

The book is divided in four parts that consist from several sections and exercise section, which allows to revise and use in practice gained information. Each of the parts considers different components of Monte Carlo methods and their implementation. Let us present them in more detail.

First part gives an overview of what Monte Carlo methods are, how they appeared and developed, what tools are used to employ Monte Carlo methods. Apart from that, first part revises concepts of probability and randomness that are required to understand considered approach. It also explains how different random number generators work and their distinctive features. It all concluded with several basic examples of how Monte Carlo methods could be used.

Second part descripts what is sampling, why is it used, and how can it benefit from the Monte Carlo methods. Next, this part presents how can standard distributions be sampled, providing the list of commands for MATLAB and code examples. Then the explanation of more complex procedure of sampling from nonstandard distributions with employing Monte Carlo approach follows.

Third part considers the concept of the heuristic optimization approach, reviewing general idea of the approach compared to the traditional view, then provides explanations for different types of the heuristic optimization algorithms and how they should be constructed and implemented. As an illustrations for the first segment of this part, book presents several applications of the optimization heuristics for the discrete and continuous search spaces.

Last part of the book explains a modern concept of agent-based modelling. Covering common points of its idea, components, why this kind of modelling could be more effective than classical models, and main distinctive features. Moreover, this part presents several examples of the agent-based models, such as urban models, models of opinion dynamics, models of supply chains and industrial networks aiming to cover all main spheres of application of the modelling.

This book uses MATLAB programming software for the algorithm illustrations and creating informative graphs and visualizations of the results. The choice of the programming language is mainly based on simplicity of it both regarding the understanding it and using it to solve problems, since it provides a wide range of mathematical and graphical tools. Moreover, the codes provided in the book are rather simple and could be executed within MATLAB (or it freeware alternative – Octave). However, when solving complex problems, that require implementation of the Monte Carlo methods, researchers might consider using richer programming languages such as Java, Python or C++.

PART 1: INTRODUCTION TO MONTE CARLO METHODS

This part will give an overview of what Monte Carlo methods are, how they appeared, what problems they could solve, and their simplest applications. It also covers how random events or numbers could be employed in scientific assessment and what randomness really is. Different algorithms for generating random numbers and their practical usage in Monte Carlo methods are also presented in this part of the book.

1.1. Monte Carlo methods

As scientific problems become more and more complex and computers become more and more powerful, new methods of assessment and analysing appear. The Monte Carlo methods (MMC) are part of them. It is essentially a technique for analysing phenomena by using computational algorithms relying on generating random numbers. This group of methods received its name from its creators – Stanislaw Ulam and John von Neumann who proposed it while working on solving the neutron diffusion problem at Los Alamos in the 1940s. The method name refers to Monte Carlo Casino in Monaco where Ulams' uncle liked to gamble and represents randomness employed in the method and in the Monte Carlo casinos as it is explained in the article *The Beginning of the Monte Carlo Method* by Metropolis (1987).

Random numbers are frequently used nowadays in a data encryption, for instance, in Bitcoin transactions, credit card payments, TOR and I2P browsing, apart from that, many internet services employ random number generation for a protection of personal data. Randomness is also a key element to all kinds of games, which adds interesting in-game events and interactions, thus more entertainment. Monte Carlo methods are widely used in science, business, mathematics, logistics and many other spheres. These methods allows creating highly reliable algorithms for predicting stochastic processes (that have random elements in them) for example, particles motion, busy seaport logistics, or data transition through the Internet, and many others. The Monte Carlo approach is also useful to get numerical solutions to complicated problems that could not be solved analytically.

As Shonkwiler and Mendivil (2009) mention the method that would later be called Monte Carlo method was first used by Georges-Louis Leclerc, Comte de Buffon in the 1777 as a solution to the problem, which he proposed even earlier, in 1733. His experiment originally involved dropping a needle on a flat ruled surface and determining the probability of the needle crossing one of the lines. Which is also directly related to estimating **π**-value using random events. Let us reconstruct this needle problem as an introduction to the Monte Carlo methods as described in the aforementioned book.

Buffon Needle Problem

Consider a needle of length **L** thrown randomly onto a flat surface ruled by parallel lines each a distance **d** apart. $\mathbf{H_n}$ is a random variable representing the number of times the needle touches or crosses a line in **n** throws, the number of hits.

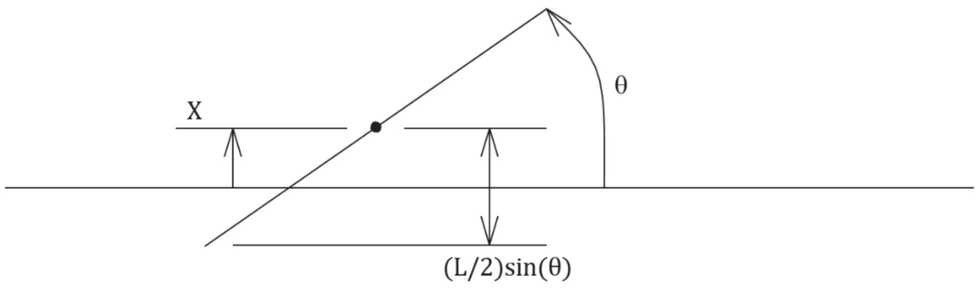


Fig. 1.1. Buffon needle problem scheme

In the Figure 1.1 ***X*** denotes the distance between center of the needle and the nearest line and **θ** – the acute angle between the needle and the line. In this situation $\mathbf{0 \leq X \leq d/2}$ and $\mathbf{0 \leq \theta \leq \pi/2}$. The needle touches or crosses a line if and only if this condition is satisfied:

$$\mathbf{X \leq \frac{L}{2} \sin\theta}\,.$$

If represented graphically, probability of needle hitting the line is equal to the area under the curve $\mathbf{y = \frac{L}{2}\sin\theta}$. So, if $\mathbf{d \geq L}$, then following is correct:

$$\mathbf{Pr(Hit) = \frac{area\ under\ the\ curve}{area\ of\ \left(0,\ \frac{\pi}{2}\right) \times \left(0,\ \frac{d}{2}\right)} = \frac{\int_0^{\frac{\pi}{2}} \frac{L}{2} \sin\theta\, d\theta}{\frac{d}{2} \cdot \frac{\pi}{2}} = \frac{2L}{\pi d}}.$$

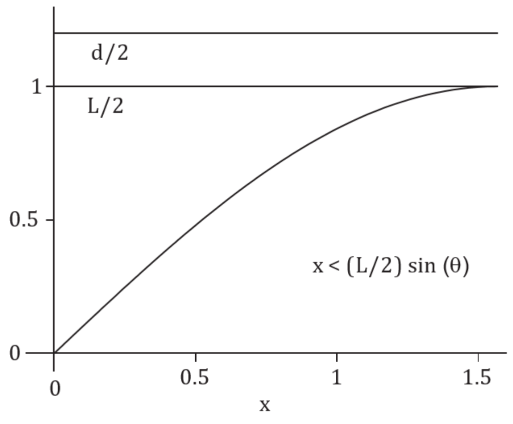


Fig. 1.2. Buffon needle "dartboard"

If, for simplicity, we would assume that distance between lines equals doubled needle length **d = 2L**, then probability of hit is **1/π**.

In the actual experiment the value of **π** is unknown in advance and its estimate could be obtained from the experiment itself. Therefore, Buffon first found a way to assess the well-known **π**-value by using random events. However, the drawback of this "innovative" method is that it requires a huge number of these random events to achieve even modest accuracy. In fact, the more experiments are conducted the less accuracy increases, to be precise it increases in proportion **1/n**, where **n** is number if needle tosses. Practice showed that far more than 500 experiments should be conducted to get viable assessment.

Originally, the simulation have been done manually, using mechanical calculators. With a development of technologies, doing simulations became easier. John von Neumann, who have been strongly connected with a development of digital computer, realized that this technology could be used instead of manual labour. Therefore, he developed the first random number generator, the "middle-square" generator, which have been used to solve neutron flux problem, as we stated earlier. We would overview modern types of random numbers generators in the Section 1.3 of this book.

To illustrate how programming could be used to generate random numbers and to solve mathematical problems with it we are presenting several program codes for the MATLAB (or Octave) with results of running them. Every code has comments after % sign. For further information on commands, symbols, and functions used in the codes refer to the MATLAB or Octave help files. First, let us present and comment the code for the Buffon needle simulation and showing histogram of the results.

Code 1.1. Buffon Needle Simulation + histogram representation (executable in MATLAB/Octave)

```
throws = 100; % 100 trails
repl = 3000; % 3000 replications
for i = 1:repl; % loop for replications
        x = rand(1,throws); % a vector of 10000 pseudorandom numbers in the range [0,1)
        theta = 0.5*pi*rand(1,throws); % 10000 random numbers between 0 and π/2
        hits = x <= 0.5*sin(theta); % vector of hits 0 = miss 1 = hit
        y(i) = sum(hits)/throws; % random variable in the numerator
end;
hist(y) % histogram
recipEst = sum(y)/repl % Estimation for 1/π
vVec = (y-recipEst)*(y-recipEst); % vector of sqr deviations
v = sum(vVec)/(repl-1); % sample variance
stddev = sqrt(v) % sample standard deviation
piEst = 1/recipEst % result of π
```

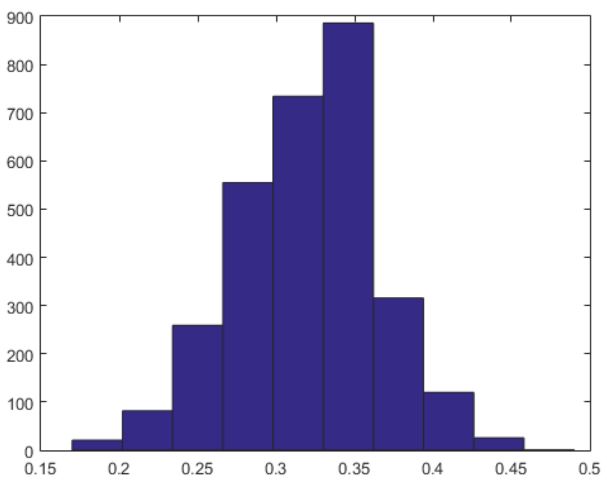


Fig. 1.3. Buffon needle histogram for 1/π

The script above (see Code 1) returns single-number estimation of π based on 3000 replications with 100 trails and assuming that distance between lines equals doubled needle length $\mathbf{d = 2L}$. It also returns a standard deviation of the result and a histogram for $\mathbf{1/\pi}$ (it is essentially a probability of “hit” with chosen parameters), which is presented in the Figure 1.3. The program generates two random numbers for computational purposes, one to play the role of $\mathbf{X}$, $\mathbf{0 \le X < d/2}$, and the other of $\boldsymbol{\theta}$, $\mathbf{0 \le \boldsymbol{\theta} < \boldsymbol{\pi}/2}$. Next it calculates whether $\mathbf{X \le \frac{L}{2} \sin\boldsymbol{\theta}}$ and record a hit if so, than it estimates $\boldsymbol{\pi}$. Using replications as “super-experiments” allows building a histogram with the data computed.

The average of the 3000 super experiments approximates the mean of the distribution and is likewise an approximation of $\mathbf{1/\pi}$. For the run shown in Figure 1.3 the average is $\mathbf{p = 0.3183}$. Histogram is considered a very useful representation of the results of the MC experiment. Moreover, analyzing it allows understanding them better. The definition provided by Shonkwiler and Mendivil (2009) states that histogram is a bar chat that shows how many values fall into established intervals of this values. Depending on the number of values laying within a certain interval, height of the bar of this interval is defined. Essentially, it is called frequency histogram. Dividing frequencies by the total number of values allows estimating relative frequencies. If we use these relative frequencies for a histogram, we would build a density histogram, which depicts probabilistic distribution of experiments values.

After running the script we received following estimates $\mathbf{1/\pi = 0.3183}$, $\mathbf{\pi = 3.1410}$, and a standard deviation $\mathbf{s = 0.0460}$. One important property if the normal distribution is that a sample taken from it falls between two standard deviations on either side of the mean with a probability of **0.954**. If a number of samples were increased than the interval would shrink respectively, in our run with 3000 samples it would be $\frac{\mathbf{2 \cdot 0.0460}}{\mathbf{\sqrt{3000}}} = \mathbf{0.0016}$ on either side of the mean. Therefore, in our case any sample taken would be inside **(0.3151; 0.3215)** interval with a 95 % probability. We also could say that $\boldsymbol{\pi}$ itself with 95 % probability lies within an interval $\mathbf{1/0.3215 < \pi < 1/0.3151}$. In our example, $\boldsymbol{\pi}$ falls between **3.1104** and **3.1735**.

Second way of presenting the results of the experiment is plotting its sample path, which is pair of values – number of the trail and its result. As an illustration, consider a gambling experiment with a gambler and a house. The gambler makes 1$ bets against the house with even winning chances. The gambler starts with 100$, while the house starts with bigger amount of money, for simplicity let us take 2000$ and they play until one of them is bankrupt. The Code 2 simulates this experiment. In this experiment we could find answers for two questions:

– Can house go bankrupt?
– How long the play will last until one of the players is bankrupt?

Code 1.2. Experiment with a gambler and a house (executable in MATLAB/Octave)

```
R = zeros(1,2000000); i = 1; R(i) = 100; % vector of 2,000,000 zeros
while( R(i) > 0 & R(i) < 2100 )
        i = i+1;
        W = (rand < 0.5); % random value of 0 or 1
        W = 2*W - 1; % random value of +/-1
        R(i) = R(i-1)+W; % gamblers new fortune
end
subplot(1,2,1); plot(1:i+1,R(1:i+1)) % plot R against its index
%%% run for investigating playing time
for j = 1:20 % cycle of trials
        i = 1; R(i) = 100;
        while( R(i) > 0 & R(i) < 2100 )
                i = i+1;
                W = 2*(rand < 0.5)-1;
                R(i) = R(i-1)+W;
        end
        T(j) = i; % result of playing time computation
end
subplot(1,2,2); hist(T) % returns histogram of the results
```

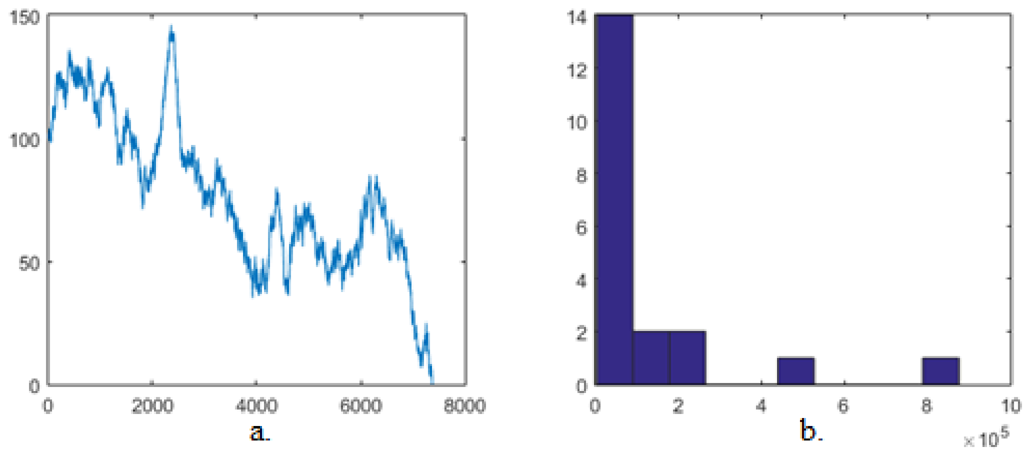


Fig. 1.4. a. Sample path of a gambler’s ruin experiment (fortune vs. gambles). b. Histogram of the number of plays until bankruptcy in 20 trials (gambles until bankruptcy)

Again, in this code we employ the built-in random number generator, which returns numbers in the range **[0, 1)**. Following the example given in in Shonkwiler and Mendivil (2009) we assume that the gambler wins if the generated random number is less than **0.5**, in all other cases he loses.

Figure 1.4a represents sample path of the experiment for the first trial built by the MATLAB tools. As we can see player goes bankrupt in less than 8000 bets in this run. Performing several more runs of the first part of the script shows the situation when house goes bankrupt is highly unlikely. Figure 1.4b represents result of the second part of the code that returns histogram of the playing time based on 20 further runs. It should be mentioned that there is no upper bound to the playing time length, as can be seen from the histogram a run can take a very long time, in our run it is 1400000 plays.

As we discussed basics of Monte Carlo methods, we also need to overview two important concepts to fully understand how Monte Carlo methods could be employed: probability of a random event and generation of random numbers.

1.2. Probability review

Understanding the concept of the probability is very important part of understanding Monte Carlo methods. This section of the book overviews what is probability and how random variables could be employed in estimations.

Consider an experiment with an event **E** that has a set of all possible outcomes **Ω**. For example, in the experiment with tossing a coin two times there are four outcomes in total. On each toss, coin could land either heads **(H)** or tails **(T)** up, therefore **Ω = {(H, H); (H, T); (T, H); (T, T)}**. In this experiment as a random variable **X**, we could use a number of 'heads' outcomes in a trial of the experiments, therefore **X** can be 0, 1, or 2. As all outcomes are equally likely, their probabilities **Pr(X)** are essentially just counting successful outcomes for the corresponding event. **Pr(X = 0) = 1/4**, because only outcome suitable is **(T, T)**, **Pr(X = 1) = 1/2** because **(H, T)** and **(T, H)** contain one 'heads' outcome, **Pr(X = 2) = 1/4**, because only outcome suitable is **(H, H)**.

Let us divide outcomes of **E** in two sets $\mathbf{E_1}$ and $\mathbf{E_2}$ that satisfy two important conditions: $\mathbf{E = E_1 \cup E_2}$ and $\mathbf{E_1 \cap E_2 = \emptyset}$, in other words, these two sets are disjoint. Therefore for two events in general:

$$\mathbf{Pr(E_1 \cup E_2) = Pr(E_1) + Pr(E_2) - Pr(E_1 \cap E_2).}$$

The subtraction is important to avoid counting twice the values that appear in the both sets. For two disjoint events this formula transforms to:

$$\mathbf{Pr(E_1 \cup E_2) = Pr(E_1) + Pr(E_2)} \text{ if } \mathbf{E_1 \cap E_2 = \emptyset}$$

Discrete and Continuous Random Variables

Following the classification of Shonkwiler and Mendivil (2009) all random variables could be divided into two basic groups: discrete or continuous. Discrete random variables are ones that have finitely many or countably infinitely many outcomes (one at a time, obviously).

The coin-tossing experiment above, for example, had four outcomes, therefore the random variable for the number of ‘heads’ there falls into the group of discrete random variables. An example of countably infinite number of outcomes is an experiment where coin is tossed until it lands heads up for the first time. In principle, it could take any number of coin tosses for this to happen, so one cannot evaluate random variable finitely, but it is still discrete.

There are many continuous processes, they include such phenomena as passage of time, movement, distance and many others. Obviously, measurements of such variables are limited to finite estimates, nonetheless they are taken as an example of continuous random variables. Formally, continuous random variables are variables for that $\mathbf{Pr(X = x) = 0}$ where $\mathbf{x}$ is a real value.

Another important probability concept is the cumulative distribution function, which describes probability of random variable $\mathbf{X}$ with some certain probability distribution will be not greater than $\mathbf{x}$. It is presented in following mathematical form:

$$\mathbf{cdf(x) = Pr\left(X \le x\right).}$$

To illustrate what cumulative distribution function is, let us get back to the double coin-tossing experiment and the random variable $\mathbf{X}$ we used above. There are several possible values of $\mathbf{x}$ (see Figure 1.5).

1. If $\mathbf{x < 0}$ then $\mathbf{cdf(x) = 0}$ since there are no outcomes for which $\mathbf{X < 0}$.
2. At $\mathbf{x = 0}$ the **cdf** jumps up to 1/4, since $\mathbf{Pr(X = 0) = 1/4}$.
3. If $\mathbf{0 < x < 1}$ the **cdf** is unchanged and so remains at the value 1/4 over this interval.
4. At $\mathbf{x = 1}$ the **cdf** raises again, this time by 1/2. At this point $\mathbf{cdf(1) = 3/4}$, since the event $\mathbf{X \le 1}$ consists of the several outcomes: **{(T, T); (H, T); (T, H)}**.
5. While $\mathbf{1} < \mathbf{x} < \mathbf{2}$ the **cdf** does not change and has the fixed value of ¾ as in the previous step.

6. Finally, at **x = 2** the **cdf** grows by 1/4 up to 1 and remains there, since all possible outcomes are now included.

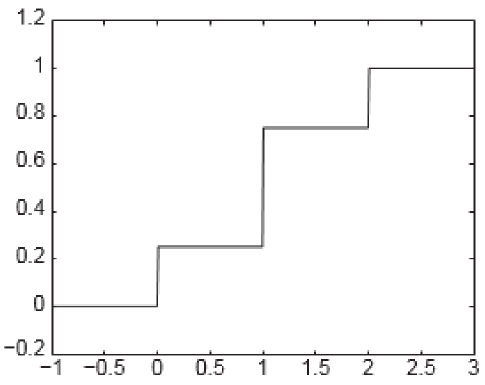


Fig. 1.5. Cdf for the number of heads (x vs. cdf) (Source: Shonkwiler and Mendivil, 2009)

In general, **x** tending to minus infinity means that more and more outcomes of the experiment are omitted and **cdf** must tend to **0**.

$$\lim_{x \to -\infty} \mathrm{cdf}(x) = 0$$

On the other hand if **x** tends to plus infinity, than more and more outcomes are included and **cdf** tends to 1.

$$\lim_{x \to \infty} \mathrm{cdf}(x) = 1$$

So it is also true that **cdf** increases or at least stays the same as **x** increases, because more and more outcomes satisfy the condition of $X \le x$. In other words, $\mathrm{cdf}(x_1) \le \mathrm{cdf}(x_2)$ is true when $x_1 \le x_2$. If a random variable is discrete, its cdf will jump at certain discrete points corresponding to the values of **x** for which one or more outcomes satisfy **X = x**. Otherwise **cdf** stays constant. As an illustration, Figure 1.6 shows **cdf** for rolling a pair of dice. Obviously, the same would not be true for continuous random variables, where **cdf** is a smooth line.

As we see, random variables and probability have their own distinctive features that are important for using them in computational purposes. However, before using, random variables need to be obtained or generated somehow. The next section looks into this.

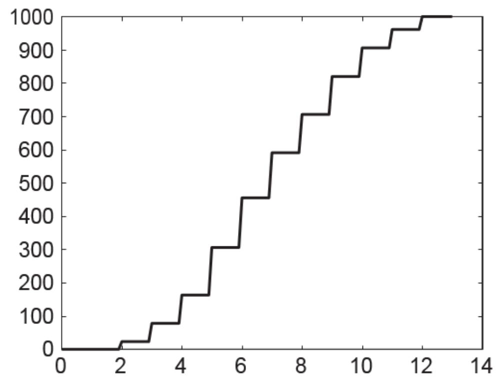


Fig. 1.6. Cdf for rolling a pair of dice. (x vs. cdf) (Source: Shonkwiler and Mendivil, 2009)

1.3. Random Number Generation

In this section the concept of the random number generation is overviewed. There are two main methods of generating random numbers. First method relies on the measurements of certain physical phenomenon, that is thought to be random, for example thermal noise, atmospheric noise, or any quantum phenomena. In purest sense, only this method could give “truly” random numbers. This method is really hard to implement practically, since it is rather slow and requires special equipment, bias correction Second method relies on using certain computational algorithms. It is mentioned in Shonkwiler and Mendivil (2009) that since computers and these algorithms are deterministic, numbers received through this method are not “truly” random. Hypothetically, if initial data and the algorithm are known, then the number could be computed with certain ease. However, these numbers are generally appear to be random and must pass tests to ensure their randomness. Random numbers generated with this method are called “pseudorandom”. This book focuses on the second method, since it is easier to implement and easier to use it for Monte Carlo simulations.

There are several main requirements for a random number generator (RNG). For example these five are listed in the book *Explorations in Monte Carlo Methods*:

1. Fast speed. Programs that employ random number generators usually require millions of random numbers in short time. So generator should be able to compute random numbers fast.

2. Repeatability. This requirement is needed for debugging purposes. If algorithm cannot be repeated than it is impossible to improve or correct it.
3. Analyzable. To ensure the distributional properties of the values received results of the generation must be possible to analyze.
4. Long period. Eventually, all algorithmic RNGs repeat themselves after the certain point of generation, so the longer is the period before this point – the better.
5. Apparent randomness. As it was mentioned earlier algorithms does not allow generating truly random numbers, so at least these numbers should appear random to the intended sphere of usage.

At first, it seems that to generate a random number one just needs to use a complicated algorithm with many steps and transformations. This is not true. The work *Seminumerical Algorithms* by Donald Knuth gives perfect explanation why is it so. Author describes his early attempt in constructing a random number generator. It was a complicated 13-step algorithm that acts on 10-digit decimal numbers changing the current number **X** into a new number in the sequence via changing it in the "randomized" way.

In the first run this algorithm almost instantly converged to the value 6065038420 (which converged to itself). He decided to run it one more time with a different starting value, and it converged to the cycle with length 3178. This example clearly demonstrates that complexity could not substitute randomness and seemingly complex algorithm may have simple behavior.

Common form of the random number generator would be

Output = f(varibles, parameters)

For some function **f**. For each request program executes the algorithm to calculate this function using variables and parameters to produce certain sample of "random" process. Parameters of the functions are fixed values, while variables change from sample to sample. Variables could be any external values, like time, or internal values that change every iteration. These internally stored variables could also be called seeds. Generator could have one such value or several of them. Let us first look into single seed generators.

Generators with one stored value

If seeds values have been taken from some external process at any time during the execution of generating algorithm, then it would be impossible to generate the exact same samples once again, thou it is impossible to debug program in this case. It is also would be hard to ensure distributional properties. Therefore, usually the seed values are internally generated by the RNG itself, except for the first one, which, for example, could be taken from the computer clock and saved for debugging purposes.

After initial seeding all other seeds are computed from the function **f** itself, simultaneously calculating output samples. The process could be simplified if output sample is used as the next seed. Doing this would not cause any biased results, since algorithm adjusts samples anyways.

As it was mentioned before, sequences of outputs in any random number generator tend to repeat itself after some point. In this situation it happens after one seed repeats, then a generator will repeat or cycle. With a single-seed RNG period of unique output sequence is no longer than a number of different possible seeds.

Considering everything mentioned above, random number generator is generally similar to the book with a long list of numbers, initial variable selects a starting point for reading the numbers and after that order of numbers follows predetermined order. If different seed is used then it starts reading from a different place. If the same seed is used then sequence of outputs is the same. When algorithm reaches the end of the list it starts from a random place of the list.

This type of generators is fast and relatively easy to check for bugs, but on the other hand, period of this type of random number generators is rather limited. Fortunately, other types do not have this drawback.

Middle-Square and Other Middle-Digit Techniques

This type of random number generators is based on the idea that after first several digits in the calculation of most mathematical functions, like **sin()** or **log()**, following digits are then virtually random. Von Neumann while working in Los Alamos has first used this type of RNG, he initially used $\mathbf{x^2}$ as a key function and then assumed, that middle digits of the variable are “random”, which gave the name for the type. This choice of function was determined by the fact that **sin()** or **log()** are relatively time-consuming to compute even with modern technologies as it is stated by Shonkwiler and Mendivil (2009). Code 3 demonstrates this.

Code 1.3. Comparison of middle-digit RNG and built-in RNG (executable in MATLAB/Octave)

```
x = 0.5;
tic
for i = 1:1000000
        x = 100000*sin(x);
        w = floor(x); % take first 5 digits
        x = x-w; % digits after the 5th
end
toc % 0.0369 Computing sin(.) is more time-consuming
% compare with Matlab’s built-in generator
tic
x = rand(1,1000000);
toc % 0.0139
```

Obviously, exact values of the execution time of the script could vary depending on computational power of machine used, but built-in RNG is always from three to thirty times faster than the middle-square technique, while both return similar pseudorandom results.

The main drawback of this technique is short period and rapid cycling for most initial variables. Apart from that, as shown above, middle-digit generators are slow for many programs. Therefore, this technique had limited practical usage. This stimulated search for more suitable random number generators.

Linear Congruential Random Number Generators

The linear congruential random number generators are one of the most widely used generators nowadays, while they depend on simple recurrence to compute following outcome in the sequence from the previous one. For example, many software systems use linear congruential random number generator (Mersenne Twister, to be precise) as its default RNG, such as MATLAB, R, Python, PHP, Ruby and many others, which is usually presented in their documentation.

The name of the generator comes from modular arithmetic, where **a** could be called congruent to **b** modulo **m** if $\mathbf{m \mid (a-b)}$. Formally defined, the linear congruential random number generator (LCRNG) is function **f** depending on three fixed parameters **a**, **c**, **m**:

$$\mathbf{f(x) = (ax + c) \bmod m,} \text{ where } \mathbf{x} \text{ is an integer.}$$

Let us look into how this random number generator works in practice. For example, take $\mathbf{a = 5}$, $\mathbf{c = 1}$, $\mathbf{m = 8}$ and use $\mathbf{x = 0}$ as a seed. As we use it, output will be $\mathbf{(5 \cdot 0 \ + 1) mod = 1}$, then 1 becomes next seed. If we compute the function repeatedly, we would receive the following sequence of samples before output begins to cycle:

$$\mathbf{R = (0, 1, 6, 7, 4, 5, 2, 3, 0).}$$

Definitely, this output could not be called random by any means, since it is generated by simple formula, but with different values of **m** and **a** results could be much more random-like. As it is seen, the output the period length (number of different return values) in this generator is equal to the value of **m**, therefore in any real generator **m** is taken as large as possible, for example Mersenne Twister, that was mentioned before, has period length of $\mathbf{2^{19937} - 1}$ which allows more than 10^{6000} unique outputs until it becomes repetitive as stated in the original paper by Matsumoto and Nishimura (1998). For the reasons of optimization **m** could be taken as $\mathbf{2^k}$ where **k** equals the word length (number of operations CPU can process in one go, usually 32 or 64 for modern devices) of the computer. Shonkwiler and Mendivil (2009) explain that it is because the modulo operation would take no extra time and happen automatically, since there is no room for the higher-order bits. In this case, LCRNG would need only addition and multiplication to get one result, which is fast and saves computational power. For floating point numbers division is also required.

If $\mathbf{c = 0}$ then the addition operation is also avoided, and the sample sequence represented by the following equation:

$$\mathbf{x_{n+1} = ax_n \bmod m}.$$

In this case, maximal period cannot equal **m**, since $\mathbf{x_n = 0}$ turns all subsequent values into 0 also. The period $\mathbf{m - 1}$ still could be achieved in the situation when m is a prime number, however the modulo operation would no longer be automatic, while **m** is not a power of 2.

In general, if $\mathbf{c = 0}$, $\mathbf{m}$ is not prime, and if $\mathbf{x_n}$ is a multiple of a certain divisor $\mathbf{d}$ of $\mathbf{m}$ then all subsequent values of $\mathbf{x}$ will also be multiples of $\mathbf{d}$. We will have $\mathbf{x_n = pd}$ for some integer $\mathbf{p}$, and since $\mathbf{d}$ divides $\mathbf{m}$, then $\mathbf{m = qd}$ for some integer $\mathbf{q}$. So $\mathbf{x_{n+1} = ax_n \bmod m} = \mathbf{apd \bmod m}$ means $\mathbf{x_{n+1} - apd = rm}$ for some integer $\mathbf{r}$. But then $\mathbf{x_{n+1} = (ap + rq)d}$. Therefore when $\mathbf{c = 0}$, the $\mathbf{x}$'s should ideally run through all the integers relatively prime to $\mathbf{m}$, if $\mathbf{a}$ is chosen correctly. Note that, if $\mathbf{c = 0}$ and $\mathbf{1}$ is in the sequence, then it would consist of consequent powers of $\mathbf{a}$.

As it was mentioned earlier many built-in PRNGs are linear congruential. However, now more and more software packages adopt better generators that are now available.

Linear Congruential Random Number Generators have several problems that should be mentioned. First, the period is limited to $\mathbf{m}$ at most. Further on, if samples truly random the repeats were possible, but for LCRNG it is not possible. Also analyzing the sequences allows making certain predictions about next output values, what is highly undesirable for a good random number generator. The main problem of LCRNGs is their simplicity. To avoid that multivariate generators of the congruential type and generators using a different mathematical basis potentially could be used.

1.4. Several Applications of Monte Carlo Method

As it was mentioned earlier, Monte Carlo methods could be used in many spheres of mathematics, science, technology, and entertainment. This section overviews several possible applications of Monte Carlo methods for solving mathematical or statistical problems.

Hit-or-miss

One of the earliest uses of Monte Carlo method is evaluation of the integrals, since some of them could be hard to solve in "traditional" manner. To solve it hit-or-miss logics could be employed; by doing a number of "throws" we could determine the desired area, the same logics are used in the Buffons Needle Simulation.

First, let us consider general situation. Let $\boldsymbol{\theta} = \int_a^b \mathbf{f(x)dx}$, which is the area under the function f between $\mathbf{x = a}$ and $\mathbf{x = b}$, assuming that $\mathbf{f(x) \geq 0}$ on the interval $\mathbf{[a, b]}$.

Let $\mathbf{M} \geq \max\limits_{a \leq x \leq b} \mathbf{f(x)}$; then target area lies within the rectangle $\mathbf{[a, b] \times [0, M]}$. If we start selecting points within this rectangle at random the probability of the point falling into target area is equal to ratio of that area to the area of the rectangle, which is $\frac{\theta}{M(b-a)}$. Considering **n** to be the number of throws and **h** – number of hits, the following is correct:

$$\frac{h}{n} \approx \frac{\theta}{M(b-a)}$$

or

$$\theta \approx \left(\frac{h}{n}\right) M(b-a).$$

So since dimensions of the rectangle, **n**, and **h** are known, target area could be easily calculated by simple operations, even thou target function may be complicated.

Monte Carlo methods are highly effective for solving integrals in the multiple integration over complicated regions in high-dimensional space essentially with the same logics. The target region is “enclosed” within certain easy-to-calculate area or space, and then points inside it are selected randomly to see if they hit the target region. The hit-“throw” ratio would be an evaluation of desired volume. To illustrate that let us estimate π value once again using the tools of the MATLAB.

Following the procedure proposed by Shonkwiler and Mendivil (2009) we take the curve forming is the first-quadrant portion of a unit circle $\mathbf{y = \sqrt{1 - x^2}, 0 \leq x \leq 1}$. Therefore, the target area would be $\boldsymbol{\pi/4}$ that is contained in the square $\mathbf{[0, 1] \times [0, 1]}$, which has area equal to 1. Following hit-or-miss logics, an estimate of π is the following:

$$\pi \approx 4\frac{h}{n}.$$

Let us present the MATLAB code that allows performing this simulation and evaluating how estimate of $\boldsymbol{\pi}$ converges with more trails of simulation conducted.

First part of the code returns two values – number of hits and estimation of π. In this run they were 7815 and 3.1260 respectively, which is rather accurate even for such relatively small amount of trials. Obviously, increasing number of “tries” would have positive effect on the estimate. Let us demonstrate that.

The second part allows seeing how the estimated value of π improves as number of trials increases. In total 20 cycles with 1048576 trials have been made. At first 5 cycles π value estimation is completely inaccurate and vary a lot, but after seventh cycle all values are close to 3.1 at the last cycles converging around 3.14, which is rather accurate (see Figure 1.7), with that absolute error decays in each new cycle at a rate approximately $\mathbf{1/\sqrt{n}}$.

Code 1.4. Estimation of π (executable in MATLAB/Octave)

```
nTrials = 10000;
x = rand(1,nTrials); % generating x-coordinate
y = rand(1,nTrials); % generating y-coordinate
hits = sum(x.^2 + y.^2 < 1) % 7815 hits
piEst = 4*hits/nTrials % 3.1260
% pause
nPower = 20;% used to define the period of RNG
for i = 1:nPower % 2^20=1048576
        x = rand(1,2^i);
        y = rand(1,2^i);
        piEst(i) = 4*sum(x.^2 + y.^2 < 1)/2^i
end
plot(piEst) % returns plot of estimated π convergence
```

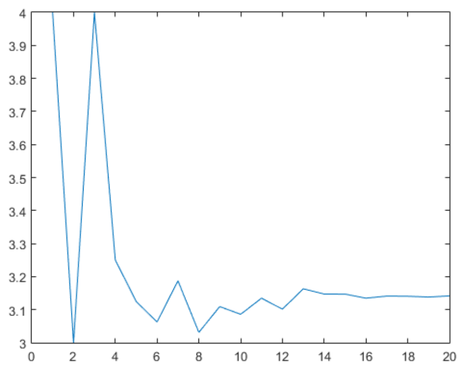


Fig. 1.7. Convergence of estimate of π (Estimated π vs Number of Trials)

As we can see, Monte Carlo methods is rather effective in this situation and does not require solving computing of the integrals.

Coupon Collecting

Another viable example is a situation of coupon collecting. Many retail companies use promotion campaigns where buyers have to collect all coupons with a letters of a certain word, while each letter is provided with the product. After collecting all letters they receive some kind of reward. Therefore, using Monte Carlo methods it is possible to find how many purchases one should make to collect the word.

Let us say customer have to collect the word MONTECARLO, which has 10 letters. To make the example simpler we would assume that all letters have equal chances to be acquired (which is usually not true for the real campaigns, where one or two letters are made to be extremely rare). To calculate the number of purchases we have to check if every purchase has the letter needed. The following MATLAB program could compute that and return the histogram of the results (see Figure 1.8).

Code 1.5. Solution to Coupon Collecting problem

```
nLetters = 10; % MONTECARLO
nTrials = 10000;
for i = 1:nTrials
        success = 0;
        nTries(i) = 0;
        for j = 1:nLetters
                MONTECARLO(j) = 0; % reset letter not achieved
        end
        while success == 0
        nTries(i) = nTries(i)+1; % inc. count
        buy = 1+floor(nLetters*rand); % letter obtained
        MONTECARLO(buy) = 1;
        if sum(MONTECARLO) == nLetters % all letters obtained
                success = 1;
        end
        end
end
hist(nTries)
```

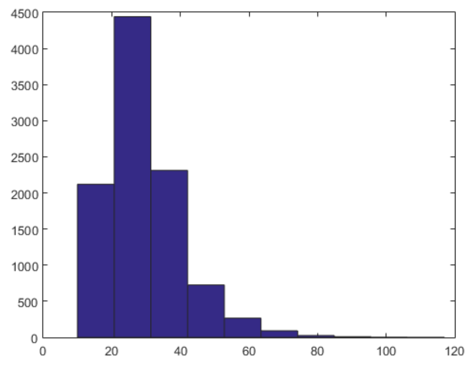


Fig. 1.8. Histogram of number of tries needed to get all coupons

The results show that even though all letters occur equally frequent it usually takes more than 20 purchases or could even take more than 100 purchases. It is interesting that graph peaks at (20; 30) bin, decreasing exponentially after the peak.

This problem could be solved analytically, but in this case, probability estimation could suggest the way of “traditional” solution. Which is also a good reason to use Monte Carlo method sometimes.

Exercises

1. Perform a large number of trials of Buffon’s needle problem and estimate the value of π. Plot the error as a function of the number of trials.
2. Perform 2000000 trials of Buffon’s needle problem for **1/π**: (a) by partitioning the trials into 10000 ‘super experiments’ of 200 trials each; (b) by partitioning the trials into 1000 ’super experiments’ of 2000 trials each; (c) by partitioning the trials into 100 ’super experiments’ of 20000 trials each. Compare and comment the results.
3. Simulate 1000 rolls of a pair of dice and histogram the results as (a) the sum of the dice and (b) the individual pairs, e.g. (1, 1) and (1, 2) and … through (6, 6). (c) Do the part (a) for the roll of 6 dice; histogram 4000 rolls.
4. Modify the gambler’s ruin problem with various number of restarts. How does this change the average duration of the game? How does this bias the sample variance?
5. Make up an LCRNG using **m = 216**; sample it 100 times and find the length of the runs-up and the runs-down. For example, in the sequence 1, 6, 31, 156, 781, 3906, 847, 12, 67, 319, 1582 the first six numbers form a run-up, then comes a three-term run-down followed by a four-term run-up. Hence, the runs-up total 10 in length and the runs-down total 3 in length. What should one expect from truly random numbers?

PART 2: DATA SAMPLING

In statistics, generally, data sampling is an analysis used to select and manipulate representative data set in order to find patterns and trends in it. Sampling allows researchers to work with a manageable amount of data, thus building and running models more quickly. Nowadays, as we mentioned before, researchers use more and more probabilistic models, since problems are often are complicated for "traditional" analytic approaches. Moreover, probabilistic models allow researchers to abandon unrealistic assumptions of normality and independence that are essential for the analytical methods.

Monte Carlo methods sampling is used to make predictions about models' behavior under defined set of circumstances or, for instance, finding appropriate values for the parameters of the model with a given set of experimental data. Common algorithm here is to divide sampling of the complex distributions problem into several sub problems that are simpler. This part of the book overviews two types of sampling for different types of distributions: inverse transformation and rejection sampling. Both these are good for the situations with single-valued outcomes; however, there are more complicated approaches (e.g. Markov chain Monte Carlo approach) that could be used with multivariate distributions.

2.1. Sampling from standard distributions

There several types of distributions, which are so widely used, that they are considered standard and usually supported in the popular software environments. MATLAB and Octave support many types of probability distributions, therefore it is easy to calculate such useful values like the probability density, cumulative density, and to sample random values from these distributions. Full list of supported distributions could be found in the MATLAB documentation on the 'Supported Distributions' page, but the most popular of them are presented in the Table 2.1. As one can see, MATLAB could simulate such popular distributions as normal, exponential, Poisson. The table also presents commands for probability density function, cumulative distribution function, and random number generation that could be found in MATLAB documentation.

Table 2.1

Examples of Matlab functions for different types of distributions

Distribution	PDF	CDF	RNG
Normal	normpdf	normcdf	norm
Uniform (continuous)	unifpdf	unifcdf	unifrnd
Beta	betapdf	betacdf	betarnd
Exponential	exppdf	expcdf	exprnd
Uniform (discrete)	unidpdf	unidcdf	unidrnd
Binomial	binopdf	binocdf	binornd
Multinomial	mnpdf	-	mnrnd
Poisson	poisspdf	poisscdf	poissrnd

To illustrate how these functions could be used let us visualize normal distribution where $\mu = 100$ and $\sigma = 15$, which could for example represent IQ coefficient variation in a certain population. Code 2.1 shows how probability density and cumulative density could be displayed along with histogramming the distribution. As, we mentioned earlier, PDF and CDF are widely used for analyzing probability. The output is shown in the Figure 2.1.

Code 2.1. PDF, CDF, and Histogram for normal distribution (executable in MATLAB/Octave)

```
mu = 100; % the mean
sigma = 15; % the standard deviation
xmin = 70; % minimum x value for pdf and cdf plot
xmax = 130; % maximum x value for pdf and cdf plot
n = 100; % number of points on pdf and cdf plot
k = 10000; % number of random draws for histogram
x = linspace( xmin , xmax , n ); % set of values ranging from xmin to xmax
p = normpdf( x , mu , sigma ); % calculate the pdf
c = normcdf( x , mu , sigma ); % calculate the cdf
subplot( 1,3,1 );
plot( x , p);
xlabel( 'x' ); ylabel( 'pdf' );
title( 'Probability Density Function' );
% pause
subplot( 1,3,2 );
plot( x , c );
xlabel( 'x' ); ylabel( 'cdf' );
title( 'Cumulative Density Function' );
% draw k random numbers from a N( mu , sigma ) distribution
y = normrnd( mu , sigma , k , 1 );
% pause
subplot( 1,3,3 );
hist( y , 20 );
xlabel( 'x' ); ylabel( 'frequency' );
title( 'Histogram of random values' );
```

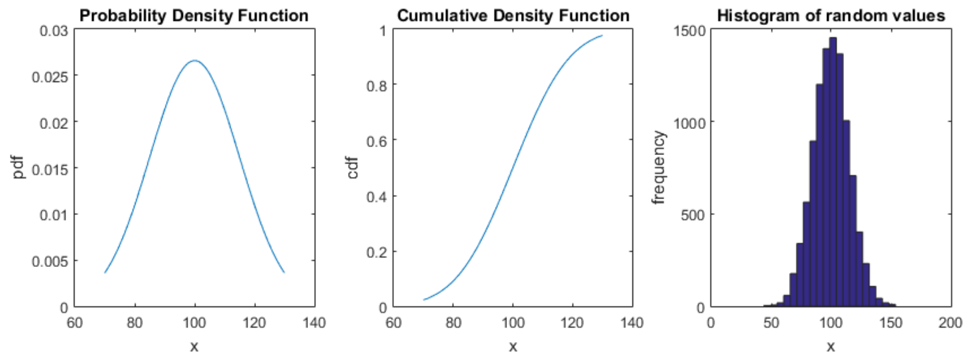


Fig. 2.1. Normal distribution with μ = 100 and σ = 15

Similarly, Figure 2.2 demonstrates in bar graphs density functions and histogram for the binominal distribution, which is common in the situations where researcher counts number of successes from the number of total trials. We overviewed several situations like this in the previous part of the book. The following figure describes a situation when there are 10 trials with 0.7 probability of success.

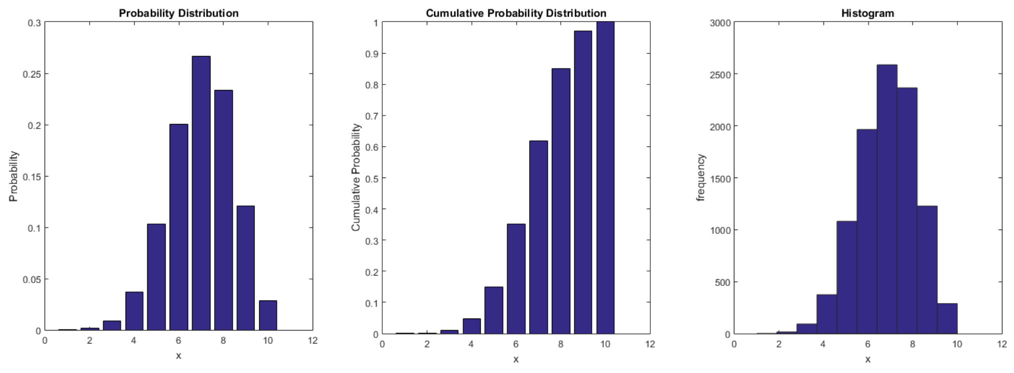


Fig. 2.2: Binomial distribution with N = 10 and θ = 0.7

So, as we showed, MATLAB/Octave provides with a range of instruments for analysing standard distributions, which allows to easily employ them in applied research in different areas of science.

2.2. Sampling from non-standard distributions

In modelling situations, there is a frequent need to use non-standard distributions, while researchers could propose new processes, thus new types of distributions. They could be completely new or could consist from the several standard distributions, but in any way they are not supported by MATLAB. In this case, computational problems often rely on sampling already known distributions. The random values taken from these distributions are then transformed of compared to the target distribution. Some of such techniques are even used by MATLAB to sample from some standard distributions, such as normal or exponential.

Inverse transform sampling with discrete variables

Inverse transform method is a sampling method for generating random numbers from any probability distribution that relies on inverse of its cumulative distribution function. It samples uniformly distributed random numbers between 0 and 1 and transforms them using the inverse cumulative distribution function. This method should be considered rather simple and almost universal, since the sampling is based on transformed uniform deviates. For example, MATLAB implements some of its random number generators through this procedure as Steyvers (2011) states.

Let us give an example of using this approach. The simplest way to illustrate this approach is using a discrete distribution where probability of each individual outcome is known. In this case, inverse transform sampling appears as a simple table lookup. We use data on how well humans can produce uniform random numbers (e.g. Treisman and Faulkner, 1987). In these series of experiments people were asked to produce a large amount of random digits and researchers counted frequency of each random digit. Obviously, humans do not always produce universal distributions. Table 2.2 shows some results from one subject of these experiments. **X** represents the generated digit; **Pr(X)** and **Cdf(X)** represent probability mass and cumulative probabilities respectively.

Table 2.2

Probability of digits observed in human random digit generation experiment

X	**Pr(*X*)**	***Cdf(x)***
0	0.000	0.000
1	0.100	0.100
2	0.090	0.190
3	0.095	0.285
4	0.200	0.485
5	0.175	0.660
6	0.190	0.850
7	0.050	0.900
8	0.100	1.000
9	0.000	1.000

As Table 2.2 states, digits 0 and 9 have not been generated at all, some digits were rather frequent (e.g. 4 and 6), and some are underrepresented. This clearly illustrates that human brain is not well suited for producing uniformly distributed random numbers.

Let us demonstrate the algorithm that could mimic this process, producing digits with a same probabilities shown in the Table 2.2. Therefore program would produce 0 with 0 probability, 1 with 0.1 probability, 2 with 0.09 probability, and so on. The Code 2.2 below shows how it could be done and presented (see Figure 2.3) using standard functions of MATLAB.

Code 2.2. Simulating sampling of random digits.

```
clear all; clc;
theta = [0.000;0.100;0.090;0.095;0.200;0.175;0.190;0.050;0.100;0.000 ]
K = 10000; y = rand(1,K); % K random values
Digitset = 0:9;
for i = 1:10
        Theta(i) = sum(theta(1:i)); % Create cdf
end
for i = 1:9
        Y(i) = sum(y > Theta(i) & y <= Theta(i+1))
end
counts = bar(Y); % Histogram of the simulated draws
xlim( [ -0.5 9.5 ] ); xlabel( 'Digit' ); ylabel( 'Frequency' );
title( 'Distribution of simulated draws of human digit generator' );
```

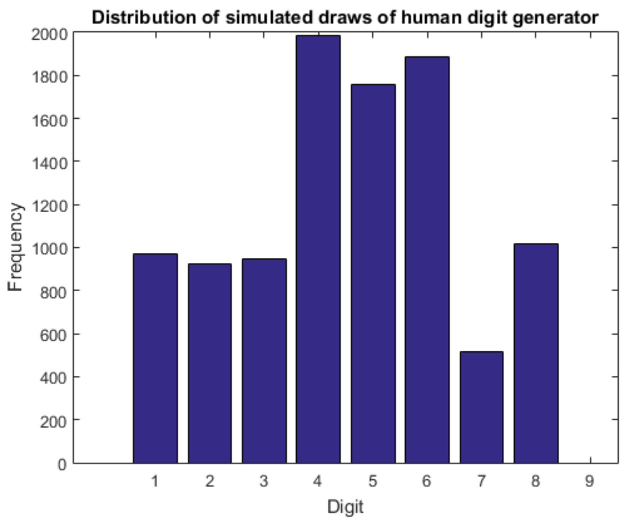


Fig. 2.3. Histogram of the generated results

Histogram obviously states that this program could simulate this exact distributions. But let us look into how implement this algorithm, without using built-in functions, but using the inverse transform method instead. First, cumulative probability distribution should be calculated to know how the observed probability of an outcome is equal to or smaller than some particular value. Therefore, we need to calculate

$$\mathbf{cdf}(\mathbf{X} = \mathbf{x}) = \mathbf{Pr}(\mathbf{X} \leq \mathbf{x}).$$

For discrete distributions, this can be done using simple summation. Result of this operation are shown in the last column of the Table 2.2. Using reverse transform algorithm we need to sample uniform random deviates and compare each random value against cumulative probabilities in the table. The first outcome for which the random deviate is smaller than (or is equal to) the associated cumulative probability corresponds to the sampled outcome. For example, a uniform random deviate of $\mathbf{U = 0.8}$ leads to a sampled outcome $\mathbf{X = 6}$. This algorithm of repeatedly sampling uniform deviates and comparing them to the cumulative distribution allows to form the basis for the inverse transform method for discrete variables. Note that an inverse function is applied in this situation, since an inverse table lookup is conducted.

Inverse transform sampling with continuous variables

As it was mentioned earlier, distributions could also be continuous and inverse transform sampling fortunately could be applied to this type of distributions. Generally, the idea is similar to the situation of discreet variables. One needs to draw uniform random deviates and apply the inverse function of the cumulative distribution applied to the random deviate. Let **cdf(X)** represent the cumulative density function of the target variable **X** and $\mathbf{cdf^{-1}(X)}$ the inverse of the function, assuming this inverse could be calculated. This could be done through repeatedly drawing random variable **U** from the uniform distribution between 0 and 1, and then setting $\mathbf{X = cdf^{-1}(U)}$.

We are going to use the following example as an illustration. Suppose we need to sample random numbers from the exponential distribution. For $\boldsymbol{\lambda > 0}$, the **cdf** would be $\mathbf{cdf(x|\lambda) = 1 - exp(-\frac{x}{\lambda})}$. The inverse of this function would be $\mathbf{cdf^{-1}(u|\lambda) = -log(1-u)\lambda}$. Therefore we use following sample algorithm:

1. Draw $\mathbf{U \sim Uniform(0, 1)}$.
2. Set $\mathbf{X = -log(1 - u)\lambda}$.
3. Repeat.

As one could see, the process is almost the same as for discreet distribution, which was overviewed in detail previously. Another important point is that aforementioned algorithm could be applied to the various situations.

Rejection sampling

The downside of the inverse transform method is that it could not be used in every situation, because inverse function is not always possible to calculate. As an alternative, rejection sampling (also called accept-reject algorithm) or Markov chain Monte Carlo approaches could be used. This section overviews the first option. The main advantage of the rejection sampling is that all samples obtained during sampling could be used straight away as samples from the target distribution, without any "preparation" process.

Let us give the following illustration. Suppose we draw points uniformly from a circle with a centre at **(0, 0)** and **R = 1**. It could be difficult to draw uniformly from a circle area; however, rejection sampling could be applied in this situation. We first draw **(x, y)** values from the square surrounding the target circle, and rejecting any samples that lie outside the circle. In other words, we reject all values for which $\mathbf{x^2 + y^2 > 1}$. This process is illustrated in the Figure 2.4 below. To conclude, we used rather simple and well-known distribution as a basis for sampling from a more complicated distribution.

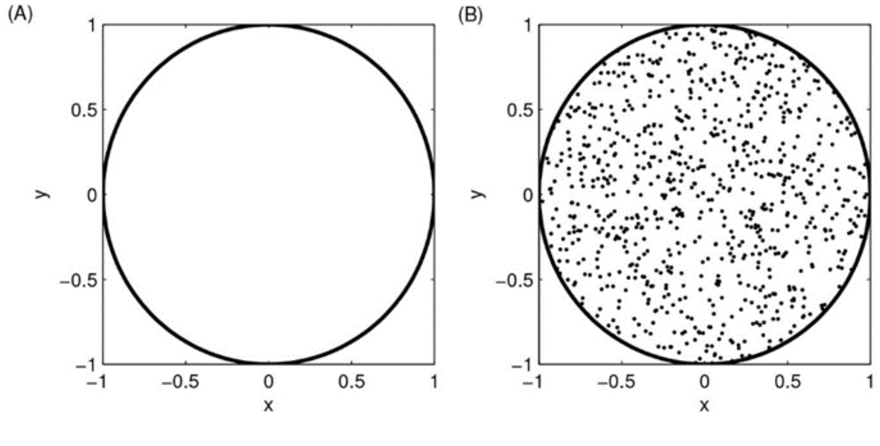


Fig. 2.4. Uniform sampling from a circle using rejection sampling (Source: Steyvers, 2011)

Thus, rejection sampling allows generating observations from a difficult to sample distribution where the probability of a particular sample could be evaluated. In other words, suppose we have a certain distribution **p(θ)** and direct sampling from this distribution is difficult, but the probability density or mass for a particular value of **x** could be evaluated. Then the researcher needs to choose the proposal distribution. The proposal distribution should be a simple distribution **q(θ)** that one can directly sample from. Then the researcher evaluates the probability of the drawn samples under both distributions he or she is using and rejects samples that are unlikely to be under the target distribution relative to the proposal distribution.

Such procedure is shown in the Figure 2.5 below. First we choose constant **c** such that **cq(θ) ≥ p(θ)** for each possible sample **θ**. This function **cq(x)**, which represents the proposal function **q(x)** multiplied by constant **c**, will always lie "above" the target distribution. Finding the appropriate constant might be non-trivial, but usually done through some calculus. After choosing the proposal function and the constant, we draw a value **u** from a distribution that lies inside **[0, cq(θ)]**. The proposal is rejected if **u > p(θ)** and accepted otherwise. If the proposal is accepted, the sampled value **θ** is drawn from the target distribution **p(θ)**.

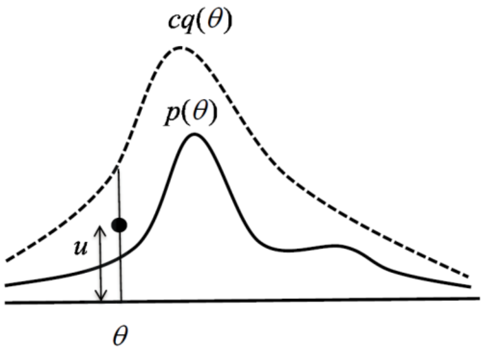


Fig. 2.5. Illustration of rejection sampling technique (Source: Steyvers, 2011)

Let us summarize the procedure by pointing out steps that are taken in this technique:

1. Choose a distribution **q(θ)**. This distribution must be easy to sample from.
2. Find a constant c such that **cq(θ) ≥ p(θ)** for all **θ**.
3. Sample a proposal **θ** from proposal distribution **q(θ)**.
4. Sample a uniform deviate **u** from the interval **[0, cq(θ)]**.
5. If **u > p(θ)** the proposal is rejected, otherwise it is accepted.
6. Steps 3, 4, and 5 are repeated until desired number of samples is reached; each accepted sample is a draw from **p(θ)**.

As pointed out by Steyvers (2011). The efficiency of this procedure heavily relies on the choice of the proposal distribution. It should be chosen so, that as many as possible samples are accepted. If the proposal distribution is highly dissimilar to the target function, then many samples would be rejected, slowing the procedure down.

Let us present the code using rejection sampling technique (see Code 2.3). Suppose we want to sample from **Beta(2,1)**, for $\mathbf{0 \leq x \leq 1}$. The code uses the algorithm discussed above and as a result returns the histogram following the target distribution $\mathbf{f(x) = 2x}$ for the interval $\mathbf{0 \leq x \leq 1}$ (see Figure 2.6).

Code 2.3. Rejection sampling technique (executable in MATLAB/Octave)

```
j = 1;
while i < 1000 % 1000 draws
        y = rand; % take Y from the uniform distribution [0, 1]
        u = rand; % take U from the uniform distribution [0, 1]
        if u <= y % accept value only if U < 2Y/2
                x(j) = y;
                j = j + 1;
                i = i + 1; % draw must be successful
        end
end
hist(x);
```

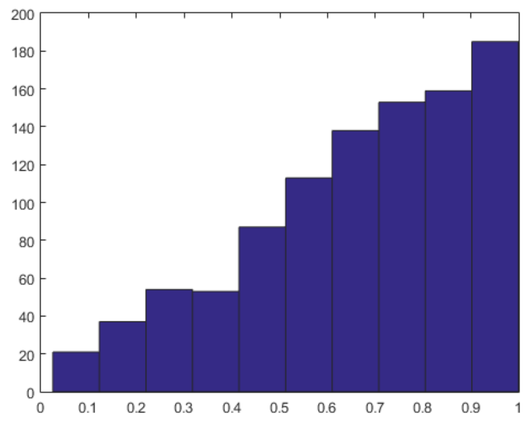


Fig. 2.6. The resulting histogram of p(x) = 2x

The value **c** is chosen to be 2, since in this example $\mathbf{max}\left(\frac{\mathbf{p(x)}}{\mathbf{q(x)}}\right) = \mathbf{2}$. Probably, for this exercise, a simple uniform proposal distribution is not optimal, but it is used, since it is easy to implement in MATLAB. Obviously, increasing number of draws would increase the precision of the method.

This method could also be extended to suit discrete distributions. The difference compared to the continuous case, which we reviewed, is that the proposal distribution **q(x)** must also be discrete distribution.

Further on, for high dimensional distributions Markov chain Monte Carlo method could be used, which has a similar approach. Essentially, it is a computational approach that replaces analytic integration by summation over samples generated from iterative algorithms. In this case, Monte Carlo sampling allows one to estimate various characteristics of a distribution such as the mean, variance, kurtosis, or any other statistic of interest.

Exercises

1. Try out different standard distribution commands with respect to the visualization of pdf, cdf and histogram.
2. When the probability distribution is skewed (first N – 1 outcomes have zero probability), the inverse transform sampling algorithm is inefficient. What adjustment can improve the efficiency?
3. Consider random number generators implemented in MATLAB/Octave like exprnd, betarnd etc. Suppose you need to sample from a Beta(3, 2). Do this using a) built-in function; b) a rejection sampling algorithm. Compare the results. Visualize the histogram of sampled values.
4. Repeat the rejection sampling from the previous exercise for the Poisson distribution with $\boldsymbol{\lambda = 0.1}$.

PART 3: HEURISTIC OPTIMIZATION METHODS

Another application of Monte Carlo methods is solving complex optimization problems. For the most popular techniques of estimation (e.g. least squares estimation for linear models) optimization is easy and could be solved by simple algebra. However, sometimes, for a number of reasons, like multicollinearity or big model itself, computing might be problematic. Even more demanding could be optimization of general maximum likelihood estimations. In this case, likelihood functions could have several local optima, which complicates computational procedure.

There are also situations, where traditional methods could not be used for the model selection, due to the discreteness of the search space. Number of researchers mentioned this situation earlier, for example one could refer to Winker (2000) or Maringer and Meyer (2006). Discrete optimization could be generally solved by enumerating the potential solution, if that is not possible then probably there is no viable deterministic approach to solve the problem.

As one could see from the examples above, number of problems, which could not be solved with traditional approach, in econometric analysis and modelling is rapidly increasing. In this case, using traditional methods of optimization could bring inappropriate results. Unfortunately, for many applications of this type there is no simple way of checking the obtained result.

There are three possible ways of approaching the mentioned problem. First, which is not advisable, but sometimes used, is ignoring the complexity and using traditional optimization methods, hoping that obtained results would be useful anyway. Usage of this method could also result from not understanding the real complexity of the model proposed. Certainly, this is not correct to address the problem like that. Second, is simplifying the model to the state where traditional optimization could be used. This method had to be used frequently in the past, because of the lack of the computational power. This method is much more adequate then the first one, but still has its' own drawbacks, such as sacrificing the possible gains from the complex models and probability of misspecification. The third, and the most preferable approach, is using the optimization heuristics. Fortunately, this option is becoming more and more popular nowadays, since most computers could provide computational power needed for it.

It is common knowledge, that optimization is very important in econometrics, whether it is about model selection, or estimation of the parameters. This section of the book would give an overview of the heuristic optimization methods as an instance of Monte Carlo methods and illustrate it with useful examples.

3.1. Overview

The term heuristic is usually strongly connected to the algorithms that mimic natural processes, like evolution of species, annealing process of solids, or self-organization of insects. Further on we will overview several of these algorithms. The more general definition of the term, presented by Gilli and Winker (2007), relies on describing the properties of the algorithm. First, heuristic algorithm should be able to provide high quality approximations to the global optimum. Second, a well-behaved heuristic algorithm should be robust to changes in problem characteristics, In other words, it should have broad usage for the whole problem class, not only for one instance. In addition, it should be possible to tweak the algorithm, through changing the parameters or constraints. Third, which is connected to the previous one, is that heuristics should be usable for many problem instances, including new ones. As aforementioned authors conclude, there should not be any subjective elements in a heuristic algorithm.

As one can see from the named characteristics, one of the main strengths of the heuristics is that no need in strongly fixed set of assumptions about the optimization problem for the method to work properly. In most cases, even just evaluating the objective function for some element of the search space is enough. There is no need in assuming some global properties of the function or in calculating the derivatives. The drawback of this type of the optimization algorithms is that they do not produce exact solutions with certainty, but rather stochastic approximations. But none the less, if there is no option to use traditional methods, heuristics could provide satisfying results or estimations.

Right now, unfortunately practical implementation of the optimization heuristics in the econometrics cannot be called well established for a several reasons. First, as we mentioned earlier, lack of knowledge or understanding of the complexity of the optimization problem results in inappropriate use of standard methods, instead of heuristics. This constraint is highly problem specific and, unfortunately, not easy to overcome, due to lack of tools to identify the ineffectiveness of the classical methods. For example, comparing standard approach and heuristics can only indicate non adequacy of the classical tool only if heuristic solution quality is much better, but not the other way around. Good "classical" solution does not necessarily indicate that standard optimization approach is suitable.

Second, lack of well-documented implementations of heuristics. The increasing number of different heuristics and hybrids might be confusing even for experienced researcher. Solution of this issue is easier to deal with than the first one, and could be solved as more and more accessible guidelines for the heuristic methods appear. Since optimization heuristics are usually rather simple to code many successful implementations could be overviewed. Hopefully, the examples and classification of the most common heuristics provided in this book could help to overcome the named issue.

Third, dealing with the stochastic nature of heuristic optimization results could pose difficulties to researchers. Regarding this issue, it is strongly connected to the previous one, and, probably, increasing number of publications regarding theoretical and practical aspects of the heuristic optimization could eradicate this problem.

Heuristic optimization methods are generally computational, therefore they have appeared and been developing as electronic computing devices developed. First contributions in the heuristic optimization methods go back to Bock (1958) and Croes (1958) who introduced algorithms for solving the traveling salesman problem. However, the main techniques in this sphere have been introduced in the late 1980s and 1990s, but only recently they became wide-spread, since desktop computers have reached the performance required for effective usage of these methods. For the last decades, number of heuristic methods only have been rapidly increasing. We will next introduce general concepts of the heuristic optimization and provide a possible classification of the some existing variants. It should be mentioned, that today there is no generally “the best” classification of the heuristic optimization algorithms, nevertheless presented classification might help identify common characteristics of the methods.

Basic concepts

Heuristic optimization methods (also named approximation methods) are usually divided in two classes, constructive methods and local search methods. First class, generally, constructs solution through the sequence of locally optimum choices. The second one uses only information about the solutions in the neighbourhood of a current solution and is thus very similar to hill climbing where the choice of a neighbour solution locally maximizes a criterion. According to Gilli and Winker (2007), this class of approximation methods was not been considered useful for a long time, and only recently in became popular. Let us present the generalized algorithm of the local search method for minimizing a given function **f(x)**.

1. Generate initial solution $\mathbf{x^c}$.
2. Repeat steps 3-4 until certain stopping criteria (usually a certain number of iterations) is not met.
3. Select $\mathbf{x^n} \in \mathbf{N}(\mathbf{x^c})$, neighbour to the current solution $\mathbf{x^c}$.
4. If $\mathbf{f(x^n) < f(x^c)}$ then set $\mathbf{x^c = x^n}$. Since we are minimizing the target function, the smaller it gets the better.

As one can notice, it is similar to the hill-climbing algorithm, where information about the gradient for the selection of a neighbour is used, but in this case $\boldsymbol{x^n}$ is chosen according to a random mechanism. This mechanism and acceptance criteria define the way algorithm searches solutions.

This broad class of methods can be divided in two smaller classes: trajectory methods for a single solution and population based methods, which produce a set of solutions. The first class includes threshold methods and tabu search, whereas genetic algorithms, differential evolution methods and ant colonies present the second one.

Each method has own rules for choosing a neighbour and accepting the solution and in the most cases allow “uphill moves” to avoid local minima. Let us now review these methods, starting with a methods considered to be trajectory methods.

Trajectory methods

The correct definition of neighbourhood solution is crucial to these methods and generally depends on the considered problem, however finding this definition could be challenging.

Simulated annealing (SA), as a part of the threshold methods (TM), is using a probabilistic acceptance criterion and based on a similarity of combinatorial optimization and the annealing process of solids. As in the classical local search method an improvement of the current solution is always accepted. Moreover, uphill moves are also accepted, but with some probability, which is usually implemented by using RNG with a uniform distribution. This probability depends on the difference between function values and a parameter **T** (called temperature as a reference to the annealing process), that is gradually decreases every “round” of the process. Therefore, during the process probability of accepting an uphill solution is steadily decreasing. Either a total number of iterations or a number of consecutive iterations without improvement of the current solution $\mathbf{x^c}$ defines the stopping criterion. The following algorithm represents how the method works step-by-step:

1. Generate initial solution $\mathbf{x^c}$; initialize number of trials **R** and temperature **T**.
2. Repeat following steps 3-7 **R** times.
3. Repeat Steps 4-6 until stopping criteria is not met.
4. Compute $\mathbf{x^n} \in \mathrm{N}(\mathbf{x^c})$, neighbour to the current solution $\mathbf{x^c}$.
5. Compute $\Delta = \mathbf{f(x^n) - f(x^c)}$ and generate uniform random variable **u**.
6. If $\Delta < \mathbf{0}$ or $\mathbf{e}^{\frac{-\Delta}{T}} > \mathbf{u}$ then set $\mathbf{x^c = x^n}$.
7. Reduce **T**.

Threshold accepting (TA) method, suggested by Dueck and Scheuer (1990), is, essentially, a deterministic version of the SA method with the sequence of temperatures

T replaced by a sequence of deterministic thresholds $\boldsymbol{\tau}$. We will give more detailed overview of the algorithm later on in this section.

Tabu search (TS) is a method designed for analysing discrete search spaces with a finite set of the neighbour solutions. The method implements neighbour solution selection that avoids cycling, in other words, the same solution could not be visited twice. This is achieved by using short term memory, tabu list that contains the most recently visited solutions. The steps of the algorithm are following:

1. Generate initial solution $\mathbf{x^c}$; initialize tabu list $\mathbf{Tabu = \emptyset}$.
2. Repeat steps 3-5 until stopping criteria is not met.
3. Compute $\mathbf{V} = \frac{\{\mathbf{x} | \mathbf{x} \in \mathbf{N}\ (\mathbf{x^c})\}}{\mathbf{T}}$.
4. Select $\mathbf{x^n} = \min(\mathbf{V})$.
5. Set $\mathbf{x^c = x^n}$ and $\mathbf{Tabu = Tabu \cup x^n}$.

In this case set **V** could be constructed through examining of the neighbour solutions of the current solution, or alternatively could employ more complicated algorithm. Usually **Tabu** is updated by simply deleting older values and adding new ones.

Population based methods

Trajectory methods, as mentioned earlier, work only on one solution at a time, whereas population based methods are able to work with on a set of solutions (called population) simultaneously. Considering this, population based methods could be much more efficient for exploring whole search space and, potentially, provide more accurate results, at a cost of complexity and higher demands on computational power. Let us overview several algorithm of this method.

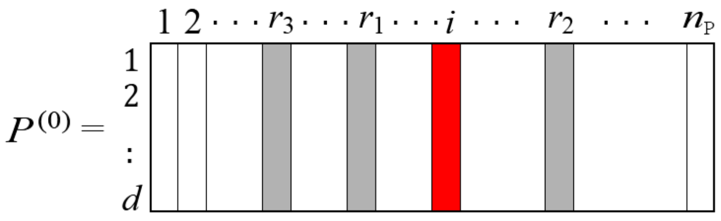
Genetic algorithm (GA) has been originally proposed by Holland (1975), and received a huge popularity. This type of search algorithms is designed to mimic the evolutionary process of species that reproduce sexually. New candidates for the solution are generated with a *crossover* mechanism, which combines part of the characteristics of each parent and then applies a random *mutation*. If the new “individual”, called *child*, inherits good characteristics from his parents it will have a higher probability to survive. The fitness of the child and parent population is evaluated in the survival function **P**, which can be formed out of several sources: the last generated individuals $\mathbf{P''}$, $\mathbf{P'' \cup \{fittest\ from\ P'\}}$, only the fittest from $\mathbf{P''}$, or the fittest from $\mathbf{P' \cup P''}$. Here is a sample algorithm representing that.

1. Generate initial population **P** of solutions.
2. Repeat steps 3-9 until stopping criteria is not met.
3. Select $\mathbf{P' \subset P}$ (mating pool), initialize $\mathbf{P'' = \emptyset}$ (set of children).
4. Repeat steps 5-8 for a desired number of iterations.
5. Select individuals $\mathbf{x^a}$ and $\mathbf{x^b}$ at random from $\mathbf{P'}$.
6. Apply crossover to $\mathbf{x^a}$ and $\mathbf{x^b}$ to produce $\mathbf{x^{child}}$.
7. Randomly mutate produced child $\mathbf{x^{child}}$.
8. $\mathbf{P'' = P'' \cup x^{child}}$.
9. **P = survive(P′, P″)** .

In other words, the algorithm in general, accepts a set of solutions on step 3, and then constructs a set of a neighbour solutions with a chosen criteria.

Another population based method is called *ant colonies* (AC) and imitates the way ants search for food and their way back to the colony. First, exploration of the neighbourhood is random. As soon as ant finds a food it starts to transport food back to the nest, leaving pheromone trail on the way that would guide other ants to the food. The intensity of the trail indicates not only the quality and the quantity of the food, but also a distance between nest the food. The shorter the distance, the more ants would travel it, thus the pheromones would be stronger. Such behaviour is able to optimize the work of the ants, since they follow the trails. This clever natural optimization mechanism have been successfully adapted in heuristics. In this case, the search area of the ant corresponds to a discrete set from which the elements of solutions are selected, the amount of food is associated with an objective function and the pheromone trail is modelled using an adaptive memory. The algorithm is working with every iteration improving the solution until the optimization goal is met.

The third algorithm of this class is called *differential evolution* (DE) algorithm, which was introduced by Storn and Price (1997). The algorithm adopts evolutionary approach and updates a population of solution vectors by addition, subtraction and crossover them and then selects the fittest solutions among the original and updated pools. The initial population of $\mathbf{n_p}$ randomly chosen solutions can be represented by a matrix $\mathbf{P^{(0)}}$ of size $\mathbf{d} \times \mathbf{n_p}$, where **d** is the dimension of the domain of the function (see Figure 3.1).



$P^{(0)} =$
1 2 · · · r_3 · · · r_1 · · · i · · · r_2 · · · n_P
1
2
:
d

Fig. 3.1. Initial population of solutions in DE algorithm (Source: Gilli and Winker, 2007*)*

The algorithm constructs $\mathbf{n_G}$ generations of the population. It takes several steps to obtain new generation:

1. For each solution $\mathbf{P_i^{(0)}}$, $\mathbf{i = 1,\ldots, n_p}$, that is highlighted column of the Figure 3.1, the algorithm constructs two intermediate solutions $\mathbf{P_i^{(v)}}$ and $\mathbf{P_i^{(u)}}$ from three randomly selected columns $\mathbf{P_{r1}^{(0)}}$, $\mathbf{P_{r2}^{(0)}}$, and $\mathbf{P_{r3}^{(0)}}$.
2. The i-th solution $\mathbf{P_i^{(1)}}$ of the new generation is assembled from the intermediate solutions and the initial one.
3. Then this steps repeat until optimization problem is not solved.

The process is also controlled by the additional parameters that affect the speed of shrinkage in the exploration of the solutions area, or crossover probability. These parameters are depend on the problem considered, thus chosen depending on the population or some other specifics.

Optimization as stochastic mapping

Stochastics is a prevailing theme in the optimization techniques, which at the same time differs from stochastic of the random sampling. In case when repeated applications of an optimization method do not generate identical results, one have to deal with this type of stochastics. As one might have guessed, for the optimization methods overviewed earlier many factors could create randomness, for example generating initial solution, selecting candidate solution, or acceptance process. On the one hand, judging from the practical side, additional randomness is highly unwelcome characteristic of heuristics. However, on the other hand, because of the limitations of the classical approaches they could provide deterministic results that are far from optimal in some cases. In the situations like this, it is obvious, that stochastic approximation to the true optimum is much better than a bad deterministic result. Therefore, stochastics of the outcome of the heuristics should be reviewed.

Consider $\boldsymbol{\psi}^{\mathbf{I,r}}$ as the result of a run **r** of some heuristic optimization algorithm **H** for a given objective function **f**. Let **I** denote a measure of the resources spent on a single run of the algorithm, and $\mathbf{f(\boldsymbol{\psi}^{I,r})}$ represent obtained value of the objective function, which could be seen as a random draw from a distribution $\mathbf{D_I^H(\boldsymbol{\mu}_I,\boldsymbol{\delta}_I,\ldots)}$ with a certain and finite expectation and variance values. It is worth mentioning that, although for the most heuristics properties of this distribution are unknown, some general characteristics could be derived. First, for minimization problems, the distribution will be left censored at the global minimum of the objective function. Second, with increasing amount of computational resources **I**, the distribution tend to shift left and become less dispersed. It is showed in detail in the work by Gilli and Winker (2007).

If heuristic optimization is repeatedly applied to the same problem, it becomes possible to calculate the empirical distribution of the objective function. This information, in its turn, allows estimating the properties of the $\mathbf{D_I^H}$ and its empirical characteristics. In particular, lower values are interesting as estimates of the best solutions for the minimization problem. Extreme value theory could possibly be used to obtain estimates of the unknown global. Finally, repeating the algorithm for different amounts of computational resources **I** allows estimating empirical rate of convergence.

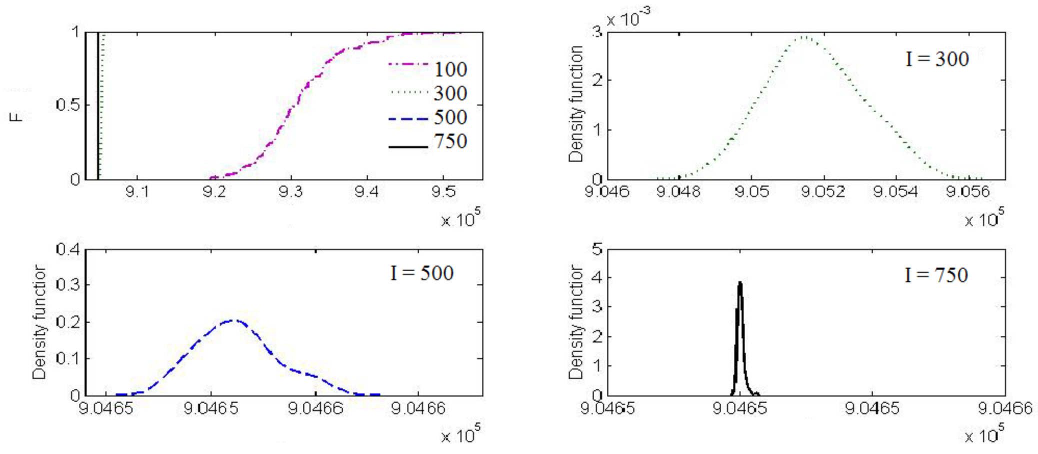


Fig. 3.2. Empirical distribution of f for different values of I (based on n = 1 000) (Source: Blüschke et al., 2013)

Figure 3.2 presents results from an application of the threshold accepting method, mentioned earlier, to the problem of lag structure selection in VAR models. The upper left plot exhibits the empirical distribution functions of the objective function (Bayesian information criterion) for different values of **I**. As predicted, the distributions move left and become less dispersed, as **I** grows. The other three plots show kernel density estimates of the distribution of $\mathbf{f(\psi^I)}$ for different numbers of replications. These plots confirm the findings from the empirical distribution function. In addition, they might hint the information about specific properties of the objective function. For example, the shape for the lower right plot probably indicates two local minima with a large domain of attraction, whereby the first one might be the global minimum.

Convergence of heuristics

As we mentioned, optimization heuristics does not provide deterministic result, so it is important to describe the convergence of these methods as a part of our overview. It is intuitively understandable, that increasing amount of computational resources (e.g. increasing a number of total runs) should have a positive effect on the accuracy of the result if algorithm is tuned correctly.

To illustrate this idea we present the results for threshold accepting obtained by Althöfer and Koschnik (1991). Similar logics could apply for other heuristics. This work are generally confirm the theoretical assumption, that with increased **I** results of the algorithm become more accurate. Let us look into this in more detail.

First, results are obviously depend on a several assumptions regarding the problem. To be precise, search space has to be connected, in other words, it should be possible to reach any point of the search space from any given starting point by performing local search steps, and the objective function has to satisfy some defined criterion. Second, authors prove that there are suitable parameters for the threshold accepting implementation such that the global optimum of the objective function can be approximated at arbitrary accuracy with any given probability close to one by increasing the number of iterations. However, finding the suitable parameter values could be rather complicated in practice.

Althöfer and Koschnik (1991) present following convergence result: with any $\boldsymbol{\delta > 0}$ and $\boldsymbol{\varepsilon > 0}$, there exists a number of iterations $\mathbf{I(\delta, \varepsilon)}$ such that for any given replication $\mathbf{r}$

$$\mathbf{P\left(|f(\psi^{I,r}) - f_{min}| < \varepsilon\right) > 1 - \delta},$$

Where $\mathbf{f_{min}}$ denotes the global minimum of the objective function. There are several notes that should be mentioned about this result. The main one is that result does not provide any information about the rate of convergence, which is extremely important since there is no possibility to send infinite resources on the algorithm. In addition, for any finite amount of **I** there is an error in the approximation of the estimators. Therefore, it seems that further research in this field is strongly needed.

Guidelines for the use of heuristics

As we covered the essential basics of the heuristic optimization methods, we now will provide several useful guidelines for using these methods in practical researches. Let us first explain when these methods are used. Generally, for the situation when the optimization problem is complex in a computational sense, for example due to several local minima, heuristic optimization should be used as a benchmark for "traditional" approach. When, in this case, heuristic optimization would provide better results than classical approach, it should be an indication to apply the heuristics as a main optimization technique. Probably, heuristics could even be used as a standard benchmark, since many software packages like MATLAB now support it by default.

Also when choosing the heuristics for optimization, researcher must correctly choose the type of the applied algorithm. First, selection could be based on the type of the search space and the objective function given, since some heuristics, like differential evolution, for example, are not suited for discrete search space or problems with noise parameters in the function. Second, researchers could use previous experience of using heuristics in certain spheres where problems have similar characteristics. In addition, it should be correct to start with more general approaches first, and then using more specific heuristics if needed.

Each of the optimization algorithm, that we earlier listed in the section has its' own set of tuning parameters, which should be controlled by researcher to increase effectiveness of the algorithm. For example, in trajectory methods the neighborhood solution should be defined carefully and neighbor solution should be easy to generate. In a genetic algorithm information is exchanged among individuals during crossover with mutations occurring, thus information transmitted from parents and diversity should be controlled.

Let us illustrate implementation of the heuristic optimization with a threshold method as done by Gilli and Winker (2007). For the following, we consider a minimization problem on a subset of $\mathbf{\Omega}$ of $\mathbf{R}^{\mathbf{k}}$. The implementation of this algorithm involves specifying number of restarts $\mathbf{n}_{\mathbf{restarts}}$, number of rounds $\mathbf{n}_{\mathbf{rounds}}$, and number of steps $\mathbf{n}_{\mathbf{steps}}$, then defining objective function, the neighborhood, and the threshold sequence. In this case the number of iterations per restart $\mathbf{I}$ is defined by number of rounds multiplied by number of steps.

Optimization in terms of calculation is crucial for the heuristics, therefore researcher should always try to achieve fast calculation when constructing the algorithm. To improve the performance of the threshold method the objective function should be locally updated, in other words with a change of $\boldsymbol{x}$ function should not be computed from scratch, but rather just updated. Such approach have also been proven useful in the population based algorithms. Unfortunately locally updating functions could be problematic in practice.

The next thing to consider is presence of constraints, in which case **Ω** is a subspace of $\mathbf{R}^k$ and generating starting and neighbor solutions could be difficult, especially if **Ω** is not connected. Therefore, as it pointed out by Gilli and Winker (2007), $\mathbf{R}^k$ should be used as a search space with adding certain penalty to the objective function if **x** does not belong to space **Ω**. Usually, in practice, this penalty is initially set to be small to allow exploration of the whole space, and then increased every run to increase the accuracy of the solution. Penalties could be presented in absolute or relative terms, while second option allows more generalized usage.

Now, let us move to the neighborhood definition. Generally, in the case of TA, for real valued variables the definition could be given by the means of **ε**-spheres:

$$\mathbf{N(x^c) = \{x^n | x^n \in R^k} \text{ and } ||\mathbf{x^n - x^c}|| < \boldsymbol{\varepsilon}\},$$

Where **N** denotes the neighborhood and **||…||** represents Euclidian metric, which is case of discrete search space could be substituted with a Hamming metric for better results.

Unfortunately, theoretical analysis of the threshold accepting algorithm does not provide a guideline on how to choose the threshold sequence. In some cases of small problems the optimal threshold sequence is not monotonically decreasing. Fortunately, there are two useful procedures that appear to provide reliable threshold sequences for use in the field of econometrics. First and the simplest one, is to just use sequence linearly decreasing to zero over the chosen number of periods, in this case only first threshold value needs tuning. The second approach is to use data driven generation of the sequence, which relies in the function structure itself. Let us look into this option further.

For the case of a finite search space, only threshold values corresponding to the difference of the objective function values for a pair of neighbors are relevant. To be applicable to the real search spaces the algorithm uses a random sample from the distribution of such local differences. This procedure can also be successfully applied to the cases of infinite and continuous search spaces. The following algorithm gives a general idea how this approach works:

1. Randomly choose $\mathbf{x^c} \in \boldsymbol{\Omega}$.
2. Repeat steps 3-4 for a chosen amount of times $\mathbf{n_{delta}}$.
3. Compute $\mathbf{x^n} \in \mathbf{N}(\mathbf{x^c})$ and $\Delta_i = |\mathbf{f}(\mathbf{x^c}) - \mathbf{f}(\mathbf{x^n})|$.
4. Set $\mathbf{x^c} = \mathbf{x^n}$.
5. Compute empirical distribution $\mathbf{F}$ of Δ_i.
6. Compute threshold sequence $\boldsymbol{\tau_r} = \mathbf{F}^{-1}\left(\frac{\mathbf{n_{rounds}} - \mathbf{r}}{\mathbf{n_{rounds}}}\right)$, $\mathbf{r} = \mathbf{1}, \ldots, \mathbf{n_{rounds}}$.

The resulting empirical distribution of Δ is shown in the Figure 3.3.

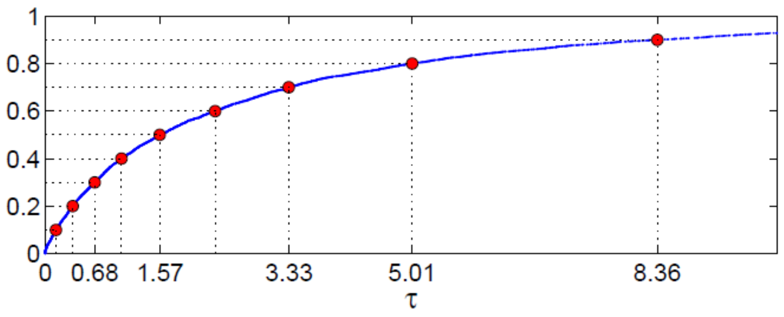


Fig. 3.3. Empirical distribution of a sequence of Δ (**τ** indicator to number of rounds) (Source: Gilli and Winker, 2007)

Instead of considering the local changes of the objective function Δ along a path through the search space as described in the algorithm, researcher might consider several restarts to produce different trajectories of shorter length, or – in the limit – generate all $\mathbf{x^c}$ randomly. It should be noted that all three methods should give the same approximation to the distribution of local changes of the objective function with infinite amount of tries.

Next we overview how the algorithm could be monitored. To understand how the algorithm works it is possible to produce a plot of the function values accepted in the succeeding rounds. The Figure 3.4 shows such plot, using threshold determined previously. As it can be seen, the uphill moves are less frequent as more rounds that are successful are taken. In the last round there are no uphill moves at all.

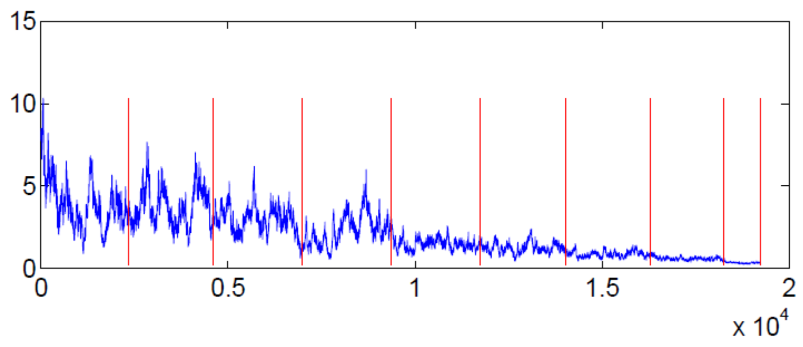


Fig. 3.4. Function values for accepted steps in the local search, divided by rounds (Source: Gilli and Winker, 2007)

The last two figures could provide researcher with useful information about how well algorithm is parameterized. For example, an irregular shape of the cumulative distribution of the Δ should be a signal of a bad local behavior of the objective function. Graph of the accepted values could also identify if number of steps and rounds is chosen correctly. Usually number of steps should by far exceed number of rounds, it is suggested that algorithm have at least 10 rounds, and increasing this number benefits the accuracy. Obviously, the choice of these parameters are problem dependent, thus if evaluation of the objective function is expensive they would be kept as low as possible.

Although as we mentioned earlier, estimates generally benefit from the increased number of iterations **I**, it might be reasonable to divide available resources into several restarts. It should be optimal in many cases to set number of restarts $\mathbf{n_{restarts}}$ from 5 to 20, rather than investing all available resources in one go. If the restarts are executed in a distributed computing environment, the optimal allocation of computing resources has to be considered differently. Again, the question is how to choose $\mathbf{n_{restarts}}$ and **I** in order to minimize execution time for a given quality of the result $\mathbf{f_{sol} = min_{r = 1,\dots,n_{restarts}} f(\psi^{I,r}) \leq c}$.

The following example is suggested by Gilli and Winker (2007). Consider Figure 3.5 showing the empirical distribution of results $\mathbf{f(\psi^I)}$ of the heuristic optimization for increasing values of **I**. Authors associate a Bernoulli random variable **z** to the solution $\mathbf{f(\psi^I)}$ of a single execution where $\mathbf{z = 1}$ if $\mathbf{f(\psi^I) < c}$ and $\mathbf{z = 0}$ else. For

$\mathbf{I = 10000}$ and $\mathbf{c = -1.265}$ they received $\mathbf{p = P(z = 1) = 0.727}$ which corresponds to the percentile of $\mathbf{f(\psi^I) = -1.265}$ in the empirical distribution. Then authors considered the random variable $\mathbf{x}$ which counts the number of solutions verifying $\mathbf{f(\psi^I) < c}$ out of solutions. Knowing that $\mathbf{x \sim B(n_{restarts},\ p)}$ is a binomial random variable, the probability for a solution being at least as good as $\mathbf{f_{sol} = \min_{r=1,\dots,n_{restarts}} f(\psi^{I,r}) \leq c}$ is given by

$$\pi = 1 - P(x = 0).$$

From which it is possible to compute the number of restarts necessary to obtain the desired quality $\mathbf{c}$ with a given probability $\boldsymbol{\pi}$. For $\mathbf{x \sim B(n,\ p)}$ we have $\mathbf{P(x = 0) = (1 - p)^n}$ and the number of restarts Gilli and Winker were seeking is given by the smallest positive integer $\mathbf{n}$ verifying $\boldsymbol{\pi} \leq \mathbf{(1 - p)^n}$.

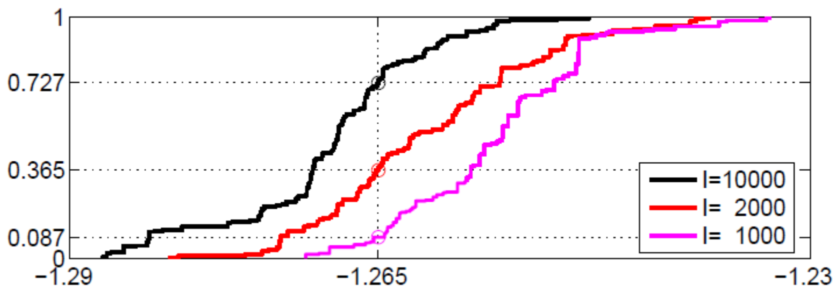


Fig. 3.5. Empirical distribution of results of $f(\psi^I)$ (Source: Gilli and Winker, 2007)

As a result authors concluded that $\boldsymbol{I = 1000}$ and $\mathbf{n_{restarts}} = \boldsymbol{51}$ would be the best choice in a distributed computing environment with at least 51 nodes, whereas $\boldsymbol{I = 2000}$ and $\mathbf{n_{restarts}} = \boldsymbol{11}$ is the best choice if the restarts have to be executed serially. Moreover, they mention that TA algorithm with restarts behaves robust within given class of problems.

Let us next consider several possible application of the heuristic optimization for discrete and continuous search spaces.

3.2. Applications to Discrete Search Spaces

In this section we will give an illustration of the several issues mentioned in the previous section. First example employs TA heuristic as an illustration of model selection in VAR models.

Consider a p-dimensional vector AR-process $\mathbf{X_t}$ that is given by

$$\mathbf{X_t} = \sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{\Gamma_k X_{t-k}} + \boldsymbol{\varepsilon}_\mathbf{t} \,,$$

Where matrices $\Gamma_\mathbf{k}$ provide autoregression coefficients and $\boldsymbol{\varepsilon}_\mathbf{t} \sim \mathbf{N(0, \sum)}$ **iid**. In this case all component time series are stationary.

Assume that maximum lag length $\mathbf{K_0}$ can be derived either from an economic background or from a certain rule of the thumb based on number of observations. Therefore, the model selection problem for a given realization of the process includes identifying **K** and non-zero elements of $\mathbf{\Gamma_k}$ in the search space described by $\mathbf{\Omega = \{0, 1\}^{p^2 \times K_0}}$ where 0 indicates parameter constraint to be zero, and 1 to the freely estimated parameter. It can also be mentioned, that for this problem different objective functions can be considered. For this problem the Bayesian or Schwarz information criterion $\mathbf{BIC = \ln\left|\hat{\sum}\right| + l \ln T/T}$ (where $\hat{\sum}$ denotes determinant of the fitted residuals covariance matrix and **l** denotes number of non-zero parameters) is used, but using any other information criterion (e.g. Akaike information criterion) is possible. It could be implemented just by replacing the objective function.

We assume that $\mathbf{\Omega}$ is finite, which allows solving the problem by simple enumeration of all elements of the set and choosing lag structure that corresponds to the lowest value of the chosen information criterion. However, this method is not applicable to the real problem, since even modest values of $\mathbf{p^2}$ and $\mathbf{K_0}$ returns a big amount of the lag structures, which makes enumeration practically impossible. Another way is to consider only a small subspace of the $\mathbf{\Omega}$, which could be enumerated. However, there is no guarantee that optimal lag structure will lie within this subset. Therefore, using heuristics is reasonable at least as a benchmark.

As mentioned in the previous section, neighborhood for the TA heuristics could be defined by the means of $\boldsymbol{\varepsilon}$-spheres defined by Hamming distance. In this case, generally, $\boldsymbol{\varepsilon}$ should be chosen to be large enough for not being stuck in a bad local minima and small enough to result in a guided local search. For the considered problem with a 5000 iterations per restart Hamming distance of 4 appears to be the most adequate choice since standard deviation of the accepted values is the lowest (see Table 3.1). The threshold sequence is constructed following the algorithms that were described earlier with 30 rounds with 100 steps each.

Table 3.1

Standard deviation of local changes and accepted solutions

distance	$\sigma_{\Delta N}$	$\sigma_{accepted}$
2	0.1706	0.9569
4	0.2578	0.5961
6	0.2916	0.7932

We next consider the problem with $\mathbf{p = 3}$ and $\mathbf{K = 3}$ for simplicity. The problem instance exhibits different lag structures for the three equations and several uncertainties such that the standard approach cannot result in a proper model selection. The selection procedure is based on the simulated data of this process with $\mathbf{T = 100}$ and $\mathbf{K_0 = 5}$, for the optimization purposes.

Next, we consider several instances of the realizations of the process with 1000 restarts, but with different number of total iterations **I**. For $\mathbf{I = 500}$ the overall best solution is found 4 times, 18 times for $\mathbf{I = 1000}$, 217 times for $\mathbf{I = 5000}$, and 332 times for $\mathbf{I = 10000}$. At the same time with increasing **I** the mean relative deviation from the best solution decreases from the 22 % in the first run to the 1.3 % in the last one. With a low number of observations of the realization of the process, it should not be expected that the model with the lowest value of the information criterion found by the heuristics corresponds to the true data generating process. The heuristics just delivers the best or at least a good approximation for the given realization. Therefore, to gain accurate results one should use larger set of different model structures and realizations for each of them. More details on model selection in VAR models including more comprehensive set of the simulations can be found in Winker (2001, Ch. 12).

3.3. Applications to Continuous Search Spaces

Next, we present a comparison of the implementations of threshold accepting and differential evolution in case of continuous optimization problem Consider the linear regression model

$$y_i = \left[x_{i1} \ldots x_{ip}\right] \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_p \end{bmatrix} + \epsilon_i,\ i = 1, \ldots, n,$$

Where matrix of θ denotes parameters and $\epsilon \sim N(0, \sigma^2 I)$ is the disturbance vector. In this case, the smallest percentage of contamination in the data that may cause the estimator to be affected by an arbitrary bias is the breakdown point of the estimator (see Hampel et al., 1986, Ch. 2). For normal OLS breakdown point is 0 %, but there are some other estimators that have breakdown point of 50 %. It is a matter of fact, that estimators with high breakdown point could not be solved as simple as OLS, since their objective functions are non-convex and have several local minima. Over the last two decades, a number of the algorithms have been proposed to solve this problem, for example one can refer to Marazzi (1992) and Rousseeuw and Driessen (2000). These methods are using, essentially, resampling techniques. Recently, the algorithms have been revised and optimized for faster execution, e.g., by Salibian-Barrera and Yohai (2006) and Rousseeuw and Driessen (2002). The resulting algorithms are complex ad hoc procedures. We demonstrate that standard heuristic optimization techniques that could be used for solving such problems.

Consider an example of least median of squares (LMS)

$$\hat{\theta}_{LMS} = \underset{\theta}{\operatorname{argmin}}\, Q_{LMS}(\theta),$$

Where $Q_{LMS}(\theta) = \operatorname{med}\left(r_1^2 \ldots, r_n^2\right)$ is the median of squared residuals r_i^2. Apart from that least trimmed squares and the S-estimator are used.

As an illustration for the computation of an LMS estimator the data generation process which has been proposed by Salibian-Barrera and Yohai (2004) is used. Consider the linear regression model mentioned earlier where 90 % variables are independent and identically distributed random variables with normal distributions, therefore $\theta_i = 0,\ i = 1, \ldots, p$. Remaining 10 % consist of outliers corresponding to slopes $\theta_2 = \frac{M}{100}$ and $\theta_j = 0$ for $i \neq 2$.

In the following the data is generated for $n = 100$, $p = 10$, and $M = 190$, and then LMS estimators are computed. It should be noted, that these estimators should significantly differ from zero. Consider Figure 3.6, which illustrates objective function non-convex shape. The left part represents data that was generated with the artificial model presented above. One could mention that local minima for this function corresponds to $\boldsymbol{\theta_2 = 1.9}$. The right part presents real data related to the biomedical problem (Brown and Hollander, 1977).

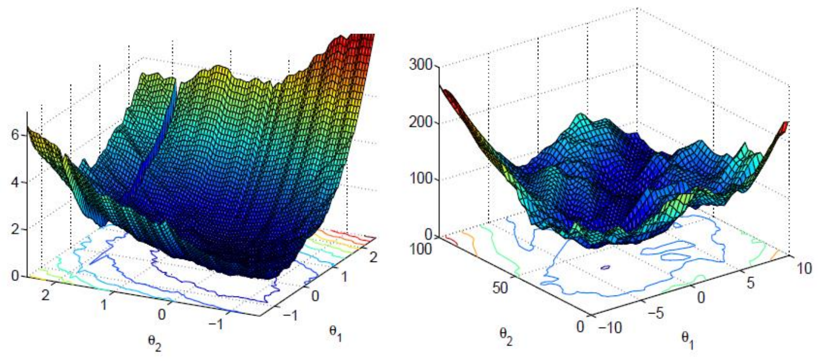


Fig. 3.6. Shape of function $Q_{LMS}(\theta)$ for the LMS (Source: Gilli and Winker, 2007)

Next, we proceed with the possible implementation of the TA algorithm for the minimization problem with a continuous objective function. As earlier mentioned $\mathbf{Q_{LMS}(\theta) = med\left(r_1^2 \ldots, r_n^2\right)}$, where $\mathbf{r_i = \left(y_i - X_i, \theta\right)^2, i = 1, \ldots, n}$. To generate the solution $\boldsymbol{\theta^n}$ the following algorithm is employed:

1. Set $\boldsymbol{\theta^n} = \boldsymbol{\theta^c}$.
2. Select uniformly $\mathbf{i} \in \{\mathbf{1}, \ldots, \mathbf{p}\}$.
3. Generate uniform random variable $\mathbf{u} \in [-\mathbf{h}, \mathbf{h}]$.
4. Compute $\boldsymbol{\theta_i^c} = \boldsymbol{\theta_i^c} + \mathbf{u} \cdot (\mathbf{1} + |\boldsymbol{\theta_i^c}|)$.

In this simulation, h is set to 0.40 and the algorithms performs total of 25000 iterations divided into 10 rounds 2500 steps each. The results could be seen in the Figures 3.3 and 3.4.

Another way to solve the problem is to employ the DE algorithm. In this case objective function remains unchanged. To make results comparable we set $\mathbf{n_P = 100}$ and $\mathbf{n_G = 250}$, which results in the same number of iteration as in the first situation. Other relevant parameters are set to be the following: $\mathbf{CR = 0.8}$ and $\mathbf{F = 0.75}$,

$\theta_i \in [-3, 3]$ for $i = 1, \ldots, p$. The empirical distribution of the objective function value for the TA and DE is shown in the Figure 3.7.

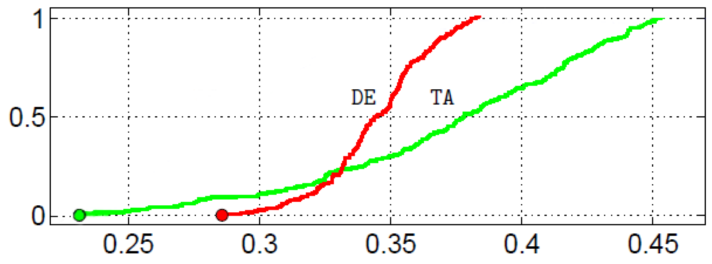


Fig. 3.7. Empirical distribution of the value of the objective function of 200 solutions for TA and DE (Source: Gilli and Winker, 2007)

In the considered problem, all solutions show the presence of outliers, in other words none of the estimated $\theta_2 = 1.9$. We could also conclude, judging by the graph, that for this case differential evolution produces slightly better solutions than TA. The very small variance of the solutions also indicates that only a small number of restarts, around 10, is needed to obtain with high probability a solution identifying the outliers.

Exercises

1. (based on Kroese et al. 2011) Consider minimization problem for the trigonometric function for the n-dimensional x_i:

$$S(x) = 1 + \sum_{i=1}^{n} \left(8\sin^2\left(\eta(x_i - x_i^*)^2\right) + 6\sin^2\left(2\eta(x_i - x_i^*)^2\right) + \mu(x_i - x_i^*)^2\right),$$

   Where $n = 10$, $x^* = (10, \ldots, 10)$, $\eta = 0.8$, and $\mu = 0.1$ are set in the named values. Function has minimal value of 1 at $x = x^*$. Solve the problem using the method of the Simulated Annealing. For construction of the algorithm use following parameters $T_0 = 10$ as a starting temperature, $\beta = 0.99$ for cooling factor with geometric cooling scheme, $\sigma = 0.75$, with 10^3 iterations. Comment on results of 20 restarts of the algorithm.

2. Repeat the same algorithm for 10^6 iterations with 10 restarts. How does this change affect results?

3. Consider minimization problem for the 50-dimensional Rosenbrock function (which is considered to be a benchmark for optimization algorithms):

$$\mathbf{S(x)} = \sum_{\mathbf{i=1}}^{\mathbf{49}} \mathbf{(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2)},$$

   Which has minimal value of 0 at **x = (1, …, 1)** (Kroese et al. 2011). Solve the problem using differential evolution method. For the algorithm use population size **N = 50**, scaling factor **α = 0.8**, crossover probability **p = 0.9**, run the algorithm for $5 \cdot 10^4$ iterations with uniform initialization of the population. Comment on the results.

PART 4: INTRODUCTION TO THE AGENT-BASED MODELING

Agent-based models (ABMs) or, alternatively, individual-based models (IBMs) are type of the computational models that are used for simulating the behavior of autonomous agents to assess how the system consisting of the several agents behaves. This group of models is rather new, but have proven to be effective in economic theory and many social sciences. The randomness needed for simulation is achieved through the employment of the Monte Carlo methods.

Compared to the variable-based approaches that use structural equations, or system-based approaches that employ differential equations, agent-based modelling is capable of simulating individual heterogeneity that represents explicitly agents' decision rules, and situates agents in a geographical or another type of space, which gives virtually infinite number of possible applications of this method.

This part of the book reviews a concept of the agent-based models, how they work, what features differ them from the other types of models, and gives several possible ways of implementing agent-based models in a research. This part also provides several useful examples of agent-based models.

4.1. Overview

Agent-based modeling is, essentially, a computational method, which allows creating, analyzing, and experimenting with complex models composed of autonomous agents that interact within an environment and each other. This approach effectively combines game theory, computational sociology, multi-agent systems, and evolutionary programming. Let us consider this definition in detail as given in Gilbert (2007).

As it is stated, agent-based modelling is a computational approach, thus involves creating models and algorithms that could be executed by using computer resources. In this case, modeling is used in its' common meaning, as creating the simplified representation of the real phenomena or object, which bears its' distinctive features of traits. In econometrics, for example, a regression equation with dependent variable and independent variables, serves as a model of the relations between the variables. Even a description in words could be viewed as a model, for example describing how foreign direct investment affects innovation activities in the certain country, we create a model of the real processes, thou rather simple one.

Computational models are usually presented as a computer programs consisting of several input parameters that following some kind of algorithm result in the output values that represent the modelled process. As an example, one could have assumptions about how consumer tastes are affected by advertising or social relations. To test them, one should create a computer program, which has a set of agents (or individuals) that make buying choices according to their preferences. In this case, every agent would be affected by choices of their "friends" (and the other way round) and the advertising as some kind of "external" power. And since traditional methods appear too complex to use in a models with a big number of agents, such processes could be modeled using agent-based approach.

There are several big advantages of the computational modeling. First, it is expressed in a precise computer language, thus returning precise numerical results, whereas theories and models in natural language could be sometimes uncertain or formally assessed. Second, computer programs are good for modelling processes, since programming environments usually provide a variety of tools for that. Third, as we several times mentioned earlier, computational power of the machines is rising rapidly, allowing modeling more complex, and sometimes unpredictable, processes with many events or agents involved.

Next thing to consider is that ABMs allow experimenting with the models. It is important mainly because in social sciences, unlike physics or chemistry for example, experimenting is very hard to implement or extremely costly. It is mainly because that classical "treatment-control" method of experimenting could not be applied in the social sphere. In the physics or chemistry studies experiment consist of applying certain treatment to an isolated system and then controlling the results, comparing this system with another one that did not received any treatment. Obviously, the same could not be applied to the social studies, since the complexity of the social systems and a number of ethical reasons. However, employing an agent-based modeling in the research negates the difficulties in isolating of the human system and the ethical problems of experimentation, since experiments are conducted on the virtual systems. This could also be stated as a major advantage of ABM. The desired experiment could me set up and conducted for an almost unlimited number of times applying a range of exogenous parameters to the system and even allowing random variations to appear.

This idea could not be called novel. For example, architects or engineers test models of buildings or planes in a wind tunnel, since it is much cheaper and easier to use the model rather than a real object. Another reason for experimenting with models is that sometimes it may be the only way to obtain results, like for example in a quantum physics, since particles are almost impossible to observe directly. Deriving the behavior of a model analytically is usually preferred, since it provides information about how the model will behave given a range of inputs, but often an analytical solution is a viable choice. For these cases, it is necessary to experiment with model differing inputs to gain information about its' behavior. The same logic is applied to the social experimentation.

Obviously, to achieve reliable results with such kind of experimentation the researcher should ensure that the model used behaves in the same way as real-world system, which could be challenging sometimes. Thus, experimentation on models is not a universal solution.

Another core concept of this section is model. Models now have a long history of usage in the social studies. Modelling social processes took place even before usage of computers, but became widely spread when statistical models began to be used as a tool for analyzing the large amounts of data in economics, sociology and demography. As it was mentioned earlier in this part of the book, models are meant to simulate some real phenomena, which is called the target of the model. As mentioned by Carley (1999) two main advantages of a model are that it succinctly expresses the relationships between features of the target, and it allows one to discover things about the target by investigating the model.

One of the earliest social science models is hydraulic model of the economy by Phillips, also called Monetary National Income Analogue Computer (MONIAC), in which water running through a system of pipes and vessels is used to model a circulation of money in the economy. The model also allows changing of the parameters, such as interest rate, by changing the rate of flow of water through the pipes to see how this affects the whole system. Let us now overview several most common types of the models. All of these types are not exclusively used by itself, but rather used together to make the model more accurate.

Scale models are just smaller versions of the target, however apart from reduced size models are reduced in the level of complexity. For instance, scale model of the building, mentioned earlier, would have the same shape as the target, but none of its water or electricity systems. City map could also be called scale model, since it is obviously much smaller than the city itself, and represents only positions of the buildings, but not their height or other parameters. While studying this type of model one should remember that results from the model should be scaled to the target's dimensions and consider the possibility that not modelled features may affect the outcome.

Another type of models are ideal models. In this case, some characteristics of the target are exaggerated as a mean of simplifying the model. Gilbert (2007) gives the example of an idealized model of a stock market, which assumes that information flows from one trader to another instantaneously, and an idealized model of traffic that may assume that drivers always know the perfect rout. The main idea of the idealizing in modeling is to remove complicated factors from the model, thus simplifying it and obtaining viable results if these factors have negligible effects on the model.

Models that rely on drawing an analogy between some already understood phenomenon and the target are called analogical models. One of the most well-known examples is the billiard ball model of atoms. Social sciences examples include the computer model of the mind by Boden (1988) and the garbage can model of organizations by Cohen, March, & Olsen (1972). These models are usually very useful, since phenomena used for the analogy are usually well known and could provide good estimations, if the analogy is adequate.

The fourth category are mathematical or equation-based models. These emotes are most used in economics and quantitative sociology. They usually specify relationships between variables, but unlike previous models, they do not use any kind of analogy or resemblance between the model and the target. Instead, they use the mathematical functions to indicate it. The quality of this kind of models is indicated by how good certain data fits the equation. As an example for this kind of models, the Cobb-Douglas production function could be mentioned. It is a widely used mathematical model of how manufactured outputs are related to inputs:

$$\mathbf{Y = AL^{\alpha}K^{\beta},}$$

Where **Y** denotes output, **L** – labor input, **K** – capital input, and **A**, α, and β are constants determined by the current technology. The form of this equation was derived practically by using the data, not by making theoretical assumptions. Mathematical models are extremely useful in analyzing how variables or phenomena are related, but unfortunately usually they do not give clear information why these phenomena are related. Therefore other types of models are better if process or mechanism are the main concern.

As we mentioned, agent-based models recreate the situations where agents interact within a certain environment. Agents are used in the social sciences to represent individuals, organizations, firms, or even countries. They are usually parts of the program (or sometimes even separate programs) that react to their computational environment as a model of the real social environment. Curtail feature of the ABMs that agents are able to interact, thus pass information between each other, which represents human communication by the means of dialogue, observation, or any way transmitting information. This feature differs agent-based modelling from the other approaches.

Let us look more into what is the environment in the ABM. It may be neutral virtual world that has little or no effect on the agents, or alternatively have strong influence on them. Environments could represent geographical spaces, for example in the models concerning international relations, environments may represent features of country or its regions. Models in which the environment represents a geographical space are called spatially explicit. In this case, agents have coordinates that show their location. In other types of models environment also represents certain space but not geographical one, like ‘knowledge space’. Agents could be also linked together in a network where relationships of one agent to the others shows its position in the network.

There are several main criterion of ‘good’ ABM. For example, Frenken (2006) presents five of them, let us overview them. First of all, ABMs usually use ‘stylized facts’, like wide-spread empirical regulations, rather than exact empirical data. Which can cause problems of the overidentification. To overcome these problems researcher could use additional criteria to select the most appropriate model, and complimentary build models to replicate multiple stylized facts in a single model for a wide range or probable range of parameter values. Therefore, if model replicates a large number of stylized facts it is considered to be good under ROSF (replication-of-stylized-facts) criterion. In addition, it is able to estimate how good existing models are able to account for the new stylized facts, thus evaluating its’ robustness.

Another popular criterion, which is usually used for the evolution models that would be explained in the next section, is called ROAS, which stands for the realism of the assumptions. This means that considered model should not contradict well known behavioral or environment facts. In the neoclassical models opposed to that instrumentalist philosophy is used, according to which the prime quality criterion of models is ability to make accurate predictions.

Third, and arguably, the most basic, is quality criterion called KISS which stands for ‘Keep it Simple, Stupid’. This is usually understood as a preliminary recommendation to start modelling from a simple model that could be easily understood and produce understandable results and only then develop the model in the specific direction. In this sense, it is challenged by newly developed KIDS (Keep it Descriptive Stupid) principle, proposed by Edmonds and Moss (2004). It means that simulation should be done in the most straightforward way possible, adopting the widest range of evidence. And simplification only used when it is justifiable by data or the model itself. Whereas, KISS models are initially simple, which could potentially lead to undesirable oversimplification. Alternatively, KISS principle could be applied ex post, checking if the model is the simplest model that could produce desired outcomes and fulfilling ROSF and ROAS criteria. Nowadays there is a number of opinions supporting the idea that simple models are more suitable to be assessed with real data than complicated models that are usually motivated by the desire to imitate the reality more closely.

Last criterion mentioned by Frenken (2006), known as ‘Take A Previous model and Add Something’ abbreviated TAPAS states that incremental approach is usually the most effective. As a previous criterion this one is meant to be a modelling heuristic, but also could be used in the optimizing the model. The main idea of this criterion is to use existing models and propositions as a base to maximize cumulativeness of knowledge and minimize time for development of the new model, moreover this approach makes new models more understandable in contrast to models built from scratch. Obviously, this does not mean that there is no place for experimental an innovative models in ABM, but rather that if it is possible to take advantage of the already existing models it should be taken, which makes them faster to construct and easy to understand. Considering the last two principles, it is obvious, that large part of the agent-based modelling should be devoted to construction of simple core models that could be then easily modified for more specific use.

To conclude this section let us overview distinctive features of the agent-based modeling that are outlined and explained by Gilbert (2007). Each of them illustrate what differs ABMs from other types of models that.

Ontological Correspondence. Agent-based models allow direct correspondence between the real-word actors and the modeled agents, which makes designing and interpreting the model easier than with equation based models. Agent-based model of the production firm may include agents representing the suppliers, the employees (that could be divided into different groups by their roles), the customers, and any other actors possible. The model could use only one agent for each class of actors, or alternatively could use separate agent for each main actor in the environment if this is important for the results.

Representation of the Environment. Another feature of ABMs is a possibility to represent directly real environment in the model, whether it includes physical characteristics (e.g. distance between agents or whether in the region), effects of the surrounding agents, or influence of other factors like recourse abundance or crowding. This feature is crucial to building reliable industrial district models that would be mentioned in the next section. Another good example could be taken from the Gimbett (2002), where movement of backpackers in the Sierra Nevada Mountains in California is modeled to assess the impact of different management policies on saving this area of the wildlife. In this model, the agents simulated moving in a landscape representing the topology of the area.

Heterogeneous Agents. It is common knowledge that traditional approaches usually deal with unified actors. In other words, all considered actors are similar in their most important traits. For example, economics usually deal with the "typical firm" or "typical economic agent", whose decisions are rational. These agents may differ in their preferences in a certain models, but follow the same behavior. Even if there are different sets of actors with their own behavior, there are small number of them, mainly because if agents are not homogenous the analytical conclusions could be impossible to derive. The ABMs overcome this constraint allowing each agent to perform according to its own preferences and rules. In this case, models of the supply chains could be an example, since each business in them has its own strategy.

Agent Interactions. One of the most important benefits of the ABMs is a possibility of interactions between the agents. In other words, in this type of models agents could transmit information between each other, it could be as simple as just passing the messages between two neighbor agents, or could be much more complicated and involve different interpretations of the single message. The good illustration of this feature are opinion dynamics models.

Bounded Rationality. Usually in traditional economic models individuals acting in the market are assumed to be rational. This could be considered a weakness of such kind of models, since many researchers assumed individuals to be “hyperrational” following complex or even infinite chains of logic in order to select the optimal course of action, which is obviously unrealistic. For the sake of truthfulness it should be mentioned, that there are models with agents acting irrationally or randomly and will not maximize their welfare, but number of such models is relatively small. As an alternative to the prevailing complete rationality, Herbert Simon (1957) proposed that individuals should be modelled to have bounded rationality, in other words, have limited optimization abilities due to limited cognitive abilities. Agent-based models not only allow limiting the rationality of agents, but also, potentially allow agents to make decisions of the same level of sophistication as humans. Several models of stock markets could be mentioned as a good example of this feature.

Learning. Another useful feature of ABMs is that they allow modelling the learning process of an agent both on individual and community levels. Such models as innovation networks (that would be mentioned in the next section) allow firms on the market learning how to produce more profitable and desired product. Moreover, sectors at whole could learn how to operate effectively and products of the firms in the sector would produce the compatible set, do one firm would by components for own production from the other firms. There are three most well-known approaches to modelling the learning process:

— Individual learning, which implies that each agent learns from its own experience separately from the others.

— Social learning, in which some agents imitate others, or either taught by them. This leads to the sharing of the individual experience among the whole population.

— Evolutionary learning. This type of learning assumes that some agents withdraw from the market and are substituted by more competitive agents, which teaches other agents how to operate successfully.

Naturally, this types of learning could be combined if needed. For example, afore-mentioned innovation networks employ the following mechanism of learning: The individual innovating firms learn how to make better products, and because poorly performing firms become bankrupt to be replaced by better start-ups, the sector as a whole can learn to improve its performance.

As we overviewed the concept of the agent-based modelling and considered its main distinctive features, let us now introduce several useful examples of ABMs to make definitions more concrete and clear, also showing possible ways of implementation of agent-based modelling in the real researches.

4.2. Several Examples

ABMs could be implicated in the number of social sciences. This section gives some illustrations to the diversity of problems where ABMs could be used effectively.

Urban models. Initially proposed in 1971 by Thomas Schelling (1971, 1978) the first urban model was aimed to explain observed racial segregation in American cities. Initially proposed, this model was very abstract, thou had great impact on the following works regarding the persistence of segregation not only in the urban centers all over the world. The model consists of a square grid of cells, which represents an urban area where agents-households are randomly distributed. There are two kinds of these agents, usually called ‘reds’ and ‘greens’ for simplicity. Each cell of the grid could have only one household a time, thus some cells are left empty. The algorithm performs following steps:

1. At each step, each household gathers information about their eight neighbors.
2. The household checks if fraction of neighbor households with different color is greater than a certain level of ‘tolerance’.
3. If the fraction is grater then the tolerance level (e.g. there are 6 red households surrounding green household with 0.5 tolerance value) the household considers itself to be unhappy and moves to the empty cell.
4. Alternatively, the ‘happy’ household stays at its cell until the next step.

Due to households moving every time the named fraction for each household constantly changing, making them reevaluate their decisions at each step, which results in further relocations. Figure 4.1 provides illustration of how initially random distribution (left panel) of households reorganizes itself into solid clusters of red and green (right panel).

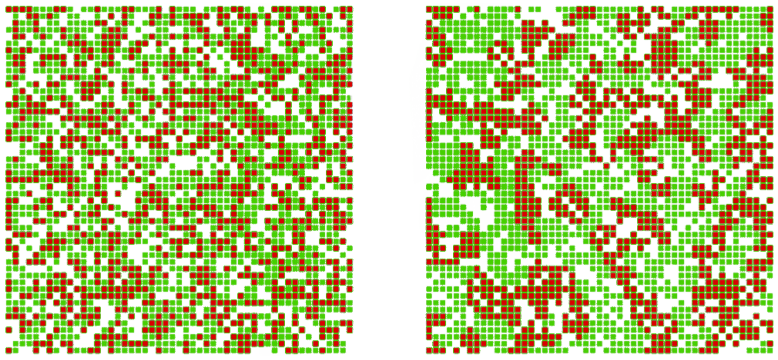


Fig. 4.1. Performance of the Schelling model (tolerance threshold = 0.3) (Source: Gilbert, 2007)

Clustering even occurs when households tolerate the majority of neighbors with different color, which was stated as an indication that even high level of racial tolerance allows forming of the strongly segregated patterns typical for U.S. cities.

According to Allen (1997) there are several main reasons why this model has been so influential. First, the outcome is not easily predicted just knowing the movement rule of the each agent. Second, the model itself is simple, depending on only one parameter, which is the tolerance value. Therefore it is easy to understand. Third, resulting behavior is robust for the different values of the tolerance threshold and initial placement. The same results are obtained for a wide range of the parameter values, movement rules, and considered neighbors. As proposed by Gilbert (2002) the agent could make random movement decisions, consider utility of the unoccupied cell, or take into account the price of the cells, and other scenarios. Forth, the model suggests how it could be tested with the empirical data, however it have been proven rather problematic in practice, due to difficulty of obtaining reliable data and measuring the "unhappiness".

So generally, this model had a huge impact because of its simplicity in both results and the composition. Nowadays this model is being updated by adding more parameters and modified to be more suitable for the real world.

Opinion dynamics models. Second group of models to consider are important in understanding how political or any other opinions develop among the population.

There is a number of works considering this problems, but we will present only one of them. One of the most interesting questions in this field of research is probably how marginal or even extreme opinions could become a norm for the most of the population, which is considered by Deffuant et al. (2002). There are a lot of historical examples from all over the world illustrating this phenomena. This phenomena shows itself in the common fashion, where first “extremist” way of dressing, becomes then socially acceptable and widespread, like wearing miniskirts for example. Unfortunately, this phenomena could be much more dramatic. Similar processes took place in the Germany of 1930-ies, or in the Middle East of the last decades, where radical minority succeeded to convince a large share of the population in their political beliefs. There are also many examples of strong bipolarization of the population, when the whole population becomes extremist. One could name Europe of the Cold War period, or a number of civil wars in different countries.

In the model proposed by the aforementioned researchers, agents have an opinion, which is a real number between –1 and 1, and a certain degree of doubt about their opinion. These parameters together form an opinion segment of an agent, which is presented as a band, centered on agents’ opinion and spreading by the degree of uncertainty. The model then consists of the following steps:

1. Every agent starts with an opinion generated randomly from the uniform distribution and with a common level of uncertainty, except small amount of extremists, who are rather certain about their opinion.
2. Agents meet each other randomly.
3. Upon meeting, one agent (***i***) can influence opinion of the other one (***j***) if their opinion segments overlap. Otherwise they are considered too different to influence each other.
4. If one agent does influence other one then their opinion is affected proportional to the difference between their opinions, multiplied by the amount of overlap divided by agent i’s uncertainty, minus one. This indicates that individuals strong in their beliefs affect others more than uncertain agents.

Following these steps extremism spreads until all agents join extremism, or every agent becomes “traditionalist”. If the simulation is ran without extremists, then all population share the middle view. Therefore this model gives a valuable example of how small group of extremist individuals with strong beliefs could have dramatic effect on the society at whole.

Consumer behavior models. This group of the models are usually employed by businesses to better understand what exactly motivates their customers to buy their products, since social factors, like the influence of friends and families, advertising, fashion, are important, but could be challenging to estimate. Fortunately, it could be done with a use of the agent-based models. One of the first researchers, who used this approach were Janssen and Jager (1999), who explored situation in which one of the competing products on the market achieves such dominance that it is difficult for individual consumers to switch to rival products (this situation is usually called “lock-in”). There are numerous examples of such situation like QWERTY keyboards (dominating over other keyboards), VHS, Gameboy consoles, or Windows operating systems. The named researchers mainly focused on the behavioral process that leads to lock-in, thus they constructed a model with “consumats” (agents) having certain decision rules that are tailored to fit common behavioral theories of social comparison and imitation.

Another example of this kind of models is the study of the “lemon” (second-hand cars) market and other markets with a quality variability and quality uncertainty by Izquierdo and Izquierdo (2006). The study demonstrates how different quality for different items could affect agents’ confidence and the market at whole. Researchers introduced two types of agents in the model: buyers and sellers. Market contains a finite number of products that buyers and sellers operate with, performing one deal per round, and the market is cleared every round as these deals are done. Sellers are acting based on their minimum price to accept, whereas buyers are offering payment based on the expected quality of the product. Buyers draw their conclusions about the quality based on the individual of social experience. The social network is created by the means of randomly connecting pairs of actors at a certain rate that is controlled by an initially set parameter.

Results of this research showed the importance of social connections for this kind of markets. If buyers have to rely only on the individual experience the consumer confidence falls and market collapses, whereas social network sharing the experience allows to maintain the stable market. Also this shows how social information can aggregate group experience to a more accurate level and thus reduce the importance of a single individual’s bad experiences.

Industrial networks models. Previously, economic theory did not pay much attention to the established links between firms, but in the last decades researchers come to understand that networks are extremely important in the industrial sectors. For example there is a well-known example of the industrial districts in the Northern Italy, like Prato, the industrial district. The main characteristic of the industrial district is that it contains large number of small firms in the small geographical region that produce the same product and have strong connections between each other. These connections are represented differently in the number of the researches, they could be supplier/consumer relations, sharing production techniques, financial connections, or even friendship relations.

Another popular example is innovation networks that could be seen in the knowledge-intensive sectors like computer technologies. These networks could be a bit different from the standard industrial districts, since they do not require geographical proximity of the firms in the sector. Apart from that connections are the same. Let us consider the example proposed by Gilbert et al. (2001) who developed a model of innovation networks in which agents have certain stock of knowledge and expertise, called “kenes”. These kenes are used to create new products that are then sold to other firms in the environment. However, a product can only be produced if components for it could be purchased from other firms, and there is demand for this product on the market. Fortunately, firms can improve their kenes to improve their products and market performance either through individual research and development activities or through incorporating knowledge obtained from other firms by collaborative arrangements. There are two levels of the model. First is the industrial market with firms trading with each other. Second is the model where agents learn through the natural selection, where firms that cannot find a customer cannot produce anything and withdraw from the market, whereas the most competitive firms survive, collaborate with other firms, and produce "child firms" that incorporate the best traits of the "parents".

This kind of models is useful in the field of production optimization and stimulating economic and innovative growth by creating clusters.

Supply chain management. Often product before getting to the user travels a long rout from suppliers to distributors, which implies presence of many strong connections between a number of companies just to produce and sell one product. Therefore optimization of the supply chain is more and more important and complex, since products now (especially in the technology-intensive sectors) require more components from all over the world. Modeling supply chains is an effective way of studying effectiveness of management policies, for this purpose multi-agent models are becoming increasingly popular. The main reason why a multi-agent model is the most suitable for this task is that it provides possibility to assign each business involved in the chain with an agent with own behavior rules. Moreover this model makes easy to imitate the flow of information and products along the line.

Let us now present on of the researches in this field undertaken by Strader, Lin, and Shaw (1998), who studied impact of the information sharing on divergent assembly supply chains, which are typical of the electronics and computer industries. Divergent assembly supply chains are characterized by a small number of suppliers producing generic subcomponents (e.g. transistors or controllers) that are used to produce a range of generic products (e.g. micro schemes, CPU, hard drives) that are then used to assemble customized products like personal computers at distribution points. The model considers three different order fulfillment policies:

1. *Make-to-order*, when production is triggered by customer orders.
2. *Assembly-to-order*, when components are built and held in stock, and only the final assembly is triggered by an order.

3. *Make-to-stock*, when production is driven by the stock level falling below a threshold.

Moreover, authors explored how different amount of information sharing between the agents affects the model.

The results showed that assembly-to-order approach together with the sharing of supply and demand information between organizations tied in the supply chain is the most efficient model. As authors mentioned, their results provide proof for the idea that information could substitute the inventory. In other words, if a producer has reliable and sufficient information, the uncertainty about demand could be reduced, allowing optimization of the held inventory.

Electricity market models. In a number of developed countries electricity supply is privatized. Usually electricity market consists of two or three major power suppliers who sell electricity to the distributors who then sell it to the consumers. Since this structure is different from the traditional state-owned electricity industry, it created an area for exploration using ABMs. Usually these models aim to understand the effects of market regulations; changes to the number and type of suppliers and purchasers; and policy changes intended, for example, to reduce the chance of blackouts or decrease the environmental impact (Bagnall & Smith, 2005; Batten & Grozev, 2006; Koesrindartoto, Sun, & Tesfatsion, 2005).

Generally, in this type of models agents are supply companies that offer market to provide it with a certain quantity of electricity at a certain price for a certain period. Which is similar to the real market where the best of such offers is accepted. However, the market situation is complicated by several factors. First, demand for electricity changes continuously, which complicates process of setting desired price by the suppliers. Second, connected to the first one, is that the cost of generating the power is very nonlinear, since only peak demand lasting for the several hours might require the full output, during the other time there is no point in producing the electricity on all stations or at full power.

There is a number of useful results that could be obtained by running the simulation, for example, what is the optimal number of suppliers for the market with given parameters, or under which conditions price of the electricity comes down to its marginal cost of generation, or how different market designs affect the producers. Lately this type of models have been enhanced with a feature allowing agents to learn trading strategies by increasing chance of using strategies that have been proven to be successful and reducing it otherwise.

Participative (companion) modeling. Another surprising area where ABMs could be used is management of scarce natural resources such as water or woods. This approach involves building a multi-agent system in close association with informants selected from the people living in the problematic area and much more practical-oriented then other approaches. Participants are usually interviewed on the object of the problematic situation, and then engaged in a specially tailored roleplaying game (RPG) to collect the data. After all needed data has been collected, researchers build a computer model that is used as a training aid or as a negotiation support, allowing the answering of “what-if” questions about possible decisions concerning the problem of scarce natural resources.

As an example one could mention the work of Barreteau et al. (2001) that describes how participative modeling has been used in order to understand why an irrigation scheme in the Senegal River Valley has not been successful in general. Researchers created a RPG and a multi-agent ABM called SHADOC to recreate the interactions between the various stakeholders involved in making decisions concerning the allocation of water to arable plots in the irrigated area. The ABM was developed first and then its main elements were converted to the roleplaying game, where players were equivalent of the agents, aiming to validate the model and ease usage amongst the participants. Authors state that their approach is “enhancing discussion among game participants” and mention as a success that “the problems encountered in the field and known by each individual separately turned into common knowledge.”

History-friendly models

Another type of models that fall under the broad class of ABMs, but have a several significant differences from the other types (and thus need more detailed overview), are “History-friendly” models. They, as many other simulation models, are able to imitate complex systems in terms of generating interactions that are not encoded. These interactions allow one to study the whole process relying on its parts. As stated in Garavaglia (2010), History-friendly models are stand-alone simulation models, precisely specified and formally structured, which build upon the definition of ‘interacting objects’. Where objects could represent both, agents and the environment, defined by their characteristics. This type of models is not aimed at prediction, but rather at understanding and explanation how certain industry behaves. It represents the routines and behavior of economic agents regarding evolutionary mechanisms of the industry, like emerging of the new technologies or demise of some companies. This models usually answer to questions like “What might have happened if …?” and “Why exactly that happened?” There are three main points of interest for the history-friendly models that are traditionally outlined:

- Explaining from the history perspective how current phenomena are possible to observe.
- Researching rules and conditions that make an event happen under certain limitations.
- Explaining the conditions and the deterministic explanations behind the “typical” history that occurred.

The earlier versions of the evolutionary models usually had a bit too simple structure and their empirical basis was given by very broad phenomena. In order to overcome this problems history-friendly models try to adopt more complex structure and limitations on the empirical basis. These models try to combine on the one hand formalized relationships of the agents and historical events on the other. They usually consider a specific industry or a market, while other evolutionary models could be applied to broad range of industries.

One should keep in mind, that the goal of this approach is not to reconstruct socio-economic properties in detain, but rather investigate historical stylized facts. Structure of history-friendly models is essentially a laboratory construction of the artificial industry with artificial agents and market environments, which is made in order to explain certain phenomena. This also allows to call history-friendly models industry-specific, since the phenomena is investigated only in the given industry. The analysis in this case consists mainly of testing the behavior of the model in order to understand what factors and fundamental processes make the model act as it does. For this testing to be successful and produce reliable results researcher should generate and then sample data so it would be a good approximation to the real one. However, history-friendly simulations do not aim at reproducing results that empirically match the quantitative specific values of available historical records perfectly, they rather should be close to the real data in a qualitative sense. First of all, model should be logical and consistent to fulfill its' main goal. However, if results of model are close to reality it is usually considered a good indication. Thus history-friendly models investigate the aspects of the chosen phenomena, both from a quantitative and a qualitative perspective.

History-friendly approach makes an extensive use of so-called counterfactual analysis (alternatively known as if-worlds, allohistory, cliometrics, uchronia). History-friendly models may even be considered a normative tool to investigate what effect do different socio-economic settings and industrial public policies have as a support for decision makers. In general, this analysis consists of "changing" previous event to investigate effects on the consequent situation, thus evaluating how some changes would alter the course of actions. This analysis is known for a long time and usually used in policy evaluation to see the effects of introduction of certain policy. Although the debate on the effectiveness of counterfactual analysis is still open, since it is not always easy and appropriate to implement it, it is clear that "good" counterfactual investigations are informative, beneficial for the research, and allow escaping some social biases (e.g. cultural). For counterfactual investigation to be good, it should be presented in simple and understandable way, but with detailed reasoning why considered changes are plausible and why they did not occur initially.

As for history friendly models, counterfactual investigations could be used both as a testing procedure and an investigation tool. They have proven to be a good test for the consistency of the logic of the model and for the robustness of the results. This kind of analysis could be viewed as a reconstruction of alternative scenarios that allows to discuss the evolution of the given industry.

This industry-based and quality-focused that heavily relies on history genuinely have been considered to represent the "fall of one particular leaf from a tree", as described by Silverberg, thus criticized because they model a "typical" history, which may not exist in reality, or even just be a coincidence. As a contradiction to this point we should mention that historical events used in the models are usually result of causal and systematic explanations, rather than a random occurrence. This is also the reason why history-friendly models should be carefully tested and calibrated, sometimes even using extreme scenarios, where the outcomes are easily predictable.

Opposed to the common view on the history-friendly models, that we mentioned, it is suggested to try developing more general models that explain the "fall of many leaves from many trees". Probably, deeper understanding of the basic processes of different industries would allow to find common traits and rules among them, which would make generalizing possible. However, at the current state of history-friendly models it does not seem easy to perform that, thus further research in this field should be done.

Before continuing to the examples let us first present main points that should be covered when presenting history-friendly model (and many other ABMs in general):

1. Clear explanation of the environment used. This, on the one hand, allows better understanding of the model, but on the other, helps researcher a deep knowledge. This would also give the model framework more creditability.
2. An overview of the theoretical background, which could be presented in analyze of already existing researches and relevant theories. This point, again, serves the purpose of "reinforcing" the model and the phenomena considered. It is crucial to give clear explanation of how this theoretical background is incorporated into programming of the model. Apart everything else, here researcher also faces a trade-off between descriptive accuracy of the model and explanatory power.

3. An outline of the most relevant features of the model. This is, obviously, made for understanding of the models structure and functioning.
4. The fourth point is an overview of the results, which also includes results of the calibration process that consists of a repeated change in the parameters and methods of assessment. If counterfactual analysis is used, then it should be also outlined here. Each result should be made as clear as possible, so it could be used in further researches. Further prospects of the model could also be mentioned at this section.

Let us now consider an examples of practical use of such models. One could mention notable research of Malerba et al. (1999) considering evolution of computer industry. In the paper several key events of the computer sector are analyzed, such as emergence of the IBM as a leading firm of the mainframe market, introduction of the microprocessors, leading to the rising of the personal computers market segment. Initial model showed key factors in the evolution of this industry, which are the timing of the introduction of the new technology, the bigger consumer lock-in for the mainframe market as compared to the PC market, and the higher price sensitivity of consumers in the PC market.

In the subsequent research Malerba et al. (2001) enhanced the initial model to consider efficiency of the economic policies in the complicated market environment such as computer industry with dynamic increasing returns and high level of lock-in. This addition made the model more complex since it considered not only companies and consumers, but also other public or private institutions. Definitely, such addition made the model more complete and approached the reality closer.

This models could also be used to examine the effects of new innovation approaches in the market. The work by Malerba and Orsenigo (2002) gives example of the pharmaceutical industry moving from era of random screening to the molecular biology era. This process involved changes in approach to development of new drugs, but also changes in the completion and market structure. This research showed that given the nature of the processes of drug development, the main features of the competition process, the low degree of cumulativeness and the fragmented nature of the markets in the given industry, a level of overall concentration of the industry soon becomes rather low. In this case, the advent of biotechnological firms does not represent a displacement of the core ones.

Another thing to consider as a conclusion is prospects of the agent-based modelling and how it could evolve. For example, Frenken (2006) gives example of four possible directions of development for the innovation models, but the same could be said about all other ABMs.

1. Applying core models on the topics not yet extensively researched with this approach, such as economic geography, evolutionary macroeconomics, CSR, and many others.
2. Recombining core approaches. For example combining supply chain models with industrial networks allows to gain interesting information on how supply chains affect clustering of the companies and vice versa.
3. Recombining ABMs with earlier evolutionary models of the industry dynamics.
4. Empirical testing of core models. This, however, could be done in several ways. In some applications, hypotheses can be derived deductively as outcomes of the model. Although, in the most situations empirical validation of ABMs follows a different methodology based on a model's capacity to replicate stylized facts, which we discussed in the end of Section 4.1.

To sum up it should be mentioned, that this of different approaches does not cover every existing implementation of the agent based modeling but merely shows that this method of modelling provides a lot of possibilities for researchers to analyze and test social and economic and environmental environments.

Exercises

1. Think of any other application of the ABMs. Why exactly could they are appropriate?
2. (based on Axelrod, 1997) A) Consider a Schelling model with 64 cells and 40 agents. Assume there are 20 greens and 20 reds. Let the greens be

satisfied if less than 2/3 their neighbors are reds, at the same time let the reds be satisfied if at least half of their neighbors are reds. Run the model for 100 cycles. Then for each of the 24 empty cells, write an A in the cell if more of its neighbors are greens than reds, and write a B in the cell if more of its neighbors are reds than greens. Then see if the greens enjoy a greater expanse of free space by seeing if there are more A's than B's. B) Do ten restarts of the model 2.A, then construct a histogram to analyze the distribution of the results.

3. (based on Axelrod, 1997) In a Schelling model with 64 cells consider 30 Reds and 10 Greens, both with the original (and equal) requirements for contentment. For density studies, use the A and B measures described for the previous exercise. You should arrange to stop a run when no further change is possible. For example, by using periods of at least 40 events, and checking whether there has been no movement in the current period since that would imply that everyone is content and no will ever move again. Answer the following questions:
    1. Do minorities get packed in tighter than majorities?
    2. Does the process of moving to the optimal positions settle down faster when the numbers are unequal?
4. Consider an opinion dynamics model as explained in Section 4.2. Create the model with 400 agents interacting, where 80 % of agents are conformist with degree of uncertainty at 0.4 and other 20 % are extremists (have opinions close to either 1 or –1) with uncertainty at 0.1. Model agent interaction as described in the book with 450 iterations (Deffuant et al. (2002) could be useful). Present results as a graph. What results are showing? Consider general level of uncertainty at 1.0; at 1.4. How results change?

SUMMARY

This book takes in consideration several important aspects of the methods Monte Carlo (MMC), since they allow to solve a wide range of problems that are out of scope of traditional approaches. The main advantage of MMC is that they allow instead of simplifying the models or using highly specified approaches, employ more general Monte Carlo algorithms and obtain reliable results. Considering usual simplicity of these algorithms, which we underlined several times in the book, Monte Carlo methods appear to be an ultimate tool that could provide surprising results in complicated problems.

Following the course of the book we overviewed several main concepts and approaches using ideas of methods of Monte Carlo. Apart from that we showed that MMC could be applied to different problems. Let us summarize notable points of each chapter in few words.

The first part showed that the concept of Monte Carlo approach first appeared even before computer technologies and over the years had proven to be effective as an alternative method for solving computational problems. As we stated, MMC are essentially group of computational methods relying on random events or numbers. This chapter also demonstrated that numbers generated by RNGs are not random in the strictest sense, but rather generated by complex algorithms that by itself could be subject for further modification. We also give several examples that include estimating value of π, gambling problem, and coupon collecting problem. That indeed shows usefulness of applying Monte Carlo methods to the problems of general mathematics and probability theory. Therefore this also opens great possibilities of implementation in computer and natural studies.

The second part, mainly dedicated to the explanation of the sampling concept, clearly showed that sampling is exceptionally useful in the statistical field. Some of the most common distributions are, indeed, built-in in the programming software. However, it usually requires creative approach since not every distribution is not sampled from with ease. In this case Monte Carlo approach are used, allowing to sample from virtually any distribution. This part contains two examples of possible ways of sampling uncommon distributions using MMC. They show how this approach could be used in the field of applied statics and possibly econometrics.

The third part is devoted to the optimization heuristics. Optimization is one of the core ideas in spheres, such as economic theory, management, logistics, and many others. It is mentioned that heuristics usually mimic some natural phenomena, like annealing, evolution, or navigation mechanism of ants. We presented several core heuristics, such as simulated annealing, threshold accepting method, tabu search, differential evolution, genetic algorithm, and ant colonies. We also provided several example of optimization heuristics for the discrete and continuous search spaces. In addition, this part provides reasons why heuristics (and usually Monte Carlo approach in general) are usually underused in researches. It is usually lack information on either how to use heuristics or when should this approach be used.

The last part is focused on agent-based modelling approach. Which is modelling approach that involves creating different agents with a certain behavior within given environment to see how they interact. We also underline that this kind of models allow experimenting with social systems, thus providing valuable insight on how they work and factors that affect them. The part also presents several guidelines for constructing ABMs, such as, for example, KISS criterion that requires these models to be as simple and understandable, as possible. In the second section several core agent-based models are presented along with their implementation in researches of the last 20 years. That includes electricity market models, supply chain models, urban models that demonstrate how citizens segregate, opinion dynamics models that demonstrate how small group of people with extreme opinions and strong beliefs could convince the whole community, and some others. We also extensively focused on the history-friendly models that allow understanding evolution mechanism of the chosen industry. This approach, as we mentioned, is highly relevant for such spheres as economics, management, sociology, biology, and others.

This, indeed, shows that Monte Carlo methods could be used in wide range of studies and applied researches, from physics and biology to social studies and opinion theories. This range also includes economics and econometrics, where Monte Carlo methods could be used to provide information about complex systems that include many economic agents and/or many significant factors affecting the system.

We also mentioned several challenges that Monte Carlo approaches are facing. The main one is usually that researchers in many fields are not sure when and how to use it properly, and rather rely on traditional approaches simplifying the model too much sometimes. Secondly, the "randomness" of the approaches is considered to be a major drawback, especially in econometrics. Both this problems could be solved mainly by popularizing MMC and developing easy and understandable ways of its implementation. We also suggest that Monte Carlo methods to be used as a tool to evaluate the results of traditional approaches, which would make choice of the main evaluation method between these approaches easier. Another important problem is that with development of computer technologies, that once made Monte Carlo methods viable, more advanced and effective evaluation methods have emerged. And rising computational power of personal computers allows more problems to be solved traditional way. This problem, obviously, could not be solved completely, but developing new Monte Carlo approaches and modifying old ones could keep MMC effective and competitive.

For the further study of the Monte Carlo methods there are several main roads to take. One is to consider theoretical framework of more complicated Monte Carlo approaches, for example Markov chain Monte Carlo, since they are more suitable for complex researches and implementations that involve high-dimensional distributions. This will provide more detailed image of nature of Monte Carlo approaches, allowing one to understand where and when they could be applied. Another is to research successful implementations of the Monte Carlo approach. We presented some of them that more or less related to the core approaches but there are a huge number of other researches, which would provide an insight about exactly how Monte Carlo methods could be applied in a given situation. Alternatively, one could explore possible combinations of already existing models to construct new ones. This will also allow studying existing models in more detail.

To conclude it should be mentioned that Monte Carlo methods have already proven themselves to be effective and have been used in the numerous researches, but they are constantly developing and, hopefully, would be even more frequently used further on.

REFERENCES

1. Aarts, E. and J. Korst (1989). Simulated Annealing and Boltzmann Machines. J. Wiley and Sons. Chichester.
2. Acosta-Gonźalez, E. and F. Ferńandez-Rodŕıguez (2007). Model selection via genetic algorithms illustrated with cross-country growth data. Empirical Economics 33, 313–337.
3. Adanu, K. (2006). Optimizing the GARCH model – an application of two global and two local search methods. Computational Economics 28, 277–290.
4. Ahn, S.K. and G.C. Reinsel (1990). Estimation for partially nonstationary multivariate autoregressive models. Journal of the American Statistical Association 85(411), 813–823.
5. Alcock, J. and K. Burrage (2004). A genetic estimation algorithm for parameters of stochastic ordinary differential equations. Computational Statistics and Data Analysis 47(2), 255–275.
6. Allen, P.M. (1997). Cities and Regions as Self-Organizing Systems: Models of Complexity. Gordon & Breach. Amsterdam.
7. Althöfer, I. and K.-U. Koschnik (1991). On the convergence of threshold accepting. Applied Mathematics and Optimization 24, 183–195.
8. Ashlock, D. (2006). Evolutionary Computation for Modeling and Optimization. Springer-Verlag. New York.
9. Atkinson, A.C. and S.W. Weisberg (1991). Simulated annealing for the detection of multiple outliers using least squares and least median of squares fitting. In: Directions in robust statistics and diagnostics, Part I (W.A.Stahel and S.W. Weisberg, Ed.). Springer-Verlag. New York.
10. Axelrod, R. (1997). The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration. Princeton University Press. Princeton, New Jersey.
11. Bäck, T., D. Fogel, Z. Michalewicz (1997). Handbook of Evolutionary Computation. Oxford.

12. Bagnall, A.J., G.D. Smith (2005). A Multiagent Model of the UK Market in Electricity Generation. Evolutionary Computation, IEEE Transactions on 9 (5), 522–536.
13. Baragona, R., F. Battaglia and C. Calzini (2001). Genetic algorithms for the identification of additive and innovation outliers in time series. Computational Statistics & Data Analysis 37(1), 1–12.
14. Baragona, R., F. Battaglia and D. Cucina (2004). Fitting piecewise linear threshold autoregressive models by means of genetic algorithms. Computational Statistics & Data Analysis 47, 277–295.
15. Barreteau, O., F. Bousquet, J.-M. Attonaty (2001). Role-Playing Games for Opening the Black Box of Multi-Agent Systems: Method and Lessons of its Application to Senegal River Valley Irrigated Systems. Journal of Artificial Societies and Social Simulation, Vol. 4, No. 2, http://jasss.soc.surrey.ac.uk/4/2/5.html.
16. Batten. D., G. Grozev (2006). NEMSIM: Finding Ways to Reduce Greenhouse Gas Emissions using Multi-Agent Electricity Modelling. In: Chapter 11 of Complex Science for a Complex World: Exploring Human Ecosystems with Agents (P. Pascal and D. Batten, Eds.), ANU Press.
17. Bauer, D. and M. Wagner (2002). Estimating cointegrated systems using subspace algorithms. Journal of Econometrics 111, 47–84.
18. Birattari, M., L. Paquete, T. Stützle and K. Varrentrap (2001). Classification of metaheuristics and design of experiments for the analysis of components. Technical Report AIDA-2001-05. Intellektik, Technische Universität Darmstadt.
19. Blüschke, D., V. Blüschke-Nikolaeva and I. Savin (2013). New insights into optimal control of nonlinear dynamic econometric models: application of a heuristic approach. Journal of Economic Dynamics and Control 37(4), 821–837
20. Bock, F. (1958). An algorithm for solving "traveling salesman" and related network optimization problems. 14th ORSA meeting, St. Louis.
21. Boden, M.A. (1988). Computer models of mind. Cambridge University Press.

22. Bratley, P., B. Fox, and L. Scrage (1983). A Guide to Simulation, Springer-Verlag, New York.
23. Brooks, S.P. and B.J.T. Morgan (1995). Optimization using simulated annealing. The Statistician 44(2), 241–257.
24. Brooks, S.P., N. Friel and R. King (2003). Classical model selection via simulated annealing. Journal of the Royal Statistical Society Series B 65, 503–520.
25. Brown, B.W. and M. Hollander (1977). Statistics: A Biomedical Introduction. Wiley. New York.
26. Carley, K.M. (1999). On Generating Hypotheses Using Computer Simulations. Systems Engineering. Vol. 2, Issue 2, 69–77.
27. Chipman, J.S. and P. Winker (2005). Optimal aggregation of linear time series models. Computational Statistics and Data Analysis 49(2), 311–331.
28. Chung, K.L., A course in probability theory, Academic Press, New York, 1974.
29. Cohen, M.D., J.G. March and J.P. Olsen (1972). A Garbage Can Model of Organizational Choice. Administrative Science Quarterly 17 (1), 1–25.
30. Colorni, A., M. Dorigo and V. Manniezzo (1992a). Distributed optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life (ECAL-91) (F.J. Varela and P. Bourgine, Ed.). The MIT Press. Cambridge MA, 134–142.
31. Colorni, A., M. Dorigo and V. Manniezzo (1992b). An investigation of some properties of an ant algorithm. In: Parallel problem solving from nature, Vol 2. (R. Männer and B. Manderick, Ed.). North-Holland. Amsterdam, 509–520.
32. Croes, A. (1958). A Method for solving traveling salesman problem. Operational Research 5, 798–812.
33. Deffuant, G., F. Amblard, G. Weisbuch and T. Faure (2002). How Can Extremism Prevail? A Study Based on the Relative Agreement Interaction Model. Journal of Artificial Societies and Social Simulation Vol. 5 No. 4, http://jasss.soc.surrey.ac.uk/5/4/1.html.

34. Dorsey, B. and W.J. Mayer (1995). Genetic algorithms for estimation problems with multiple optima, nondifferentiability and other irregular features. Journal of Business and Economic Statistics 13, 53–66.
35. Dueck, G. and T. Scheuer (1990). Threshold accepting: A general purpose algorithm appearing superior to simulated annealing. Journal of Computational Physics 90, 161–175.
36. Eberhart, R.C. and J. Kennedy (1995). A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science. Nagoya, Japan, 39–43.
37. Edmonds, B. and S. Moss (2004). From KISS to KIDS – an ‘anti-simplistic’ modelling approach. CPM Report no. 04-132, http://cfpm.org/cpmreps.html.
38. Frenken, K. (2006). Technological Innovation and Complexity Theory. Econ. Innov. New Techn., Vol. 15(2), 137–155.
39. Gan, L. and J. Jiang (1999). A test for global maximum. Journal of the American Statistical Association 94(447), 847–854.
40. Garavaglia, C. (2010). Modelling Industrial Dynamics with “History-Friendly” Simulations. Structural Change and Economic Dynamics 21, 258–275.
41. Gilbert N. (2002) Varieties of Emergence. University of Chicago and Argonne National Laboratory Sociology: Agent 2002 Conference: Social agents: ecology, exchange, and evolution. Chicago, 41–56.
42. Gilbert N. (2007). Agent-based models. Sage Publications Inc.
43. Gilbert, N., A. Pyka, P. Ahrweiler (2001). Innovation networks – A simulation approach. Journal of Artificial Societies and Social Simulation, Vol. 4 No. 3, U131–U150.
44. Gillman, L. (1992). The Car and the Goats. American Mathematical Monthly Vol. 99, 3–7.
45. Givens, G and J. Hoeting. (2005) Computational Statistics. Wiley-Interscience. Hoboken. New Jersey.
46. Glover, F. and M. Laguna (1997). Tabu Search. Kluwer Academic Publishers. Boston, MA.

47. Goffe, W.L., G.D. Ferrier and J. Rogers (1994). Global optimization of statistical functions with simulated annealing. Journal of Econometrics 60(1-2), 65–99.

48. Goldberg, D.E. (1989). Genetic Algorithms in Search. Optimization, and Machine Learning, Addison-Wesley, Reading, Massachusetts.

49. Grimaldi, R. (2003). Discrete and Combinatorial Mathematics. Addison Wesley.

50. Hajek, B. (1988). Cooling schedules for optimal annealing. Mathematics of Operations Research, 13, No. 2, 311–329.

51. Hammersley, J.M., D.C. Handscomb (1964). Monte Carlo Methods. Wiley. New York.

52. Hawkins, D.S., D.M. Allen and A.J. Stromberg (2001). Determining the number of components in mixtures of linear models. Computational Statistics & Data Analysis 38(1), 15–48.

53. Holland, J.H. (1975). Adaptation in Natural and Artificial Systems. The University of Michigan Press. Ann Arbor, MI.

54. Hull, J.C. (2003). Options, Futures, and Other Derivatives, Fifth Edition, Prentice Hall, Upper Saddle River, New Jersy.

55. Hüsler, J., P. Cruz, A. Hall and C.M. Fonseca (2003). On optimization and extreme value theory. Methodology and Computing in Applied Probability 5, 183–195.

56. Izquierdo, S.S., L.R. Izquierdo, J.M. Galán, C. Hernández (2006). Market Failure Caused by Quality Uncertainty. Lecture Notes in Economics and Mathematical Systems, Vol. 564, 203–213.

57. Jacobson, S.H. and E. Yücesan (2004). Global optimization performance measures for generalized hill climbing algorithms. Journal of Global Optimization 29, 173–190.

58. Jacobson, S.H., S.N. Hall and L.A. McLay (2006). Visiting near-optimal solutions using local search algorithms. In: COMPSTAT 2006, Proceedings in Computational Statistics (A. Rizzi and M. Vichi, Ed.). Physica. Heidelberg, 471–481.

59. Janssen, M. and W. Jager (1999). An Integrated Approach to Simulating Behavioural Processes: a Case Study of the Lock-in of Consumption Patterns.

Journal of Artificial Societies and Social Simulation, Vol. 2 No. 2, http://jasss.soc.surrey.ac.uk/2/2/2.html.

60. Jerrell, M.E. and W.A. Campione (2001). Global optimization of econometric functions. Journal of Global Optimization 20(3-4), 273–295.
61. Kalos, M.H., Whitlock, P.A., Monte Carlo Methods, Wiley-Interscience, 1986.
62. Kirkpatrick, S., C. Gelatt, M. Vecchi (1983). Optimization by simulated annealing. Science 220, 671–680.
63. Knuth, D. (1997). Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd edition). Addison-Wesley Professional. Massachusetts.
64. Koesrindartoto, D., J. Sun and L. Tesfatsion (2005). An Agent-Based Computational Laboratory for Testing the Economic Reliability of Wholesale Power Market Designs. IEEE Power Engineering Society General Meeting 2005, 2818–2823.
65. Liu, J. (2001). Monte Carlo Strategies in Scientific Computing, Springer-Verlag. New York.
66. Maddala, G.S. and F.D. Nelson (1974). Maximum likelihood methods for models of markets in disequilibrium. Econometrica 42(6), 303–317.
67. Madras, N. (2002). Lectures on Monte Carlo Methods. American Mathematical Society, Providence.
68. Malerba, F., L. Orsenigo. (2002). Innovation and market structure in the dynamics of the pharmaceutical industry and biotechnology: towards a History-friendly model. Industrial and Corporate Change 11, 667–703.
69. Malerba, F., R.R. Nelson, L. Orsenigo, S.G. Winter (1999). History-friendly models of industry evolution: the computer industry. Industrial and Corporate Change 8, 3–40.
70. Malerba, F., R.R. Nelson, L. Orsenigo, S.G. Winter (2001). Competition and industrial policies in a History-friendly model of the evolution of the computer industry. International Journal of Industrial Organization 19, 635–664.
71. Marazzi, A. (1992). Algorithms, Routines, and S-Functions for Robust Statistics. Wadsworth & Brooks/Cole.

72. Maringer, D. (2005). Portfolio Management with Heuristic Optimization. Springer. Dordrecht.
73. Matlab Documentation. http://www.mathworks.com/help/matlab/.
74. Matsumoto, M. and T. Nishimura (1998). Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. ACM Transactions on Modeling and Computer Simulation Vol. 8, No. 1, 3–30.
75. Metropolis, N. (1987). The Beginning of the Monte Carlo Method. Los Alamos Science Special Issue 1987, 125 – 130.
76. Mladenovic, N. and P. Hansen (1997). Variable neighborhood search. Computers and Operations Research 34, 1097–1100.
77. Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical report. Caltech.
78. Niederetter, H., (1992). Random Number Generation and Quasi-Monte Carlo Methods. CBMS-NSF Regional Conference Series in Applied Mathematics 63. Society for Industrial and Applied Mathematics, Philadelphia.
79. Ohlmann, J.,J. Bean, S. Henderson (2004). Convergence in Probability of Compressed Annealing. Mathematics of Operations Research 29, Issue 4, 837–860.
80. Reeves, C.R. and J.E. Rowe (2003). Genetic Algorithms – Principles and Perspectives. Kluwer. Boston.
81. Rousseeuw, P. J. and K. Van Driessen (2000). An Algorithm for Positive-Breakdown Regression Based on Concentration Steps. In: Data Analysis: Scientific Modeling and Practical Application (W. Gaul, O. Opitz, and M. Schader, Ed.). Springer-Verlag, New York, 335–346.
82. Rubenstein, R.Y. (1981). Simulation and the Monte Carlo Method. John Wiley & Sons, New York.
83. Rudolph, G. (1997). Convergence Properties of Evolutionary Algorithms. Kovăc. Hamburg.
84. Salibian-Barrera, M. and V.J. Yohai (2006). A Fast Algorithm for S-Regression Estimates. Journal of Computational and Graphical Statistics 15, 414–427.

85. Schelling, T.C. (1971). Dynamic Models of Segregation. Journal of Mathematical Sociology 1, 143–186.
86. Schelling, T.C. (1978). Micromotives and Macrobehavior. W. W. Norton. New York.
87. Shonkwiler R.W. and F. Mendivil (2009) Explorations in Monte Carlo Methods. Springer. New York.
88. Simon, H.A. (1957). Models of Man, Social and Rational: Mathematical Essays on Rational Human Behavior in a Social Setting. John Wiley and Sons. New York.
89. Staszewska, A. (2007). Representing uncertainty about response paths: The use of heuristic optimization methods. Computational Statistics & Data Analysis 52(1), 121–132.
90. Steyvers M. (2011). Computational Statistics with Matlab, http://psiexp.ss.uci.edu/research/teachingP205C/205C.pdf.
91. Storn R. and K. Price (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359.
92. Storn, R. and K. Price (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359.
93. Strader, T.J., F.-R. Lin and M.J. Shaw (1998). Simulation of Order Fulfillment in Divergent Assembly Supply Chains. Journal of Artificial Societies and Social Simulation Vol. 1, No. 2, http://jasss.soc.surrey.ac.uk/1/2/5.html.
94. Taillard, E.D., L.M. Gambardella, M. Gendreau and J-Y. Potvin (2000). Adaptive memory programming: A unified view of metaheuristics. European Journal of Operational Research 135, 1–16.
95. Winker, P. (1995). Identification of multivariate AR-models by threshold accepting. Computational Statistics and Data Analysis 20(9), 295–307.
96. Winker, P. (2000). Optimized multivariate lag structure selection. Computational Economics 16, 87–103.

97. Winker, P. (2001). Optimization Heuristics in Econometrics: Applications of Threshold Accepting. Wiley. Chichester.
98. Winker, P. and D. Maringer (2004). Optimal lag structure selection in VAR and VEC models. In: New Directions in Macromodeling (A. Welfe, Eds.). Elsevier. Amsterdam, 213–234.
99. Winker, P. and K.-T. Fang (1997). Application of threshold accepting to the evaluation of the discrepancy of a set of points. SIAM Journal on Numerical Analysis 34, 2028–2042.

Учебное электронное текстовое издание

Савин Иван Валерьевич
Пушкарев Андрей Александрович

MONTE CARLO METHODS IN ECONOMICS
(МЕТОДЫ МОНТЕ-КАРЛО В ЭКОНОМИЧЕСКИХ ИССЛЕДОВАНИЯХ)

Научный редактор
Подготовка к публикации

Рекомендовано Методическим советом
Разрешено к публикации
Электронный формат –
Объем уч.-изд. л.

Уральский федеральный университет

620002, Екатеринбург, ул. Мира, 19

Информационный портал УрФУ
http://www.urfu.ru