Randomized Algorithms

Monte Carlo Algorithm

Farriha Afnan
Abrar Rahman Abir
Ananya Shahrin Promi

Department of CSE,
Bangladesh University of Engineering and Technology

February 20, 2024

Overview

Contents

Introduction

What is Randomized Algorithm?

A randomized algorithm is defined as an algorithm that receives a stream of random bits in addition to its input. It utilizes this stream for making random choices during its execution.

Las Vegas and Monte Carlo

Las Vegas Algorithm

It constitutes those randomized algorithms that always give a correct answer, or do not give an answer.

Monte Carlo Algorithm

It always answers, but may occasionally produce an incorrect answer. The probability of producing an incorrect answer can be made arbitrarily small by running the algorithm repeatedly with independent random choices.

Contents

Beyond the Casino's Games of Chance

The name *Monte Carlo* comes from the Monte Carlo Casino in Monaco, known for its games of chance and randomness.

Figure: Casino de Monte-Carlo

A Little History

- Stanislaw Ulam, recovering from an illness, was playing a lot of solitaire
- Tried to figure out the probability of winning but failed
- Thought about playing lots of hands and counting the number of wins but it would take years
- Asked Von Neumann if he could build a program to simulate many hands on ENIAC

Contents

Monte Carlo Simulation

About Monte Carlo

- A method of estimating the value of an unknown quantity using the principles of inferential statistics
- Inferential Statistics

  Population: a set of examples ○
  Sample: a proper subset of a population

An Example

If I were to flip a single coin an infinite number of times, what fraction of heads do you think I would get?

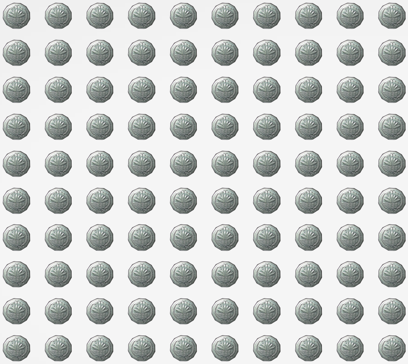
Consider One Flip

How confident should I be about answering 1.0?

Consider Two Flips

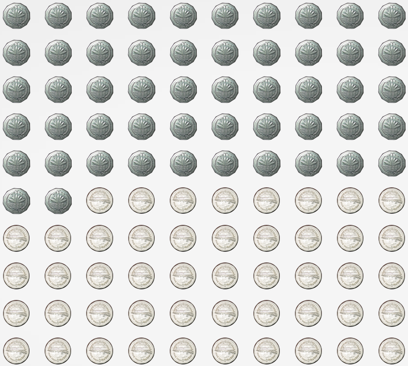
Can I predict that the next flip will result in heads?

Consider Hundred Flips



Now, can I guarantee that the next flip will come up heads?

Consider Hundred Flips



Is the probability of the next flip coming up heads 0.52?
Given the data, it is the best estimate. But confidence is low.

Difference in Confidence

Confidence in our estimate depends upon two things:

1. Size of sample
2. Variance of sample

As the variance grows, we need larger samples to have the same degree of confidence

Contents

Steps Performed for the Monte Carlo Simulation of a Physical Process

- Static Model Generation
- Input Distribution Identification
- Random Variable Generation
- Analysis and Decision-Making

Static Model Generation

Start with a deterministic model that closely resembles the real-world scenario being studied.

Static Model Generation

Use the most likely values (also known as the base case values) for the input parameters of the model.

Static Model Generation

Apply mathematical relationships to these input variables to transform them into the desired output.

Static Model Generation

Apply mathematical relationships to these input variables to transform them into the desired output.

This step involves creating a **simplified yet accurate representation** of the complex system or process in question, focusing on capturing its essential characteristics without the variability inherent in real-world scenarios.

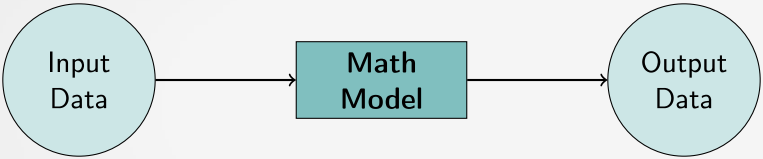
Static Model Generation

Apply mathematical relationships to these input variables to transform them into the desired output.

This step involves creating a **simplified yet accurate representation** of the complex system or process in question, focusing on capturing its essential characteristics without the variability inherent in real-world scenarios.

The goal is to establish a **reliable and validated foundation** upon which randomness and variability can be introduced in subsequent steps of the Monte Carlo simulation.

Static Model Generation



Input Distribution Identification

- Add the risk components
- The stochastic nature of the input variables
- Identify the underlying distributions if any, which govern the input variables
- Needs historical data for the input variables

Input Distribution Identification

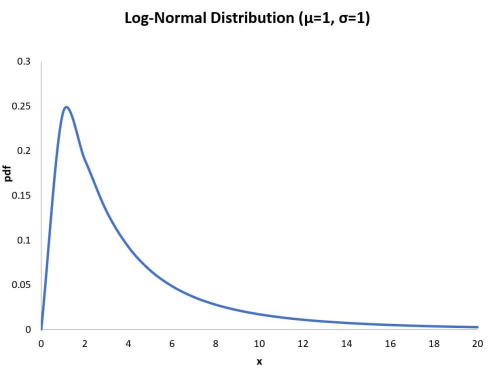


Figure: Log-Normal Distribution

Input Distribution Identification

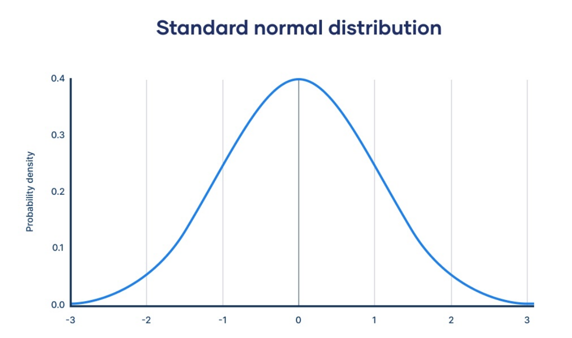


Figure: Standard Normal Distribution

Statistical Procedures to Identify Input Distributions

Method of Maximum Likelihood (ML)

- Sample data drawn is independent and identically distributed (iid)

Statistical Procedures to Identify Input Distributions

Method of Maximum Likelihood (ML)

- Sample data drawn is independent and identically distributed (iid)
- Let the sample drawn from the distribution be $x_1, x_2, \ldots, x_n$. Then the likelihood of getting the sample from the distribution is given by $L(\theta) = f\theta(x_1, x_2, \ldots, x_n|\theta)$

Statistical Procedures to Identify Input Distributions

Method of Maximum Likelihood (ML)

- Sample data drawn is independent and identically distributed (iid)
- Let the sample drawn from the distribution be $x_1, x_2, \ldots, x_n$. Then the likelihood of getting the sample from the distribution is given by $L(\theta) = f\theta(x_1, x_2, \ldots, x_n|\theta)$
- Given the independence of each of the data points, this can be expanded to the equation $L(\theta) = \prod_{i=1}^{n} f\theta(x_i|\theta)$

Statistical Procedures to Identify Input Distributions

Method of Maximum Likelihood (ML)

- Sample data drawn is independent and identically distributed (iid)
- Let the sample drawn from the distribution be $x_1, x_2, \ldots, x_n$. Then the likelihood of getting the sample from the distribution is given by $L(\theta) = f\theta(x_1, x_2, \ldots, x_n|\theta)$
- Given the independence of each of the data points, this can be expanded to the equation $L(\theta) = \prod_{i=1}^{n} f\theta(x_i|\theta)$
- Finding the value of θ so that the value of $L(\theta)$ can be maximized

Goodness-of-Fit Statistics

The correctness of fitting a dataset to a distribution.

- Chi-square Test
- Kolmogorov-Smirnov Statistic (KS)
- The Quadratic Statistics

Analysis and Decision Making

- **Calculating Trial Outputs:** the simulation model calculates trial values for the output variable(s).

Analysis and Decision Making

- **Calculating Trial Outputs:** the simulation model calculates trial values for the output variable(s).
- **Averaging Outputs:** Determining expected values by averaging trial outputs.

Analysis and Decision Making

- **Calculating Trial Outputs:** the simulation model calculates trial values for the output variable(s).
- **Averaging Outputs:** Determining expected values by averaging trial outputs.
- **Visualizing Distribution:** Creating frequency histograms to approximate the probability density function.

Analysis and Decision Making

- **Calculating Trial Outputs:** the simulation model calculates trial values for the output variable(s).
- **Averaging Outputs:** Determining expected values by averaging trial outputs.
- **Visualizing Distribution:** Creating frequency histograms to approximate the probability density function.
- **Empirical Distribution Analysis:** Using output values to calculate percentiles and other statistics directly.

Analysis and Decision Making

- **Calculating Trial Outputs:** the simulation model calculates trial values for the output variable(s).
- **Averaging Outputs:** Determining expected values by averaging trial outputs.
- **Visualizing Distribution:** Creating frequency histograms to approximate the probability density function.
- **Empirical Distribution Analysis:** Using output values to calculate percentiles and other statistics directly.
- **Fitting Theoretical Distributions:** Optionally fitting output values to known distributions to develop confidence bands.

Analysis and Decision Making

- **Calculating Trial Outputs:** the simulation model calculates trial values for the output variable(s).
- **Averaging Outputs:** Determining expected values by averaging trial outputs.
- **Visualizing Distribution:** Creating frequency histograms to approximate the probability density function.
- **Empirical Distribution Analysis:** Using output values to calculate percentiles and other statistics directly.
- **Fitting Theoretical Distributions:** Optionally fitting output values to known distributions to develop confidence bands.
- **Improving Precision:** More number of trials enhances the accuracy of expected values and distribution approximations.

Basic statistical analysis for output values

$$\textbf{Mean } \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\textbf{Median} = 50^{th} \text{ Percentile}$$

$$\textbf{Standard Deviation } \sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\textbf{Variance } \sigma^2 = \frac{1}{N-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

$$\textbf{Skewness} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})^3}{(N-1)s^3}$$

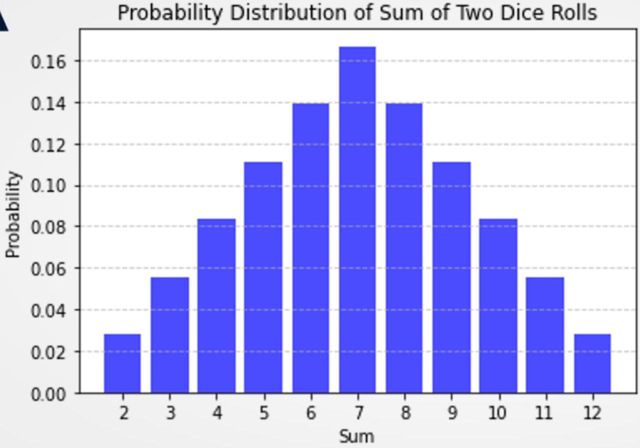
$$\textbf{Kurtosis} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})^4}{(N-1)s^4} - 3$$

Contents

Visualizing All the steps of Monte Carlo

```python
import numpy as np


die_prob = np.ones(6) / 6

# Convolution of two dice rolls
sum_prob = np.convolve(die_prob, die_prob)

plt.bar(np.arange(2, 13), sum_prob, width=0.8, color='b', alpha=0.7)
plt.title('Probability Distribution of Sum of Two Dice Rolls')
plt.xlabel('Sum')
plt.ylabel('Probability')
plt.xticks(np.arange(2, 13))
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

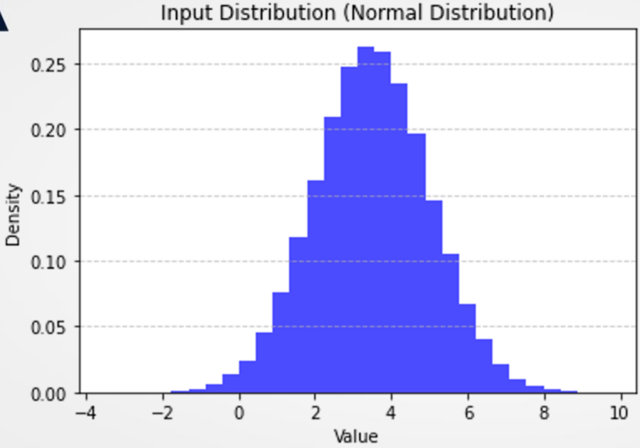
Visualizing steps of Monte Carlo



Visualizing steps of Monte Carlo

```python
# Step 1: Static Model Generation
def roll_dice():
    return np.random.randint(1, 7), np.random.randint(1, 7)

# Step 2: Input Distribution Identification (Using a normal distribution)

mean = 3.5
std_dev = 1.5

input_distribution = np.random.normal(mean, std_dev, 100000)

plt.hist(input_distribution, bins=30, density=True, color='b', alpha=0.7)
plt.title('Input Distribution (Normal Distribution)')
plt.xlabel('Value')
plt.ylabel('Density')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

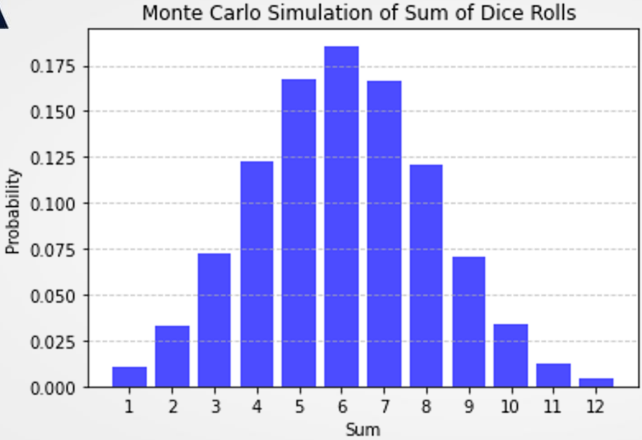
Visualizing steps of Monte Carlo



Visualizing steps of Monte Carlo

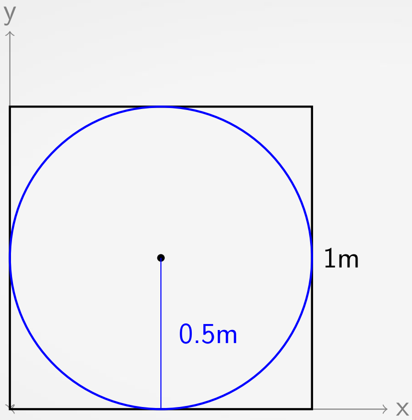
```
# Step 3: Random Variable Generation
num_simulations = 1000000
dice_rolls = np.array([np.random.normal(mean, std_dev, 2).astype(int) for _ in range(num_simulations)])


# Step 4: Analysis and Decision Making
sum_of_dice = dice_rolls.sum(axis=1)


plt.hist(sum_of_dice, bins=np.arange(1, 14) - 0.5, rwidth=0.8, density=True, alpha=0.7, color='b')
plt.title('Monte Carlo Simulation of Sum of Dice Rolls ')
plt.xlabel('Sum')
plt.ylabel('Probability')
plt.xticks(np.arange(1, 13))
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

Visualizing steps of Monte Carlo



Contents

A Simple Example - Calculate π



$$\frac{A_{Circle}}{A_{Square}} = \frac{\pi(0.5)^2}{1^2} = \frac{\pi}{4}$$

A Simple Example - Calculate π

Throw a dirt
(choose a random x and y value)

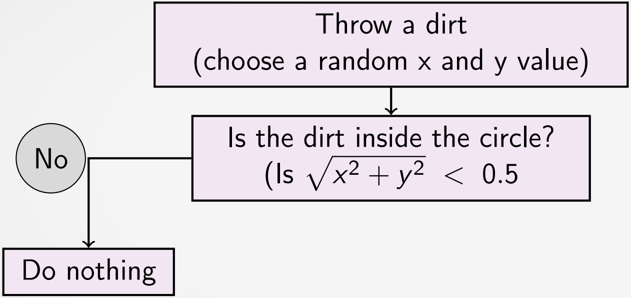
A Simple Example - Calculate π

Throw a dirt
(choose a random x and y value)

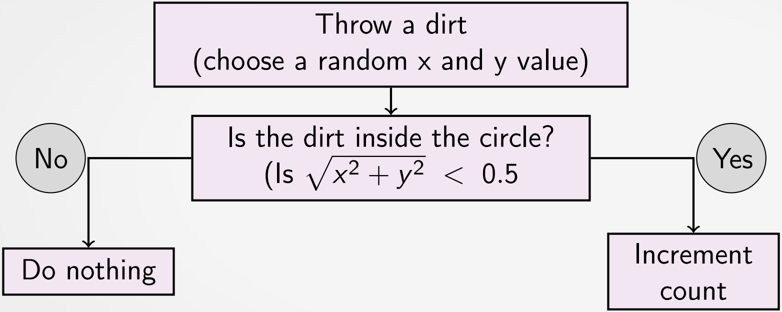
↓

Is the dirt inside the circle?
(Is $\sqrt{x^2 + y^2} < 0.5$

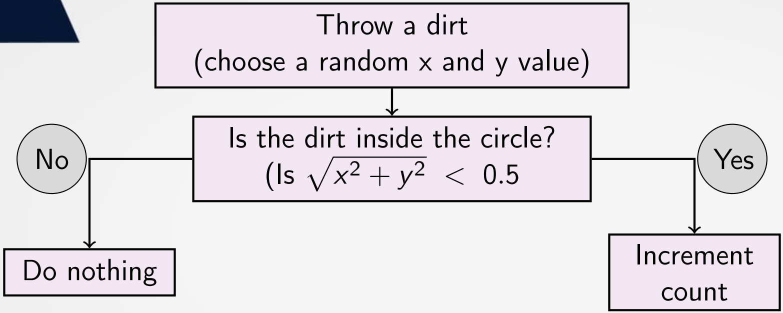
A Simple Example - Calculate π



A Simple Example - Calculate π

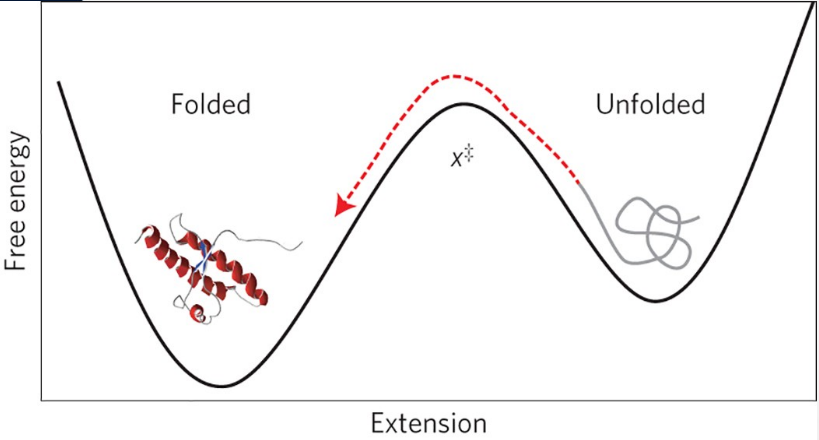


A Simple Example - Calculate π



The number of dirt inside the circle divided by the number thrown will be $\frac{\pi}{4}$!

A Biological Example - Protein Folding



A Biological Example - Protein Folding

Connect your atoms correctly, but in a guessed shape

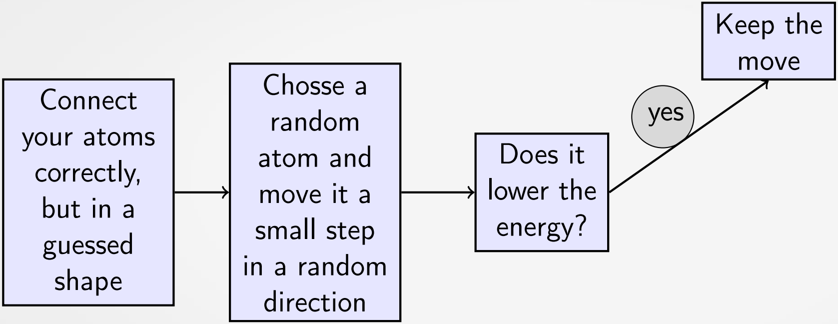
A Biological Example - Protein Folding

Connect your atoms correctly, but in a guessed shape $\longrightarrow$ Chosse a random atom and move it a small step in a random direction

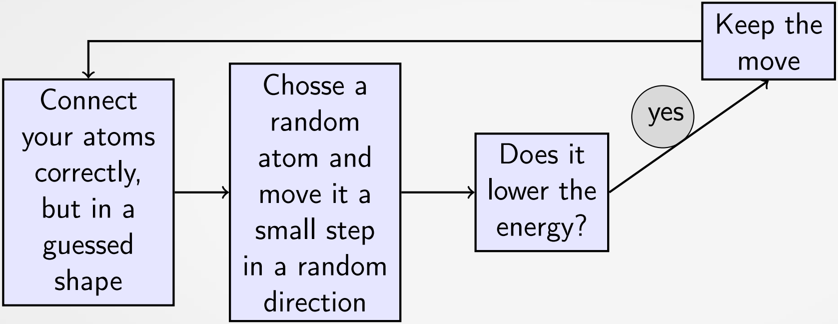
A Biological Example - Protein Folding

Connect your atoms correctly, but in a guessed shape → Chosse a random atom and move it a small step in a random direction → Does it lower the energy?

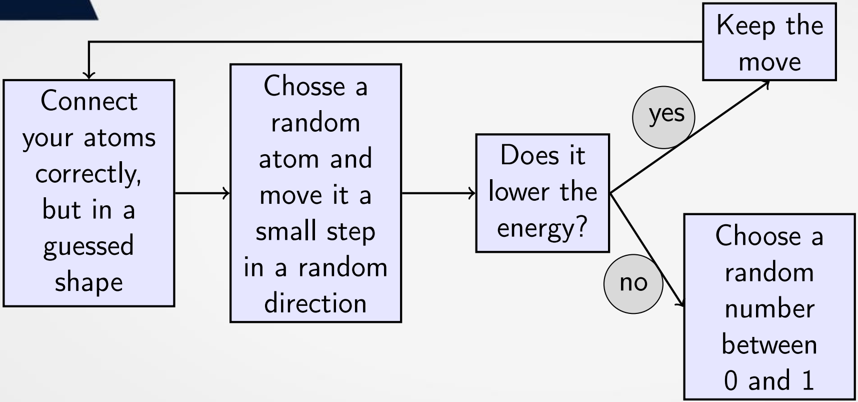
A Biological Example - Protein Folding



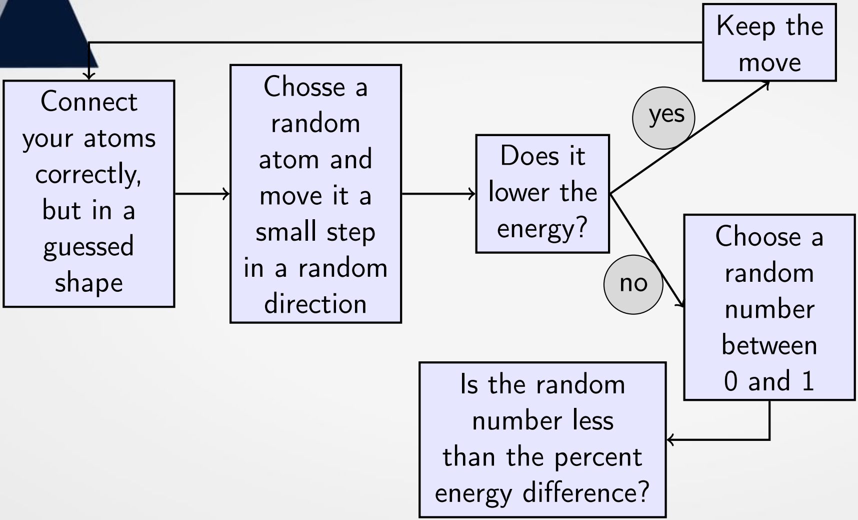
A Biological Example - Protein Folding



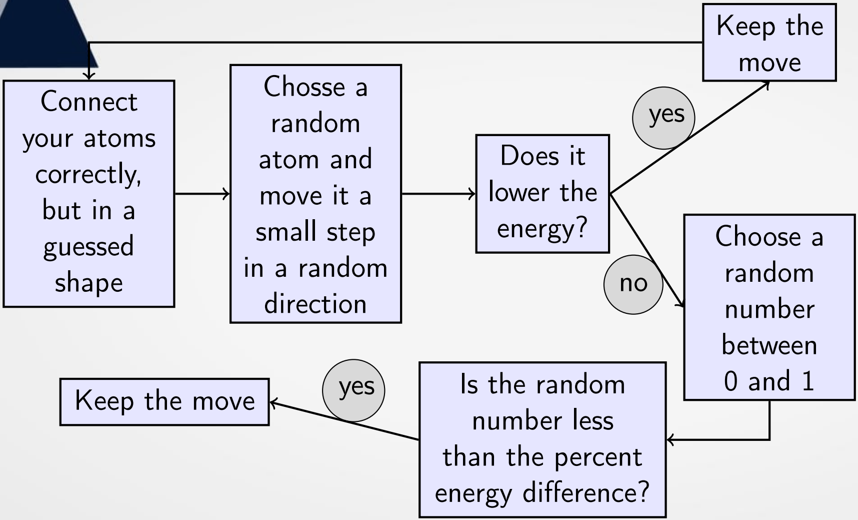
A Biological Example - Protein Folding



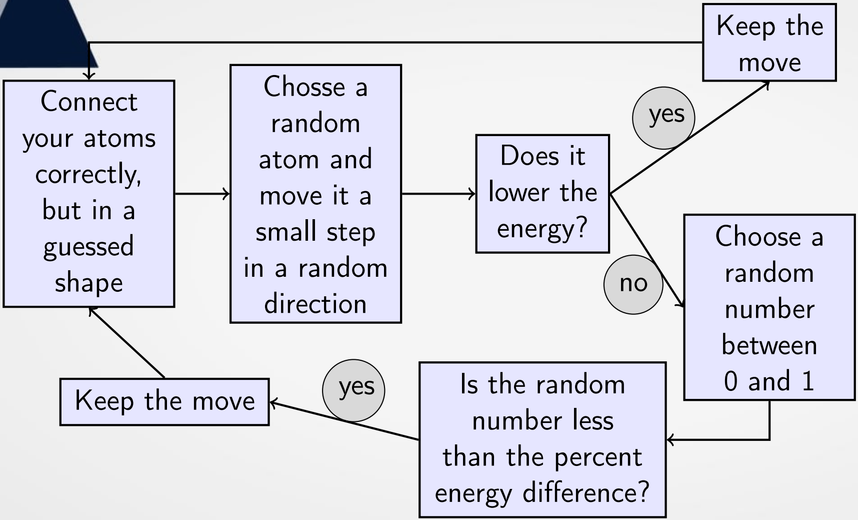
A Biological Example - Protein Folding



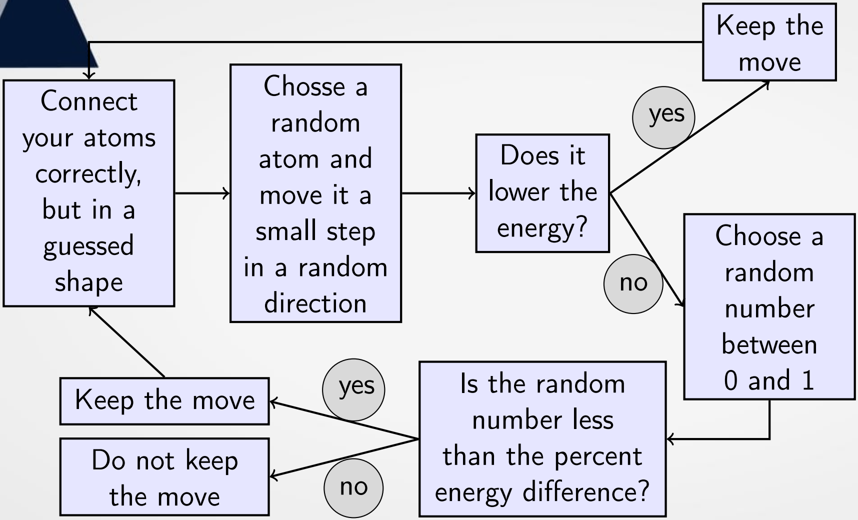
A Biological Example - Protein Folding



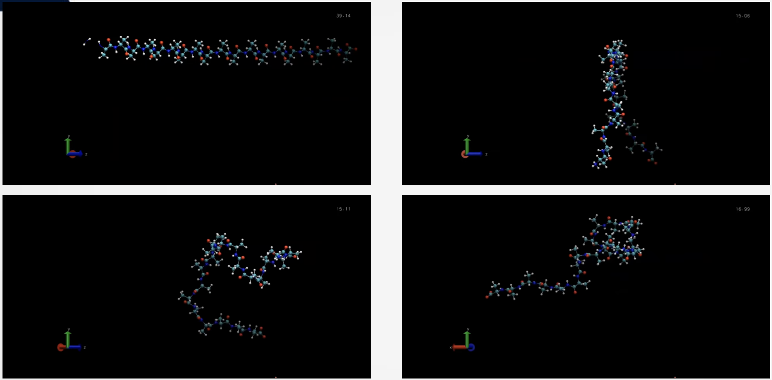
A Biological Example - Protein Folding



A Biological Example - Protein Folding



Monte Carlo Simulation of Polypeptide Chain Folding, 2015



Contents

Summary

- Monte Carlo (MC) methods use random numbers to solve problems that cannot be solved any other way.

Summary

- Monte Carlo (MC) methods use random numbers to solve problems that cannot be solved any other way.
- Used in a variety of fields:
  - Physics
  - Biology
  - Geosciences
  - Finance
  - ...

Summary

- Monte Carlo (MC) methods use random numbers to solve problems that cannot be solved any other way.
- Used in a variety of fields:
  - Physics
  - Biology
  - Geosciences
  - Finance
  - ...
- The versatility and efficiency of Monte Carlo algorithms make them an indispensable tool for tackling computationally challenging problems across diverse domains.

Contents

References

- Dr. Ana Bell Prof. Eric Grimson, Prof. John Guttag.
  Introduction to computational thinking and data science, 2016.
  MIT OpenCourseWare.
- Samik Raychaudhuri.
  Introduction to monte carlo simulation.
  Oracle Crystal Ball Global Business Unit, 2008.
- UMass Physics 13X.
  Introduction to monte carlo methods, 2020.

THANK YOU!

Any Questions?