Lecture I - Optimal Stopping

Programming: Everyday Decision-Making Algorithms

Dr. Tobias Vlćek

About this Course

Teaching Team

(a) **Dr. Tobias Vlcek**

(a) **Dr. Nils Roemer**

About me

- **Field:** Optimizing and simulating complex systems
- **Languages:** of choice: Julia, Python and Rust
- **Interest:** Modelling, Simulations, Machine Learning
- **Teaching:** OR, Algorithms, and Programming
- **Contact:** vlcek@beyondsimulations.com

. . .

> Tip
>
> We really appreciate active participation and interaction!

Course Outline

- **I:** Optimal Stopping
- **II:** Explore & Exploit
- **III:** Caching
- **IV:** Scheduling
- **V:** Randomness
- **VI:** Computational Kindness

Participation

- Try **actively participating** in this course
- You will find it much (!) easier and more fun
- Material and slides are hosted here: beyondsimulations.github.io/Programming-Everyday-Decisions

Teaching

- **Lecture:** Presentation and discussion of algorithms related to everyday decision-making
- **Tutorial:** Step-by-step assignments to be solved and discussed together in groups
- **Difficulty:** Strongly depends on your background and programming experience

. . .

> Tip
>
> No worries, we will help you out if you have any questions!

Passing the Course

- **Pass/fail course** without exams
- 75% attendance required for passing the course
- Hand in the assignments of at least **two lectures**
- **Short presentation** and discussion at the end
- You work together in groups of **three students**

Handing in Assignments

- Each student group submits **one solution**
- Provide us **all** working notebooks of the lecture
- Hand in is due at the **beginning of the next lecture**
- At least **50 %** have to be correct to pass
- You have to pass at least twice

. . .

> Tip
>
> This is just in order to provide you with working solutions after each deadline.

Learning Python

We will mostly **not cover Python during the lectures!**

. . .

Question: Anybody know why?

. . .

- In our experience, the best way to learn is by **doing**!
- Here, we will focus on decision-making algorithms
- You will learn Python by doing the tutorials

. . .

> Tip
>
> Don't worry, we will help you out if you have any questions!

Difficulty of the Course

- At first it might be a little bit overwhelming
- Programming is similar to learning a **new language**
- First, you have to **get used to it** and learn words
- **Later**, you'll be able to apply it and see results
- Important: *Practice, practice, practice!*

Goals of the Course

- Learn the basics of programming
- Learn about **algorithmic thinking**
- Be able to **apply** methods and concepts
- **Solve practical problems** with algorithms

> Tip
>
> We are convinced that this course will be quite interesting and **teach you more for your daily life** than most other courses!

Why Python?

- **Origins**: Conceived in late 1980s as a teaching and scripting language
- **Simple Syntax**: Python's syntax is mostly straightforward and very easy to learn
- **Versatility**: Used in web development, data analysis, artificial intelligence, and more
- **Community Support**: A large community of users worldwide and extensive documentation

Help from AI

- You are allowed to use AI in the course, we use it as well (e.g., Claude, ChatGPT, LLama3 …)
- These **tools are great** for learning Python!

- Can help you a lot **to get started** with programming

. . .

> ⚠ Warning
>
> But you should *not* simply use them to *replace* your learning.

How to learn programming

Our Recommendation

1. Be present: Attend the lecture and solve the tutorials
2. Put in some work: Repeat code and try to understand it
3. Do coding: Run code, play around, modify, and solve

. . .

> Tip
>
> Great resources to start are books and small challenges. You can find a list of book recommendations at the end of the lecture. Small challenges to solve can for example be found on Codewars.

Don’t give up!

- Programming is problem solving, don’t get **frustrated**!
- Expect to **stretch** your comfort zone

Setting up Python

The Setup

- We will use Jupyter Notebooks for the tutorials
- Allow to **combine code and text** in one document
- We will use Visual Studio Code as an IDE

. . .

> Tip
>
> IDE = Integrated Development Environment

Install Python

- Sources are the Python website or Anaconda
- On macOS, Python is often **already installed**
- If not, I recommend Miniconda (via command line)

. . .

> Tip
>
> If the installation does not work, **let us know!**

Install VS Code

- Download and install from the website
- Built for **Windows, Linux and Mac**
- Install the **Python** and **Jupyter** extension
- Now you are ready to go!

. . .

> Tip
>
> Unsure on how to work with VS Code and notebooks? Take a look at the tutorial from VS Code and/or ask us! We are happy to help you out!

Python on iPads

- You **can run Python scripts** on your iPad
- But it is not recommended for the course
- However, you could use Juno **if you want to**
- It works **locally** on your iPad and can run notebooks

. . .

> **Caution**
>
> Not all packages available in Python are available here, thus you might need a computer to solve certain problems. For our course, this should not be a problem.

Your first code

Hello, World!

Task: Create a directory for the course and create a new file called `hello_world.py` with the following code:

```
# This is a comment in Python
print("Hello, World!")
```

```
Hello, World!
```

. . .

Run it with the green 'run' button or by pressing F5!

. . .

> **Note**
>
> "Hello world" is a classic example to start with. Often used as a test to check if your computer is working properly and that you have installed the necessary software.

Any questions

so far?

Optimal Stopping

What is Optimal Stopping?

Question: Anybody know what optimal stopping is?

- Optimal stopping is the problem of:
  - choosing the **best option**
  - from a **sequence of options**
  - where the options are revealed **one by one**

. . .

Question: Anybody have an example of optimal stopping?

Flat Hunting

Photo by Aditya Ghosh on Unsplash

Hiring applicants

Photo by Scott Graham on Unsplash

Dating

Photo by Shelby Deeter on Unsplash

Searching for a parking spot

Photo by Joseph Pearson on Unsplash

37% Rule

The Secretary Problem

- Imagine you're **hiring a secretary**
- You must interview candidates **one by one**
- Now, you must decide: **hire or continue searching**
- Once you reject a candidate, **you cannot go back**
- How to maximize chance of selecting the best candidate?

. . .

> **Note**
>
> The name is a bit misleading, as the problem is not about hiring a secretary, but about finding the best candidate. It comes from the 1960s and thus a little outdated.

Ideas?

The 37% Rule

The optimal strategy is to:

1. Look at the first 37% of options
2. Remember the **best one seen so far**
3. Choose the next option that's **better than the best seen**

. . .

Done - at least for this scenario!

. . .

> **Note**
>
> Chance of **selecting the best candidate is 37%**!

Why 37%?

- This is based on the **geometric distribution**

- The optimal stopping point is at `n/e`
- `e` is the base of the natural logarithm
- It is the limit of `(1 + 1/n)^n` as `n` approaches infinity

. . .

> **Tip**
>
> This is a bit more advanced, so don't worry if you don't understand it! We will not go into the details of the math here and focus more on the insights.

In Python

We can check this in Python:

```
import math

percentage = 1/math.e
print(f"Percentage of options to look at: {percentage:.3f}%")

candidates = 20
lookout_phase = candidates/math.e
print(f"Look at first {lookout_phase:.3f} candidates")
```

```
Percentage of options to look at: 0.368%
Look at first 7.358 candidates
```

. . .

> **Note**
>
> No worries if you don't understand the code! We are essentialy just using the formula to calculate the percentage of candidates to look at.

Variations

Cost of Searching

- What if each **additional search costs money**?
- Trade-off between finding better options and search costs
- Optimal stopping point can change!

Choose past candidates

- What if you could **choose past candidates**?
- Trade-off between choosing a candidate you like and continuing to search
- Optimal stopping point changes!

Questions?

. . .

> **Note**
>
> **That's it for todays lecture!**
> We now have covered a brief introduction into optimal stopping and seen how to set up Python.

Literature

Interesting Books to start

- Christian, B., & Griffiths, T. (2016). Algorithms to live by: the computer science of human decisions. First international edition. New York, Henry Holt and Company.

. . .

> **Note**
>
> The main inspiration for this lecture. Nils and I have read it and discussed it in depth, always wanting to translate it into a course.

Books on Programming

- Downey, A. B. (2024). Think Python: How to think like a computer scientist (Third edition). O'Reilly. Here
- Elter, S. (2021). Schrödinger programmiert Python: Das etwas andere Fachbuch (1. Auflage). Rheinwerk Verlag.

. . .

> **Note**
>
> Think Python is a great book to start with. It's available online for free. Schrödinger Programmiert Python is a great alternative for German students, as it is a very playful introduction to programming with lots of examples.

More Literature

For more interesting literature, take a look at the literature list of this course.