Simulation and Optimization

Rubén Ruiz-Torrubiano

Invalid Date

Table of contents

Welcome

This is the website for the **Simulation and Optimization** book that will teach you the basics of simulation approaches and optimization techniques in the context of modern AI systems. The source code of the book and the examples are provided as open-source and free to use

Preface

This book was created as companion material for a semester graduate course on simulation and optimization. It is the author's opinion that in an age of rapid advances in the field of artificial intelligence, it is of utmost importance to focus not only on machine learning, but to study in detail the techniques that make current advances in AI possible. From those, the areas of simulation and optimization have the highest potential to reveal how intrincate current AI is intertwined with other areas of mathematics, statistics and computer science.

Simulation techniques are widely used in many scientific disciplines, ranging from climate models, epidemiology, and engineering to finance and logistics. These methods allow researchers and practitioners to analyze complex systems, evaluate scenarios, and make informed decisions when analytical solutions are infeasible or unavailable. Throughout this book, we will explore foundational concepts and practical approaches to simulation and optimization, providing both theoretical background and hands-on examples. In the context of AI, simulation approaches can be used to produce synthetic data for training in situations where these data are scarce, expensive, or simply impossible to collect. Another uses of simulation approaches include stress-testing algorithms, validating models under various hypothetical scenarios, and supporting decision-making in uncertain environments. By leveraging simulation, practitioners can gain insights into system behavior, identify potential risks, and optimize performance before deploying solutions in real-world settings.

Optimization approaches lie at the core of how machine learning is used in modern AI systems. Foundational algorithms like stochastic gradient descent make it possible to find optimal parameters for machine learning models using training datasets composed of millions of data points. Additionaly, optimization algorithms are used for hyperparameter tuning and can be found at the heart of classical approaches like support vector machines and logistic regression. In this context, both classical and metaheuristic approaches play a pivotal role in finding optimal or near-optimal solutions which are used in the broader context of specific applications in practice.

Throughout the book, we will assume that the reader has familiarity with linear algebra and calculus and possesses a good command of statistics and the basics of machine learning. Additionally, good background knowledge of the Python programming language is adviced for the practical part of this book.

1 Introduction

Simulation and optimization approaches are present in our everyday lives, albeit most of the time operating in a background plane. For example, when navigating with a GPS, the system simulates different routes and optimizes for the shortest or fastest path. Similarly, supply chains use optimization algorithms to minimize costs and maximize efficiency, while simulations help predict demand and manage inventory. These techniques are fundamental tools in decision-making processes across various industries, from transportation and logistics to finance and healthcare.

But what do simulation and optimization approaches have in common, apart from being complementary tools? The answer lies in the concept of a *model*. In the context of machine learning, we normally refer to a model as a mathematical or computational representation that captures the relationships between input data and output predictions. In simulation and optimization, a model similarly serves as an abstraction of a real-world system or process, allowing us to analyze, predict, and improve its behavior through experimentation and algorithmic techniques.

In the following, we will delve deeper into the concept of a model and how models are used in simulation and optimization contexts using some practical examples.

1.1 Simulating supermarket dynamics

Imagine you are in your favourite grocery store waiting at the checkout queue. For simplicity, let's assume there is only one open counter. When you arrive at the queue, there might be other customers already waiting, while the first customer at the queue is currently being served. Shortly after you, a new customer arrives, taking the next free spot right behind you. And then another customer arrives, and another one, and another one…

Let's try to break down how this system behaves and what are the most important interactions between the parts of the system. In general, we will distinguish between *components*, *states*, *events*, *inputs* and *metrics*.

- **Components:** These are the entities that interact with each other. In our example, we have customers, cashiers and the queue itself.

- **States:** The configurations of the system that represent valid combinations of specific properties of the components at a given moment of time. For instance, at each time the queue has a specific length: zero if it's empty, one customer, two customers, etc. Additionally, the cashier can be busy or idle. We can also count the number of customers currently present in the supermarket which have not yet arrive at the checkout queue.
- **Events:** The interactions themselves, like a new customer arriving at the queue, checkout start or checkout completion.
- **Inputs:** Whatever information is fed into the system, e.g. arrival times, service times, etc. These inputs can contain statistical assumptions, like the distribution of arrival times.
- **Metrics:** How we evaluate the system as a whole in a given time step. For instance, what is the average waiting time? How much time are the cashiers busy? How is the queue length distributed?

The system could be represented by the following Python code as a minimal variant.

```
import heapq, random

# event = (time, type, customer_id)
event_list = []
heapq.heappush(event_list, (first_arrival_time, 'arrival', 1))

while event_list and time < sim_end:
    time, ev_type, cid = heapq.heappop(event_list)
    if ev_type == 'arrival':
        if any_queue_free():
            start_checkout(cid, time)
            heapq.heappush(event_list, (time + service_time(cid), 'departure', cid))
        else:
            enqueue(cid, time)
        heapq.heappush(event_list, (time + next_interarrival(), 'arrival', next_id()))
    elif ev_type == 'departure':
        finish_service(cid, time)
        if queue_not_empty():
            next_cid = dequeue()
            start_service(next_cid, time)
            heapq.heappush(event_list, (time + service_time(next_cid), 'departure', next_cid
```

This code assumes that customers arrive at regular subsequent intervals after each arrival event. The parameter `sim_end` defines how long (how many steps) we want to simulate in this case. The function `service_time` returns the time the cashier needs for checking out customer `cid`. The next customer will arrive after a time given by the function `next_interarrival`, which can implement different stochastic behaviours.

```
%%| filename: stick-figure
%%| caption: A Stick Figure

\begin{tikzpicture}[scale=2, transform shape]
\draw  (9,19) circle (0cm);
\draw  (6.25,21) rectangle  node {\LARGE $Cashier_1$} (10.25,19);
\draw  (6.25,17.75) rectangle  node {\LARGE $Cashier_2$} (10.25,15.75);
\draw  (6.25,13.5) rectangle  node {\LARGE $Cashier_n$} (10.25,11.5);
\node [font=\Huge] at (8.25,14.75) {...};
\draw  (12.75,20) circle (0.75cm) node {\LARGE $C_{11}$} ;
\draw  (15.75,20) circle (0.75cm) node {\LARGE $C_{21}$} ;
\draw  (18.75,20) circle (0.75cm) node {\LARGE $C_{31}$} ;
\draw [->, >=Stealth] (12,20) -- (10.25,20);
\draw [->, >=Stealth] (15,20) -- (13.5,20);
\draw [->, >=Stealth] (18,20) -- (16.5,20);
\draw [->, >=Stealth] (21,20) -- (19.5,20);
\node [font=\Huge] at (21.75,20.25) {...};
\draw [->, >=Stealth] (24,20) -- (22.5,20);
\draw  (25,20) circle (0.75cm) node {\LARGE $C_{m1}$} ;
\draw  (12.75,16.75) circle (0.75cm) node {\LARGE $C_{21}$} ;
\draw [->, >=Stealth] (12,16.75) -- (10.25,16.75);
\draw  (12.75,12.5) circle (0.75cm) node {\LARGE $C_{n1}$} ;
\draw  (15.75,12.5) circle (0.75cm) node {\LARGE $C_{n2}$} ;
\draw  (18.75,12.5) circle (0.75cm) node {\LARGE $C_{n3}$} ;
\draw [->, >=Stealth] (12,12.5) -- (10.25,12.5);
\draw [->, >=Stealth] (15,12.5) -- (13.5,12.5);
\draw [->, >=Stealth] (18,12.5) -- (16.5,12.5);
\node [font=\LARGE] at (11.25,20.75) {$s_{11}$};
\node [font=\LARGE] at (11.25,17.25) {$s_{21}$};
\node [font=\LARGE] at (14.25,20.75) {$a_{21}$};
\node [font=\LARGE] at (17.25,20.75) {$a_{31}$};
\node [font=\LARGE] at (23.5,20.75) {$a_{m1}$};
\node [font=\LARGE] at (11.25,13) {$s_{n1}$};
\node [font=\LARGE] at (14.25,13) {$a_{n2}$};
\node [font=\LARGE] at (17.25,13) {$a_{n3}$};
\end{tikzpicture}
```

Part I

PART I: SIMULATION

2 Simulation basics

3 Monte Carlo

4 Discrete events

5 Agent-based simulation

Part II

PART II: OPTIMIZATION

6 Optimization basics

7 Gradient descent

8 Evolutionary algorithms

9 Summary

In summary, this book has no content whatsoever.

References