D0020E: Project report

Group 3: Simulation environment for demand-supply matching

Markus Blomqvist	amuobi-4@student.ltu.se
Elliot Palokangas Karlsson	kareli-6@student.ltu.se
Jakob Olsson	jakols-9@student.ltu.se
Axel Johansson	axejoh-0@student.ltu.se
Cristian Fierro Phillips	criphi-8@student.ltu.se

June 28, 2023

Contents

1 Introduction

1.1 Background

Efficiency in making resources and materials usable multiple times during its life-cycle is important and it's one of the reasons why this project is created. The circular economy is one way of using resources and materials in an efficient way. One way to get the benefits of a circular economy can be achieved by using a tool called the demand-supply matching (DSM) model [1]. An automated DSM model can be described by Fernández, S. et al. (2022) in [2] as a model to rank the match of someone's demand with someone else's supply in the most efficient way while still considering the environmental impact, seller reputation and bulk discount to name a few criteria. The automated DSM model can then automatically select the suppliers for auctioning based on the ranking.
An auction is similar to a special marketplace where sellers and buyers go through a process to buy or sell something. There is the auctioneer, the seller and the buyer. The auctioneer announces an initial price for goods provided by the seller. The potential buyers start to bid against each other until there is only one buyer left with a bid higher than everybody else's.
A program of a DSM model can be used to simulate auctions to see what the best combination between demand and supply is where auctioneers are sellers and bidders are buyers. One challenge is then to find the best or the most fair combination that can be described by Jain, R. et al. (1999) in [4] that the fairness index is a number between 0 and 1, where a number close to 1 is considered to be very fair.
A programming language that can be used to construct a simulation is Python that is an easy but slow language, which is great for prototyping systems. In order to create a simulation of the (DSM) model many aspects will need to be covered but a core part of it will be the minds of the operation the simulation engine. The engine will control the simulated environment and instance the actions of the actors. Although speaking of actors, their behaviour will also have to be simulated to test different bidding tactics to either match human behaviours or to optimise certain criteria that one could place on a order of supplies.
A simulation can be described by Ingalls, R. G. (2011) in [3], where the simulation has activities that is the processes and logic in the simulation and the executable events are placed in an event queue. There are entities in the simulator that can cause changes in the states of the simulator, according to Ingalls, R. G. (2011). Ingalls, R. G. (2011) highlights that the simulator can have system state variables such as a variable for the current time and global variables that is always accessible and configurable from the entire simulator model. When a simulator runs several times (iterates) the output of the simulation will be more correct with average values, according to Ingalls, R. G. (2011). A simulator can have different levels of complexity. The simulator that will be used in this project will be of simpler complexity in order to focus on single variable changes from the different actors in the (DSM) model. The changes between the runs single variables will then be graphed in order to compare what is the most efficient way to run auctions.

1.2 Problem description

An auction involves a lot of competition and psychological games. So, in order to win an auction, without triggering the bidding too much, you would like to have a strategy. Because, even if you have a lot of money and you want an object of the auction more than the other participants, you don't want to pay more than necessary for every object.
So, what's a good winning strategy in an auction? In this project auctions are simulated, and data is collected about the process. By looking at certain parameters the data is then analyzed, and different strategies can be compared to each other to evaluate which ones are successful or not.
How are the parameters chosen? There is a lot to be looked at when analyzing a strategy. Price is an obvious one but then there is a lot of space to be creative. In these times most auctions are probably held online, and people and machines can bid on objects from the other side of the world. Now it's not only the price of the object that matters but also the cost to transport the object to the winner. Transportation can be bad for the climate if the object needs to fly instead of taking the train.

This project will initially analyze the strategies by using price as an indicator to evaluate success. The purpose is to simulate the auctions, analyze the strategies of the bidders and to find the best DSM by analyzing the fairness index after the simulation is completed, because of the challenges when considering the environmental impact for example.

1.3 Tasks and goals

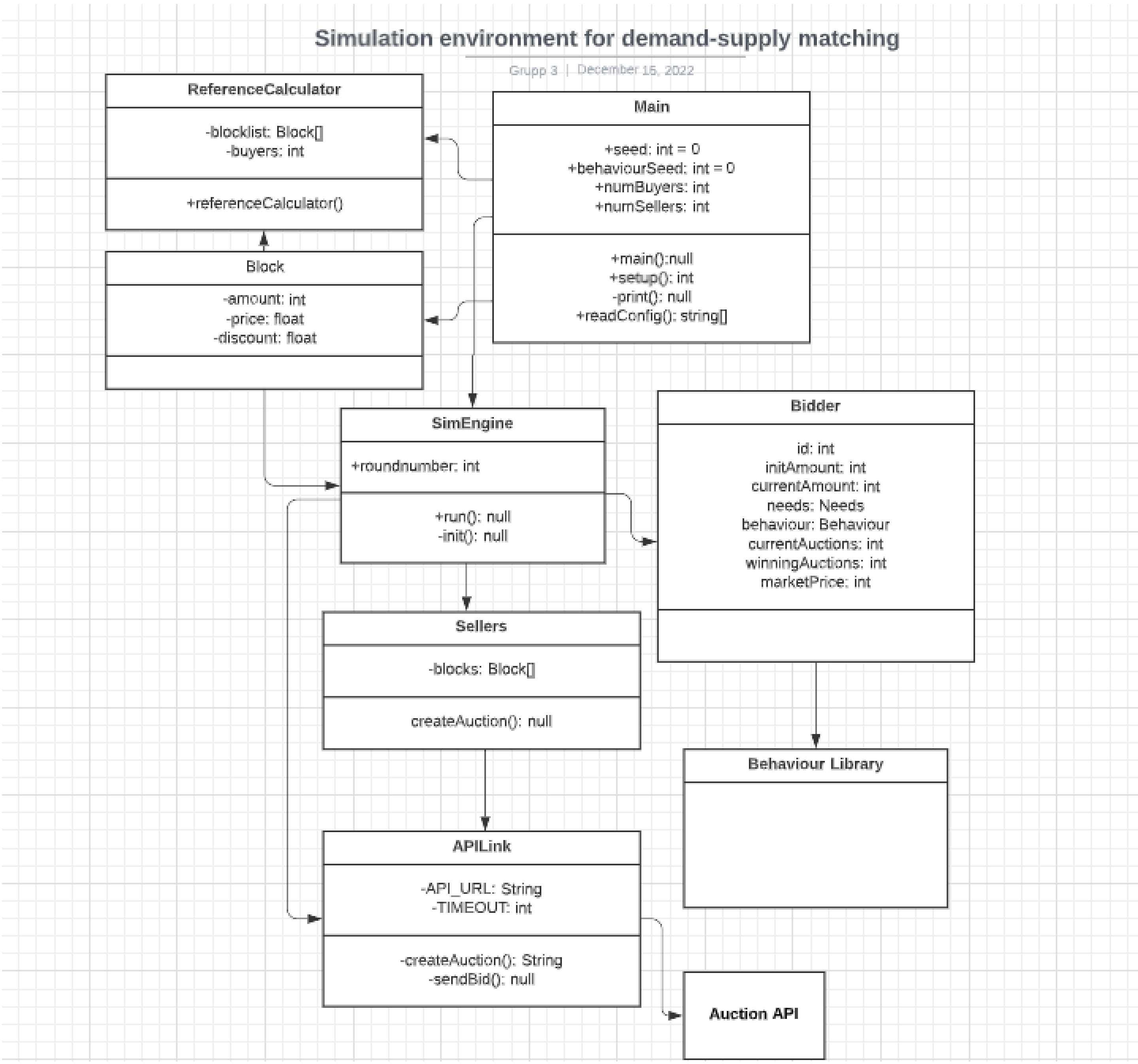
In order to solve the problem described in the preceding section we will need to create a system capable of both calculating the best possible fairness for a given set of buyers and sellers and calculating the fairness of simulated auctions on the same set given different auctioning strategies. We expect that the fairness data gained from this can be analysed in order to further research into demand-supply matching, especially in regards to how the behaviour of auction participants influences fairness.
We can express what we want our system to achieve as follows:

1. A user of the program should be able to decide on how many buyers and sellers they would like to simulate, how much product each demands or supplies and what prices and potential discounts that apply to the product with each seller. They will also specify what strategies the bidders of the auctions will utilise. A seed number will in addition to this be used so that a re-run can be performed with exactly the same outcome if necessary.
2. The program should be able to calculate the optimal Raj Jain fairness score possible given the input data.
3. The program should then simulate several parallel auctions where the buyers will be participating in one or more auctions set up by the sellers. Their behaviour in these auctions will be decided by the strategies chosen in (1). After the auctions have concluded a fairness score like in (2) will be calculated based on the results of the auctions.
4. The program should output the fairness index score from (2) and (3) along with other data that is deemed relevant. This data can then be utilised by the user, potentially in conjunction with data from different input, to analyse the fairness impact of different combinations of auctioning strategies.

Due to constraints in resources and time this system will only be built to handle the calculation of fairness in regards to the cost of products. Expanding it to handle fairness in other things of interest such as transportation distance, environmental impact or quality of product may be a subject of future work. For the same reason as above the system will not be handling cases where an auctioneer is running several auctions in parallel since this would increase complexity. As this is an abstraction of reality the system will also work under the assumption that all products being sold are wanted by the buyers and are equal to all the other batches of products. Lessening this abstraction by linking our system to filtering ranking model like the one conceptualised by Fernández, S. et al (2022) [2] may be an interesting avenue of further research.

2 System design

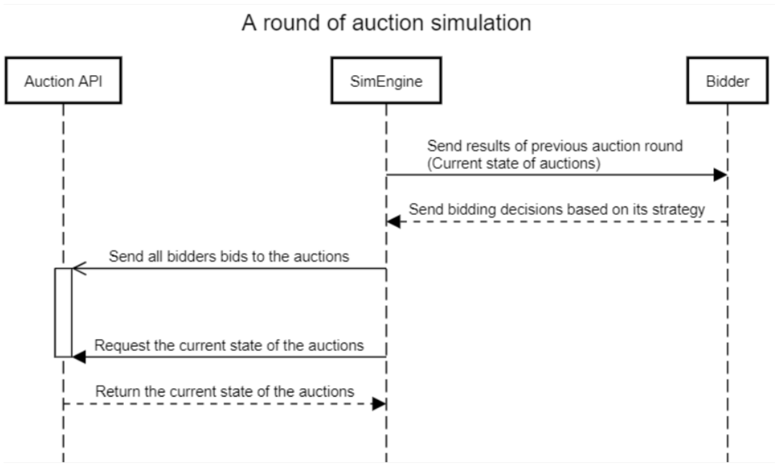
2.1 Static Design



In order to implement the product requirements this static design has been developed. It had three main goals: To separate the program into as many pieces that can be developed in parallel as possible, to provide an architecture that is easy to modify and expand and to centralise the extensive Auction API calls to a single place.
The program starts in Main with reading its input data from a configuration file. From this data it creates

a number of "blocks" of product to be sold by sellers and bought by bidders. The Reference Calculator uses this to calculate an optimal fair distribution of product between buyers to compare the simulation result with. The SimEngine coordinates the auction simulator, acting as a middleman between the bidders, sellers and the Auction API that handles the actual auctions. It is this class that creates the sellers and bidders, passing on the blocks and other configuration data from the configuration. The sellers will create auctions to sell the product blocks they are given while the bidders will bid on these according to the design described in 2.2. The bidders will do this according to a bidding behaviour chosen by configuration and taken from the Behaviour Library.
In order to simplify the implementation of the system no class communicates with the Auction API directly but instead with the APILink that handles the many calls and responses that have to be made in the course of a simulation.

2.2 Dynamic Design


A round of auction simulation
Auction API
SimEngine
Bidder
Send results of previous auction round (Current state of auctions)
Send bidding decisions based on its strategy
Send all bidders bids to the auctions
Request the current state of the auctions
Return the current state of the auctions

This design describes how a single round of the auction simulation that SimEngine is managing functions. By acting as a middleman between the many bidder objects the SimEngine class can sort out conflicting bids and uphold a rigid and fair round based structure that eliminates the risk of the time-sensitive Auction API influencing the results of the simulation.

3 Result

We can analyse the final level of feature completeness of the system by looking at section 1.3 and comparing this to the developed simulation environment. We will compare each of the goals there to the finished product:

1. Regarding the configuration options all those mentioned and more have been implemented. The user can specify a seed, decide numbers of buyers with differing strategies, sellers, numbers of blocks of products in each seller and their price, discounts and amounts. In addition to this they can also decide on the balance between supply and demand, how many auctions to run in parallel and how many rounds of inactivity an auction will stay open for.

2. The program does calculate the optimal Raj Jain fairness score and one can also choose to output not just the optimal fairness but all calculated fairness scores along with what combination of purchasing products that led to them.

3. The program does indeed simulate auctions according to the user configuration.

4. In addition to the fairness score data that may be of research interest like auction winners, market prices, prices paid and to which degree a buyers needs were fulfilled are also output.

The constraints placed upon the program were all upheld except for one, auctioneers do run several auctions in parallel. This is because a seller selling all of their products in one auction does not produce enough granularity for the buyer strategies to show themselves in action. The sellers instead sell smaller portions at once creating more auctions in total.
Regarding section 2 the design was largely unchanged. The only exception from this in the static design (2.1) is that all output is handled in a separate class instead of in the main class. The dynamic design (2.2) remained entirely unchanged.
The method in which this result was achieved was almost entirely in the base version of Python with only the "requests" library needed to be added. This makes the final result quite easy to use and install since there is not a wide range of dependencies needed.
The maturity of the different sections within the project are quite close and most parts reached a good level, this was in part due to our planning and communication when developing. Parts of the configuration as well as bidders could possibly be on a lower maturity level. This is mainly due to their code and structure falling behind to the other parts when it comes to understanding and changing it.
As for any planned parts which were not realized there are few. Most of the sections which were planned were a must have for the system to work properly. The few extensions we had were not needed at all, however the potential automatic graphing and analysing part of the output would have been a good feature to showcase and easier do the tests.

4 Conclusion

4.1 What is implemented now

In this version that which has been implemented is:

1. A configuration file reader that can read the users specified input.
2. A Raj Jain fairness calculator that calculates the optimal fairness of the given input.
3. A connection to an API were auctions are created and stored.
4. An engine that holds a specific amount of auctions at a time for the buyers to bid on.
5. A library of behaviours for the buyers to dictate their bidding patterns.
6. Several points of configuration for buyers and sellers.
7. And also some auxiliary functions.

What has essentially been implemented is the foundation of a supply demand matching simulation were generated data can be an indicator of how a real world auction can be held to get wanted results.

4.2 What can be worked on further

Some stretch goals that weren't reached and can be the focus of future work is an automatic graph plotter for the output data to automatically show the internal thoughts of the different buyer strategies. A further goal was to implement an automatic tester for if one wants to run multiple tests with close to the same input it would run all tests automatically. Another unrealised goal was to implement a GUI to help setup the configuration file so that initial setup would be easier. Lastly one of our goals was to decrease the run time of the program since it currently has exponential properties that make running simulations on large amounts of data unfeasible.

4.3 What it can be used for

In short, a simulated auction using a demand and supply matching engine has the potential to create significant value for both buyers and sellers, while also improving the overall efficiency of the marketplace.
During the auction, the matching engine would analyze the supply and demand for the product being sold, taking into account factors such as pricing, distance and or quality. Based on this analysis, the engine would match the highest bidder with the most matching seller, creating a more efficient and effective transaction.
One of the key benefits of using a demand and supply matching engine for auctions is the ability to create more accurate matches between buyers and sellers. By analyzing a range of data points like pricing, distance and customer behaviours the engine can identify the most suitable matches for both parties involved. This not only improves the likelihood of successful transactions but also leads to more satisfied customers and higher levels of repeat business including having a less impact on the environment which is also something that can be measured with the Raj Jain fairness index.
Another benefit of using a matching engine for auctions is the ability to create a more streamlined and personalized experience for buyers and sellers. By matching buyers with sellers that meet their specific needs and eagerness to buy, the engine can create a more efficient and effective marketplace, reducing the time and effort required to find the right match. This, in turn, can lead to faster sales and increased revenue for sellers.
Moreover, using a demand and supply matching engine for auctions can provide significant value in terms of data insights. By analyzing the patterns and trends in buyer and seller behaviour, the engine can help identify opportunities for growth and optimization, enabling sellers to better understand their markets and tailor their offerings accordingly. This can lead to more effective pricing strategies, targeted marketing campaigns, and other key business decisions.
One potential downside of using a matching engine for auctions is the risk of technical glitches or errors. If the engine malfunctions or produces inaccurate matches, it could lead to frustrated buyers and sellers and damage the reputation of the marketplace. To mitigate this risk, it's important to regularly test and update the engine, as well as provide clear communication and support for users considering it is hard to analyze mass psychology.
Overall, a simulated auction using a demand and supply matching engine has the potential to create significant value for both buyers and sellers, while also improving the overall efficiency of the marketplace. By leveraging the power of technology and data analysis, these engines can create more accurate matches, provide a personalized and streamlined experience for users, and generate valuable data insights for sellers. As digital technologies continue to evolve, demand and supply matching engines are likely to become even more prevalent in auctions and other marketplaces, transforming the way we buy and sell goods and services.

References

[1] Ulf Bodin et al. "Demand-supply matching through auctioning for the circular economy". In: *Procedia Manufacturing* 54 (2021), pp. 82–87.

[2] Shai Fernández, Ulf Bodin, and Kåre Synnes. "An automated demand-supply matching (DSM) ranking model for the circular economy". In: *IECON 2022–48th Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2022, pp. 1–6.

[3] Ricki G Ingalls. "Introduction to simulation". In: *Proceedings of the 2011 winter simulation conference (WSC)*. IEEE. 2011, pp. 1374–1388.

[4] Raj Jain, Arjan Durresi, and Gojko Babic. "Throughput fairness index: An explanation". In: *ATM Forum contribution*. Vol. 99. 45. 1999.

Appendices

A Instructions for installing, testing and source code

More information about the project and the instructions for installing, testing and recommended further development can be found on the **Github repository**. All source code for the project is also found there.