Planning and Reinforcement Learning - Assignment 1
Inventory Problem

Wolf, Florian

November 7, 2019

Introduction

This assignment aims to apply Reinforcement Learning (RL) to a specific inventory problem. We will use a very popular method called Dynamic Programming (DP). DP is able to find the optimal policy given a finite time horizon and a perfect model of the environment. We assume this environment to be a Marcov Decision Process (MDP) [3].

Over the last centuries DP has been widely used to optimize inventory size, hold-overs and plan restocking [2].

The given 'Ínventory Problem' challenges us to find the optimal policy for restocking a seasonal item within a finite timespan. The demand is stochastically determined and can there fore never be precisely determined.

The challenge in predicting the optimal policy lies in the stochasticity of the environments. Our case is two-dimensional in time and inventory size. Adding more dimensionalities to simulate more complex problems is possible. A common example is the item placement in the warehouse [1].

1 Algorithm

In this section we will discuss the implemented algorithm as well as the parameter settings and model constrains. The full code can be found on Github 3.1.

We implemented it in python. For reproducibility we used fixed random seeds. Each experiment was performed 10 times to define error margins and ensure significance of the results. To constrain our problem we took the following assumptions:

- The starting inventory can be chosen arbitrarily high w/o costs
- Left over items are discounted from the profit applying the purchase price
- No inventory cost or maximum purchase amount
- A maximum of one item can be sold per timestep
- The demand increases linearly over time
- The maximum inventory size is equal to the maximum number of timesteps

Using these assumptions we can build a model and apply DP policy optimization [3]. We define our parameters as follows:

- Selling price: $s = 20$

- Purchase cost: $k = 10$ for timesteps $1 - 500$
- Purchase cost: $k = 15$ for timestep $501 - 900$
- No more purchase possible after timestep 900
- Maximum timestep: $T = 1000$
- Maximum inventory: $M = 1000$

Based on the value function discussed in class we implement a profit function. Opposite to the value, the profit function wants to be maximized. Function 1 takes in account all future profits. We hardcode an exception for the last timestep, in which we solely discount the value off all unsold items.

$$P_t(x) = \underset{d_t}{\operatorname{argmax}}\{s\mathrm{I}\{\mathrm{d_t} > 0\} - k\mathrm{I}\{a > 0\} + P_{t+1}\,(x - d_t + a)\} \tag{1}$$

We initialize our algorithm by creating an $1000x1000$ empty array array grid for all possible t and x values. This gird will store the maximum profit in each state. We do initialize the same grid for the optimum amount of orders. For later visualization we also store the demand in every timestep.

The demand is chosen randomly for each state considering the probability distribution at that timestep. We start the mainloop over each timestep. Inside we loop further over all possible inventory states. Depending on the current demand and inventory we create an array with all possible number of order.

$$x \in \begin{cases} [d_t - x_t, M - x_t + d_t], & \text{if } d_t > x_t \\ [0, M - x_t + d_t], & \text{otherwise} \end{cases} \tag{2}$$

The best profit for this state is stored in the corresponding grid. At the end of both loops we save store the data using Pandas dataframe. This experiment is repeated 10 times with individual random seed for each run.

2 Results

In this section we will elaborate the results. There fore we look closer at the plots of both profit and order amount. To assure correctness we will also have a look on the demand distribution over the whole timespan. For presentation purposes we will cherry-pick the results of most suitable experiment. All remaining results can be found in the appendix.

2.1 Expected Profit

Under the assumptions of a maxium of 1 sale per timestep, a total of 1000 timesteps and a linear rising sale probability with margins $[0, 1]$ we get an expected total sale of 500. To calculate the expected profit we multiply this by the selling price.

$$\mathbb{E}(P) = s * \mathbb{E}(\sum_t d_t) = 10000 \tag{3}$$

To visualize this we plot the Inventory, the orders and the summed up demand over time. We use the policy with highest profit from each run and recursively determine the optimal next step. In 1 we can see how the starting inventory size matches the total demand and most importantly ends in 0.

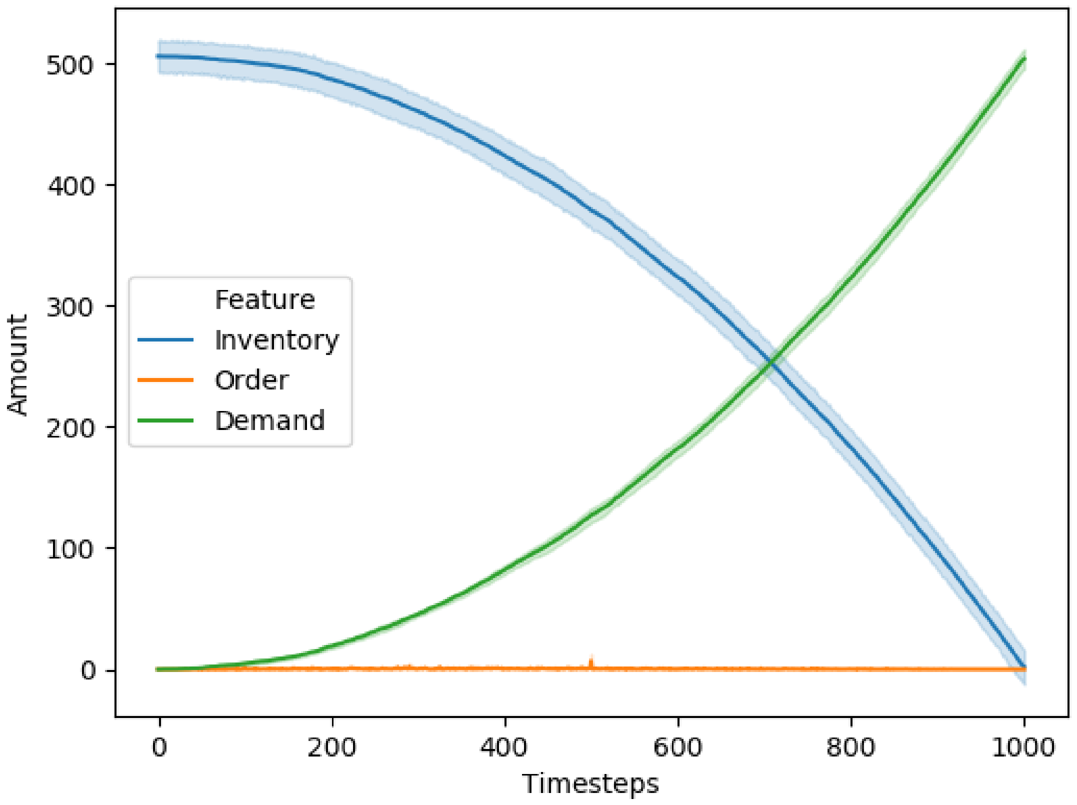

Figure 1: Inventory, order amount and demand over timesteps for the optimal run from all experiments.

2.2 Optimal Order Policy

The idea behind this model is to find the optimal policy. Looking at the heatmap 2 we can find the optimal behaviour by walking on the edge of the light yellow area. The heatmap shows the profit in each state, with the horizontal axis representing the inventory count and the vertical axis the timesteps. The profit ranges 1000 to -1000. Considering the constrains of our model the optimal policy is to start with an inventory of 500 items, which is the expected amount to be sold. This policy approximates parabolic the end point with 0 inventory. An empty inventory at the end of the timespan ensures minimum penalty for unsold items. We can also observer ourconstrain of rising purchase price as the policy get significantly darker after the 500 mark for states with sates which lower inventory than the expected sell.

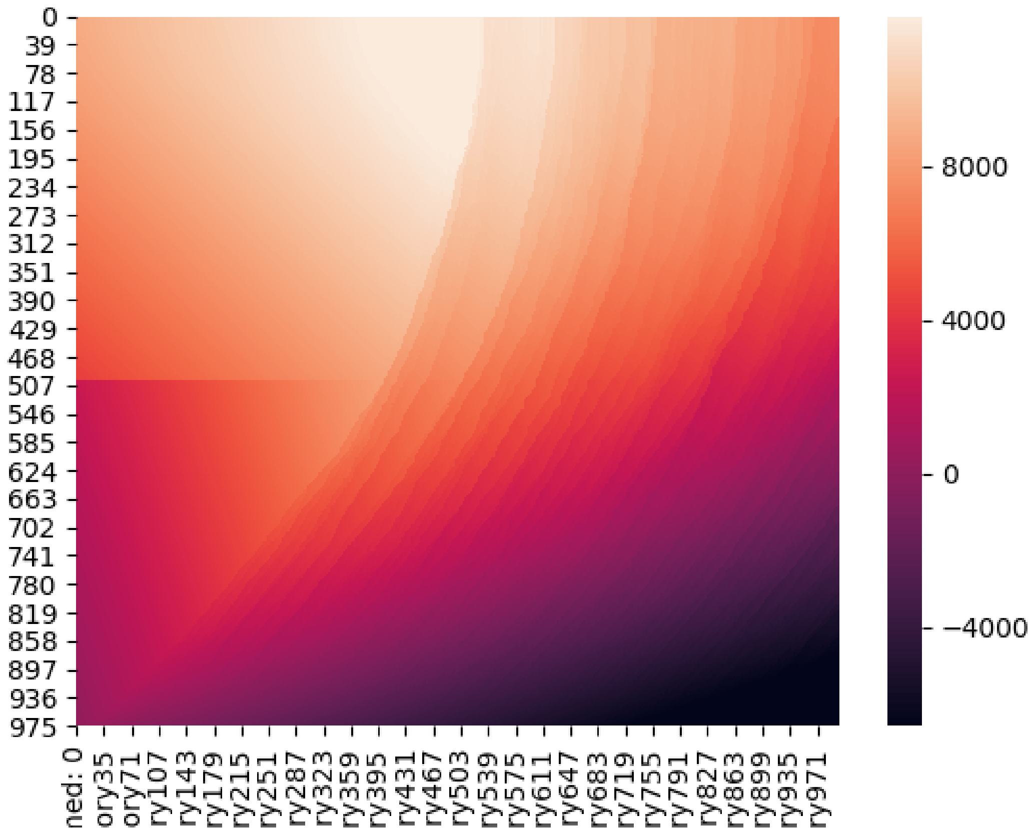

Figure 2: Profit heatmap for each time-inventory state of experiment nr.9 out of 10.

The profit heatmap gives a good insight on the optimal policy but does not give direct information about the ordering behaviour. There fore we decided to plot the optimal order amount for every state. Considering the stochasticity of our problem the ordering heatmaps show high variance. None the less a clear behaviour pattern in 3 is visible.

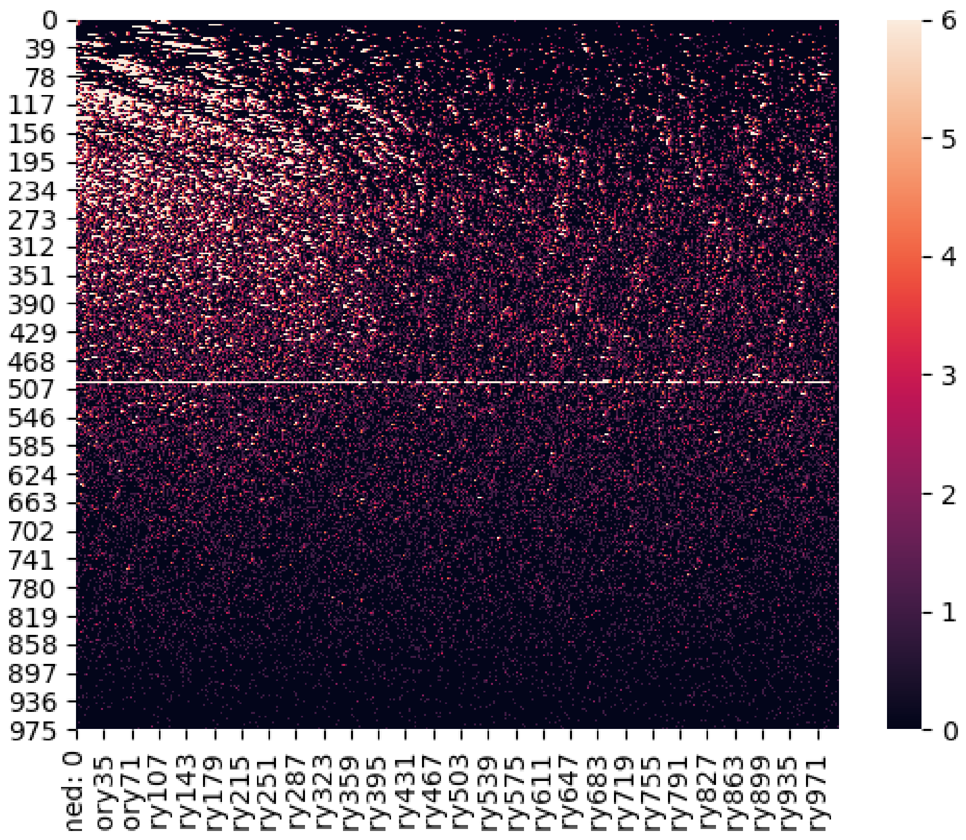
Figure 3: Order heatmap for each time-inventory state of experiment nr.4 out of 10.

The maps shoes that most orders are done in the begining time steps and in states l=with low inventory. Most states order at timestep 500, right before the price rises. After timestep 900 barley orders are made since we penalized these orders with a very high purchase price. These insights on the order behaviour were expected and more than reveling new insights help to confirm correctness of our model.

2.3 Demand over time

The demand represents the amount of items sold at each timestep. We defined a maximum of 1 per timestep. The probability of a maximum demand increases linearly over time. The demand can be seen as a sell when multiplied by the selling price. There fore the probability of a sell is equal to the demand probability which is defined by:

$$p = \frac{t}{T} \tag{4}$$

t is the current timestep and T is the total amount of timesteps. Due to stochasticity the demand changes for every experiment. To ensure correctness of our experiments we plot the demand of all 10 experiments over the timespan. For better readability the results are smoothed for each 10 timesteps.

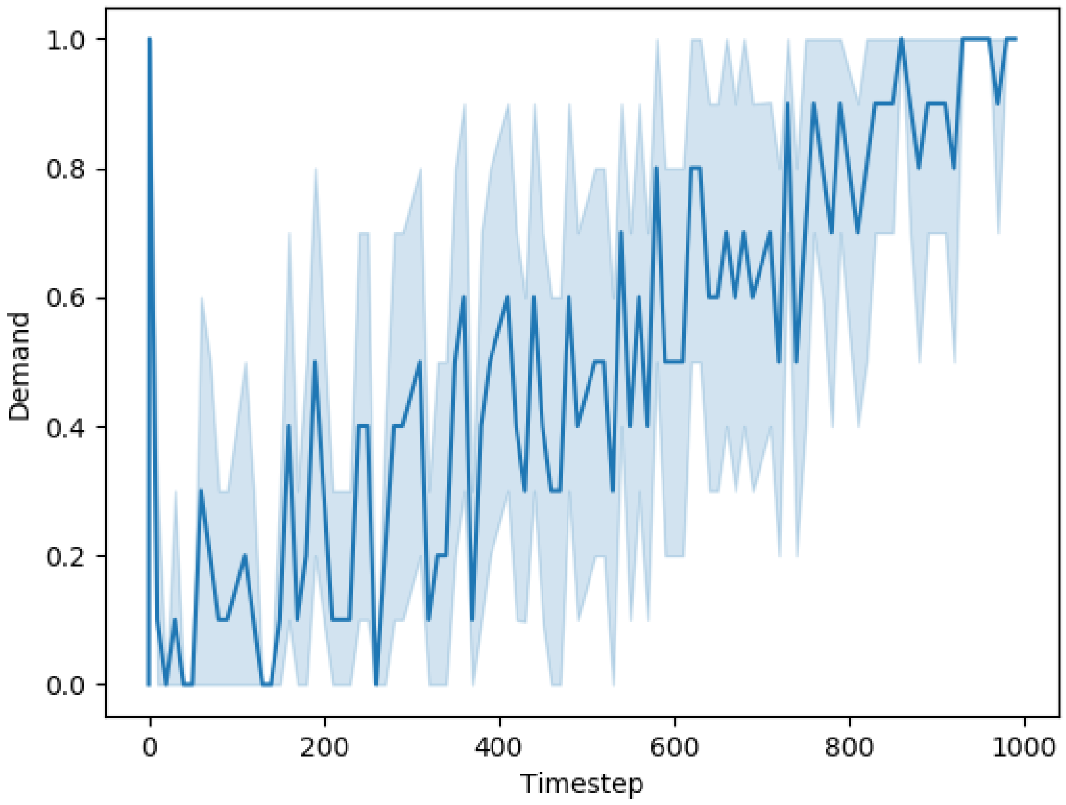

Figure 4: Resulted Demand in all experiments over timspteps. Smoothed every 10 steps.

We can see the expected variance of the weighted normal distribution. Thus the overall graph can be approximated linearly. This matches with our probability definition in 4.

3 Conclusion

This helps solve finite problems like this. Fails with higher dimensions and infinity.

DP successfully found the optimal policy to our problem. Astonishing as this sounds we need to keep in mind the constrains and simplicity of this model. Applied to real world problems this algorithm might encounter itself with the curse of dimensionality.

Further this algorithm is stuck within a finite timespan. Would it not be useful to find an algorithm able to solve infinite problems and reveal converging or diverging behaviour of parameters?

The algorithm was implemented after the guidelines of the accompanying lecture. A computational bottleneck are the two for-loops over the two parameters time and inventory size. A solution to this would be to use matrix multiplication which can then benefit of GPU or even TPU acceleration.

Bottleneck for loops Accelerate using matrix multiplication, profit from GPU acceleration

Well and last but not least (the word GPU probably already made your brain make the connection): Neural Networks. Applying Deep Learning to find this problem would not only find the same optimal policy but would also open the door to higher complexity. If it was not for the deadline being in one hour I would enthusiastically implement this alternative solution alongside the original scope assigment.

References

[1] Anton J Kleywegt, Vijay S Nori, and Martin WP Savelsbergh. Dynamic programming approximations for a stochastic inventory routing problem. *Transportation Science*, 38(1):42–70, 2004.

[2] Patricia B Reagan. Inventory and price behaviour. *The Review of Economic Studies*, 49(1):137–142, 1982.

[3] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

Appendix

3.1 Links

GitHub repository:
`https://github.com/3lLobo/ReinforcementLearningAndPlanning`

3.2 Profit Heatmaps

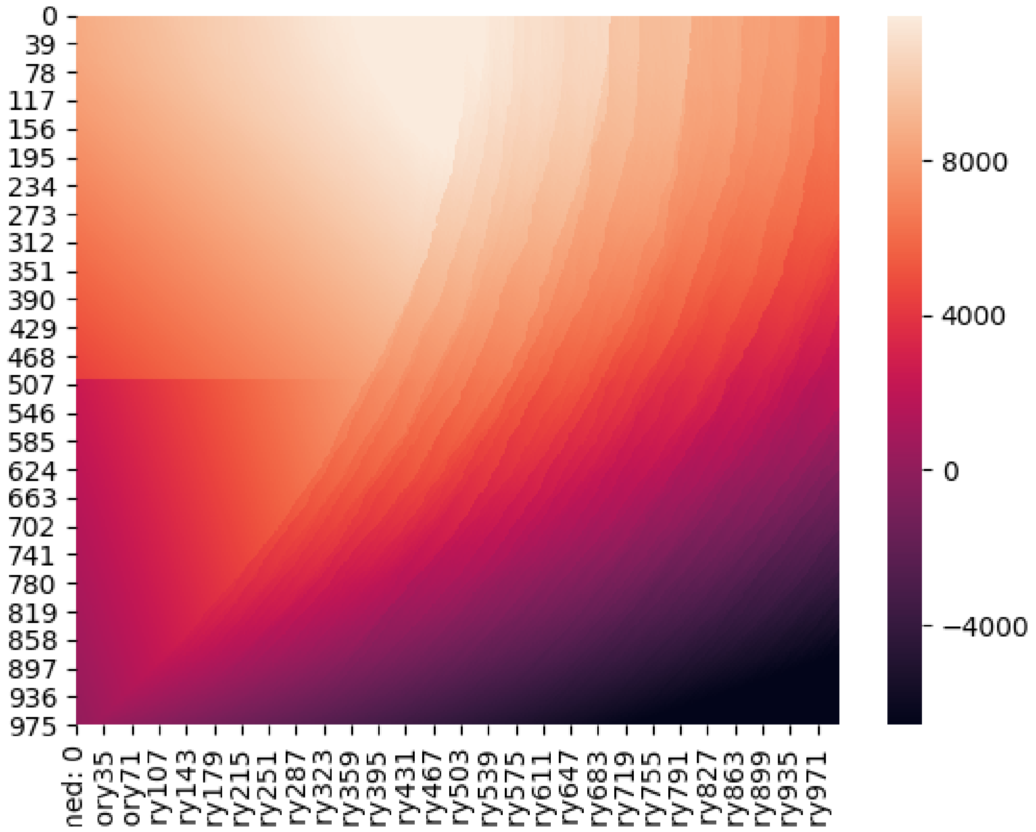

Figure 5: Profit heatmap for each time-inventory state of experiment nr.1 out of 10.

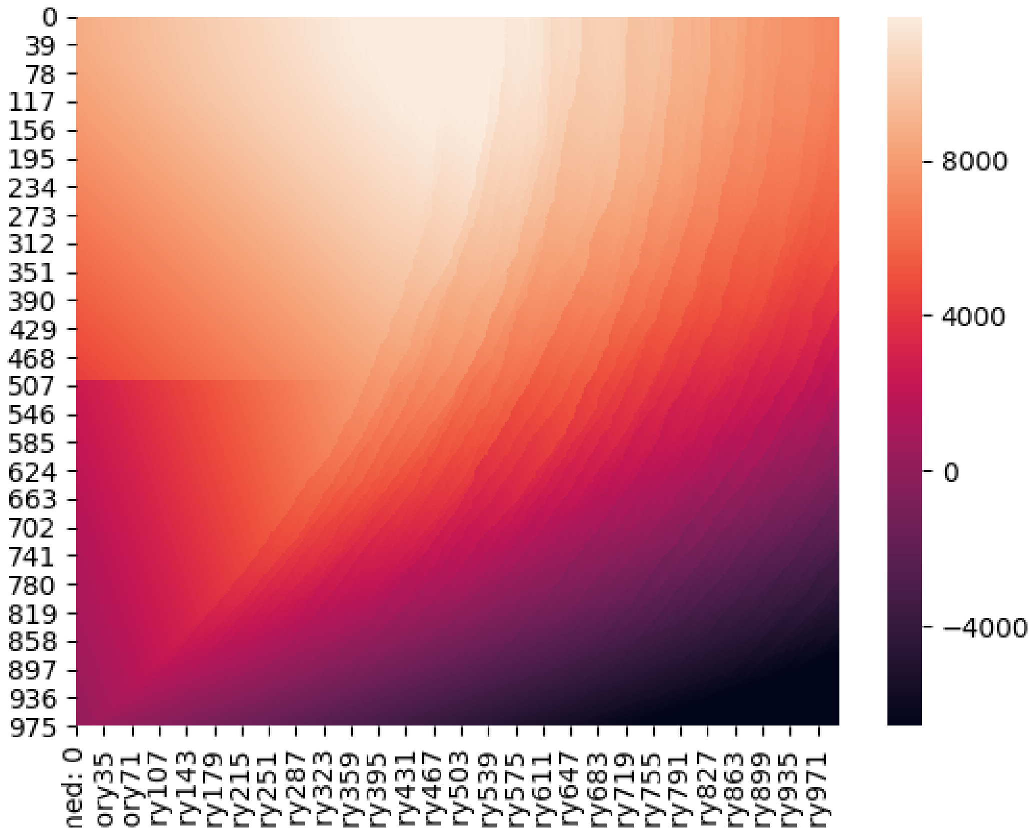

Figure 6: Profit heatmap for each time-inventory state of experiment nr.2 out of 10.

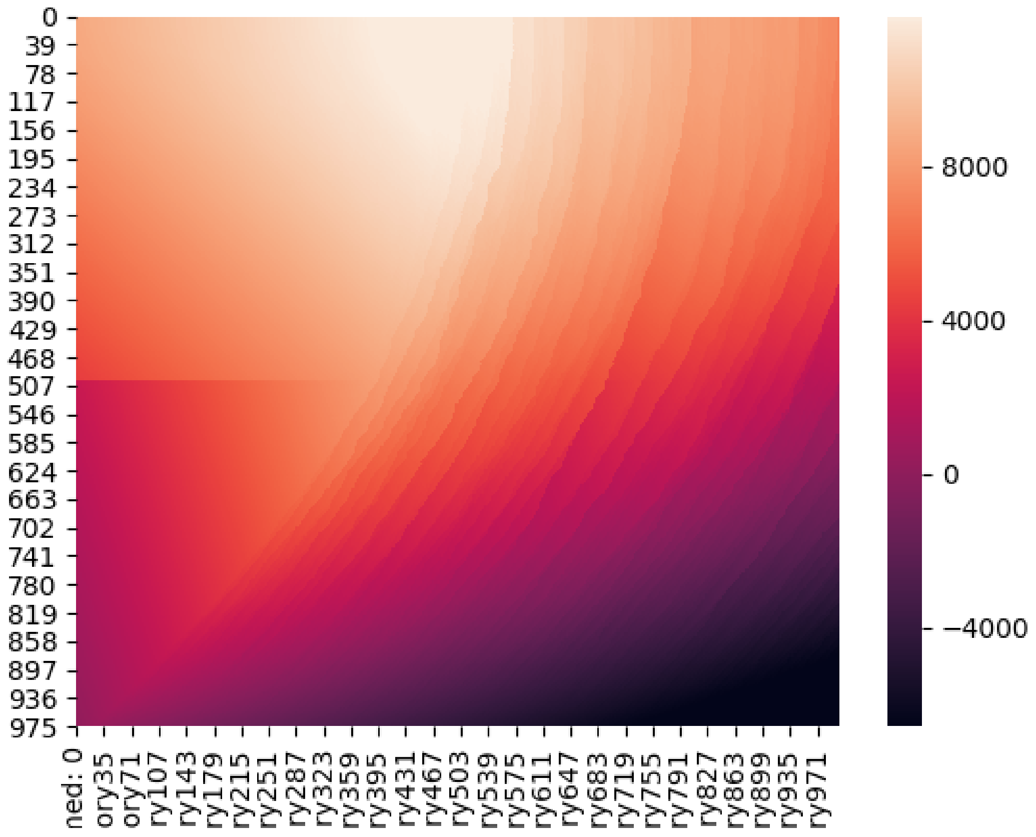

Figure 7: Profit heatmap for each time-inventory state of experiment nr.3 out of 10.

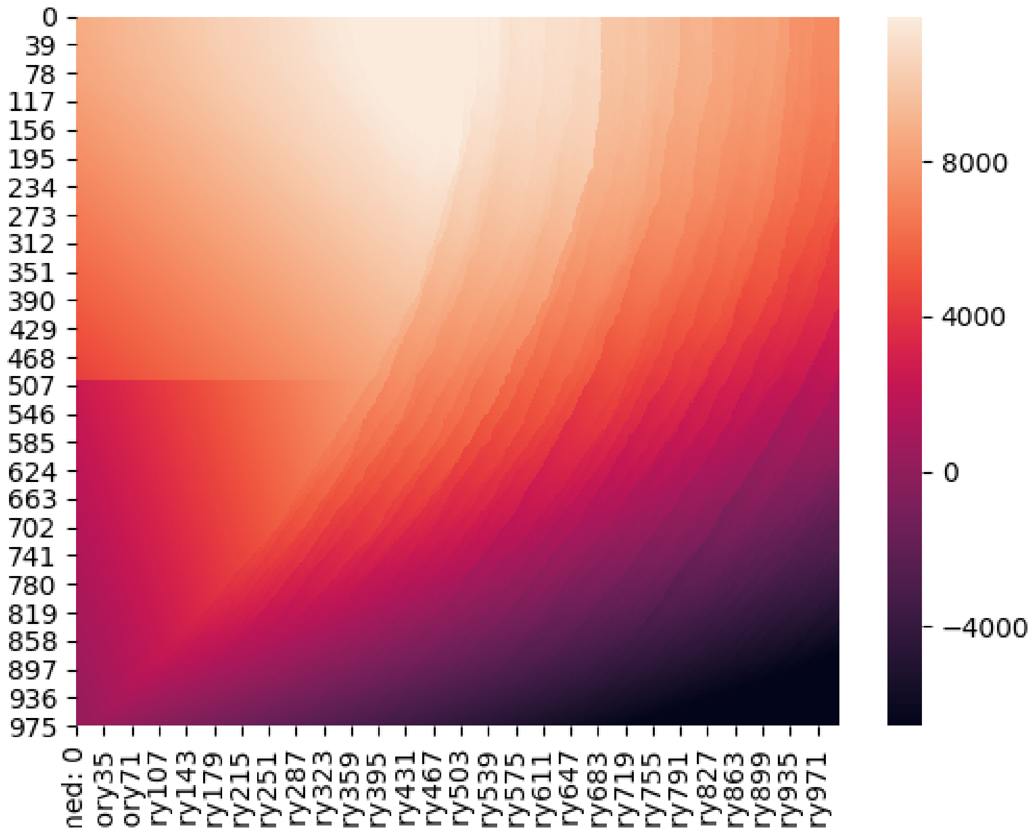

Figure 8: Profit heatmap for each time-inventory state of experiment nr.4 out of 10.

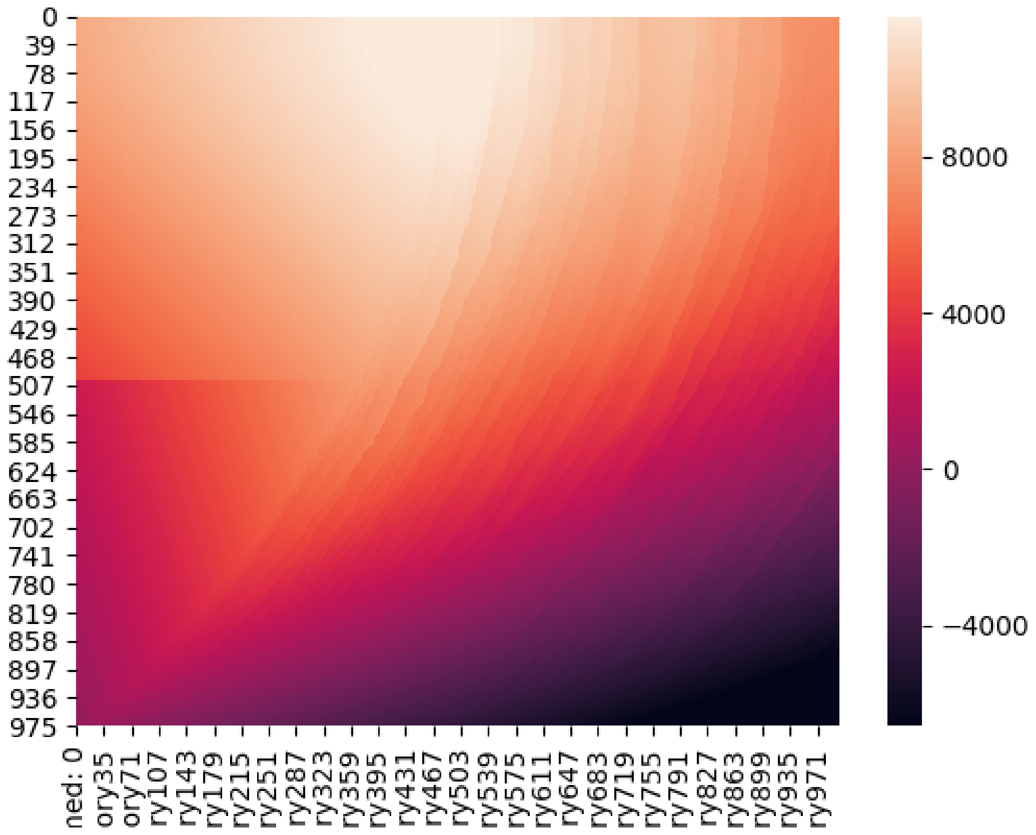

Figure 9: Profit heatmap for each time-inventory state of experiment nr.5 out of 10.

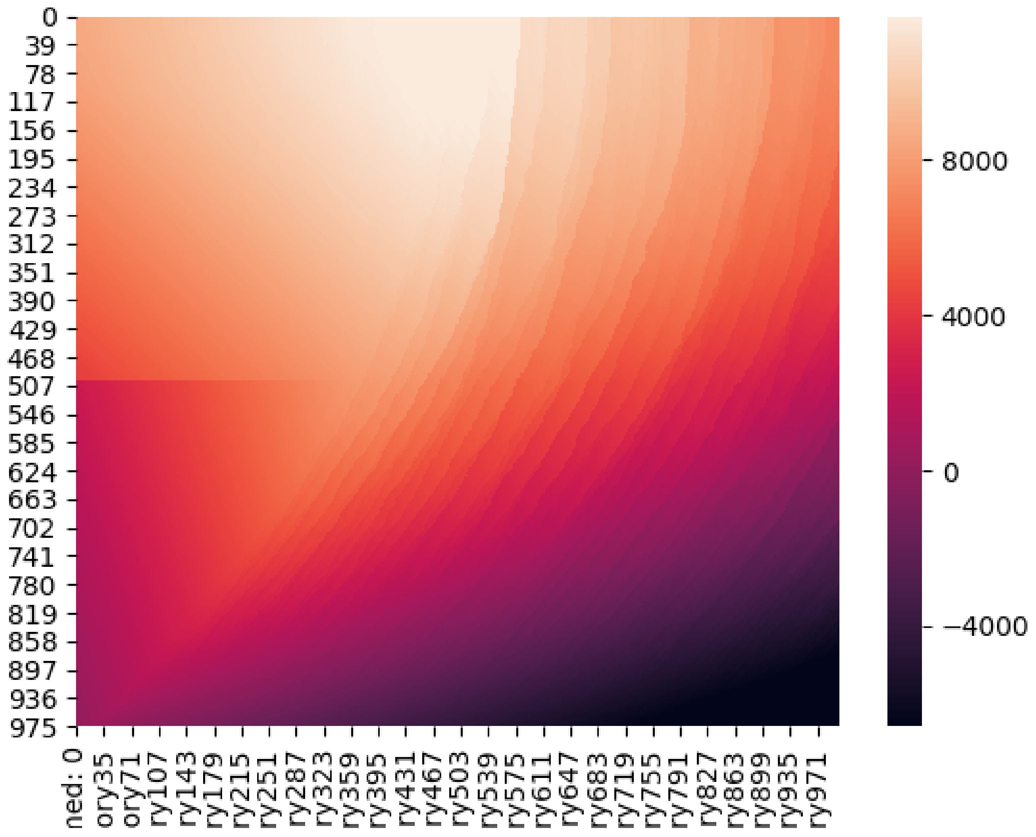

Figure 10: Profit heatmap for each time-inventory state of experiment nr.6 out of 10.

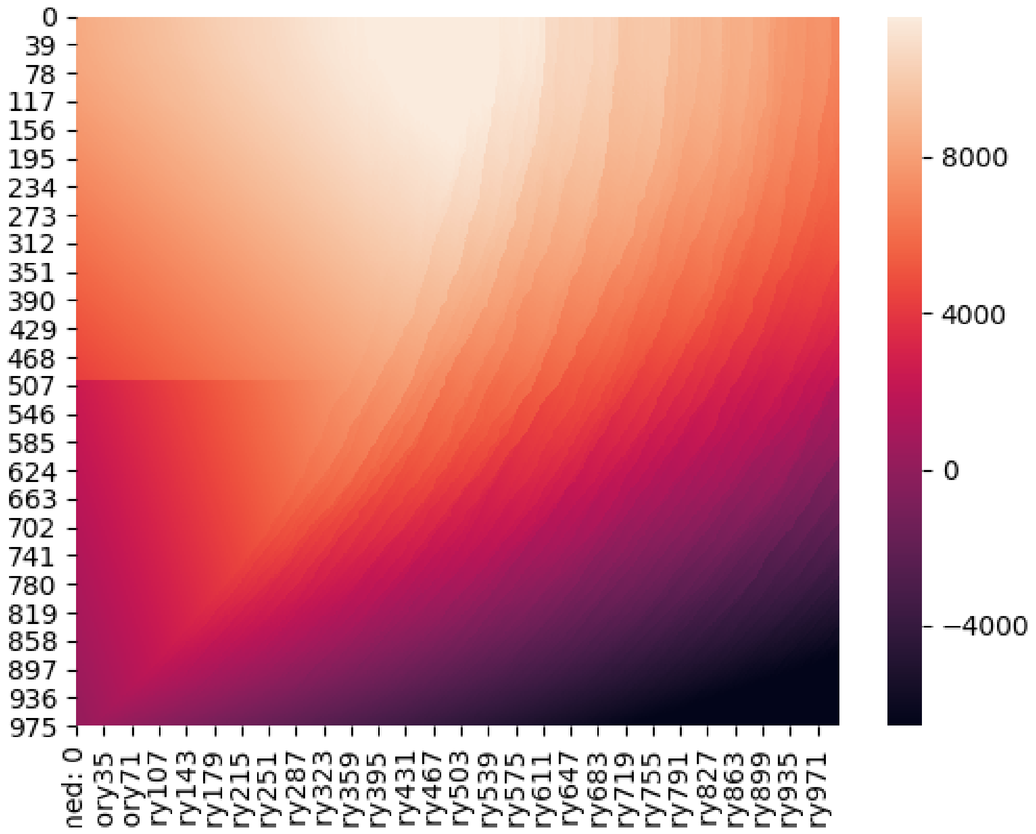

Figure 11: Profit heatmap for each time-inventory state of experiment nr.7 out of 10.

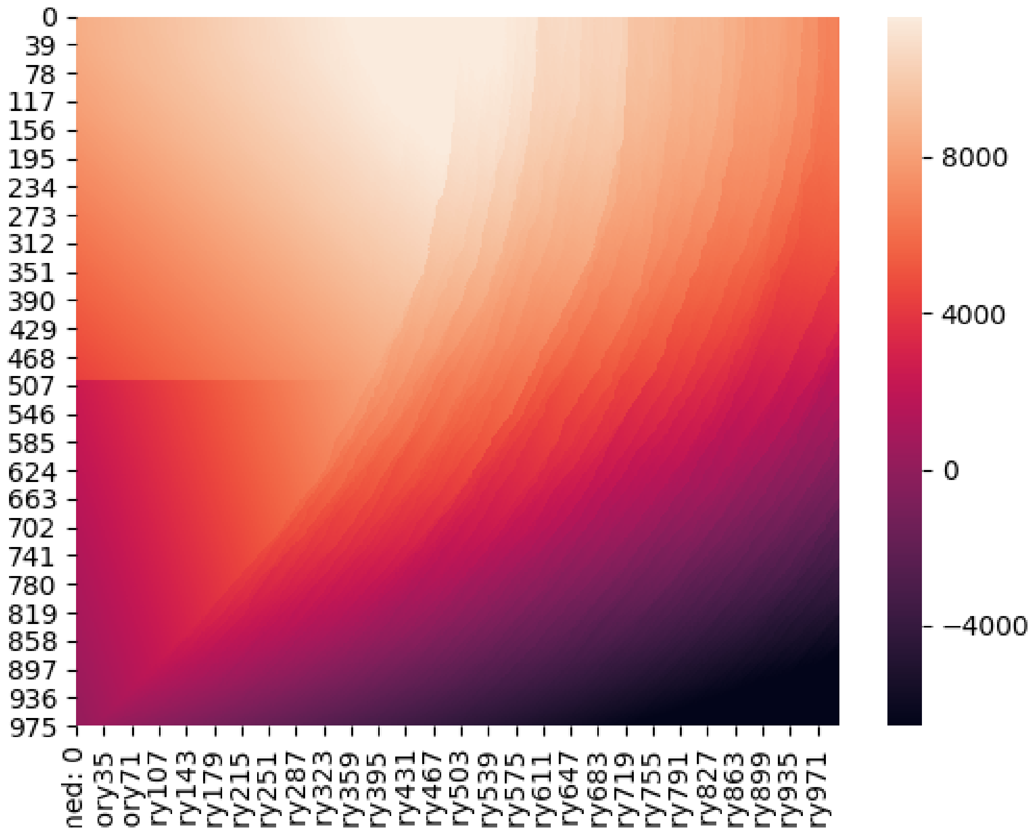

Figure 12: Profit heatmap for each time-inventory state of experiment nr.8 out of 10.

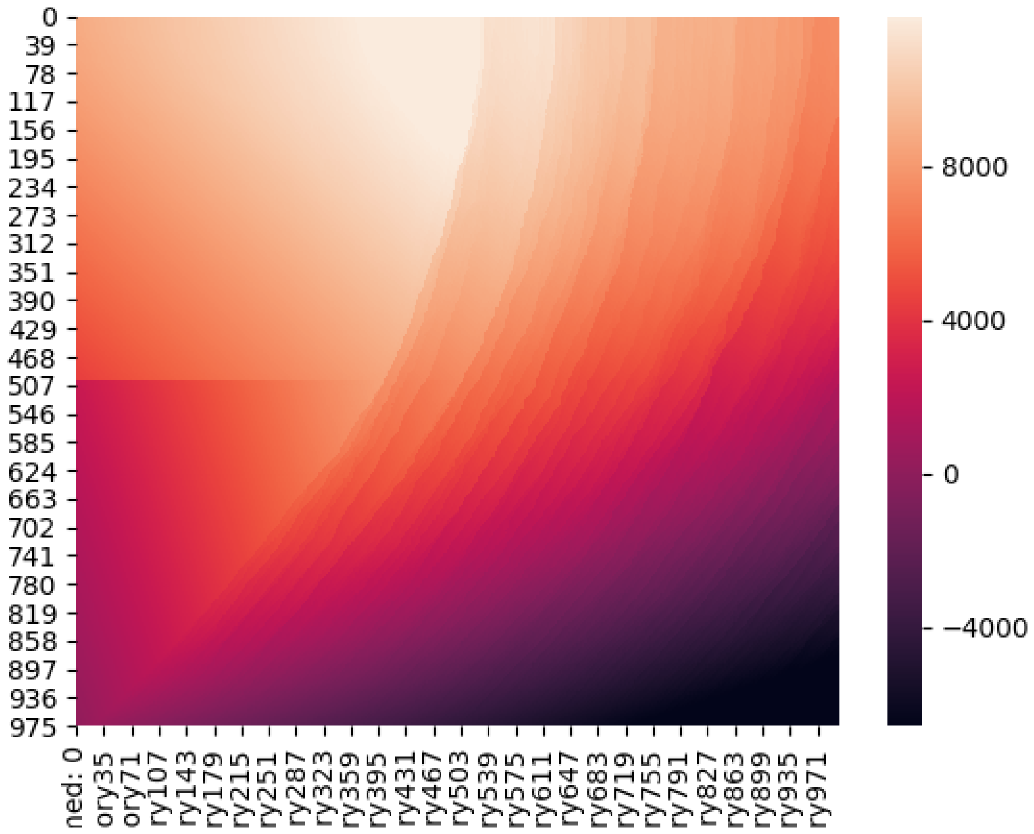

Figure 13: Profit heatmap for each time-inventory state of experiment nr.9 out of 10.

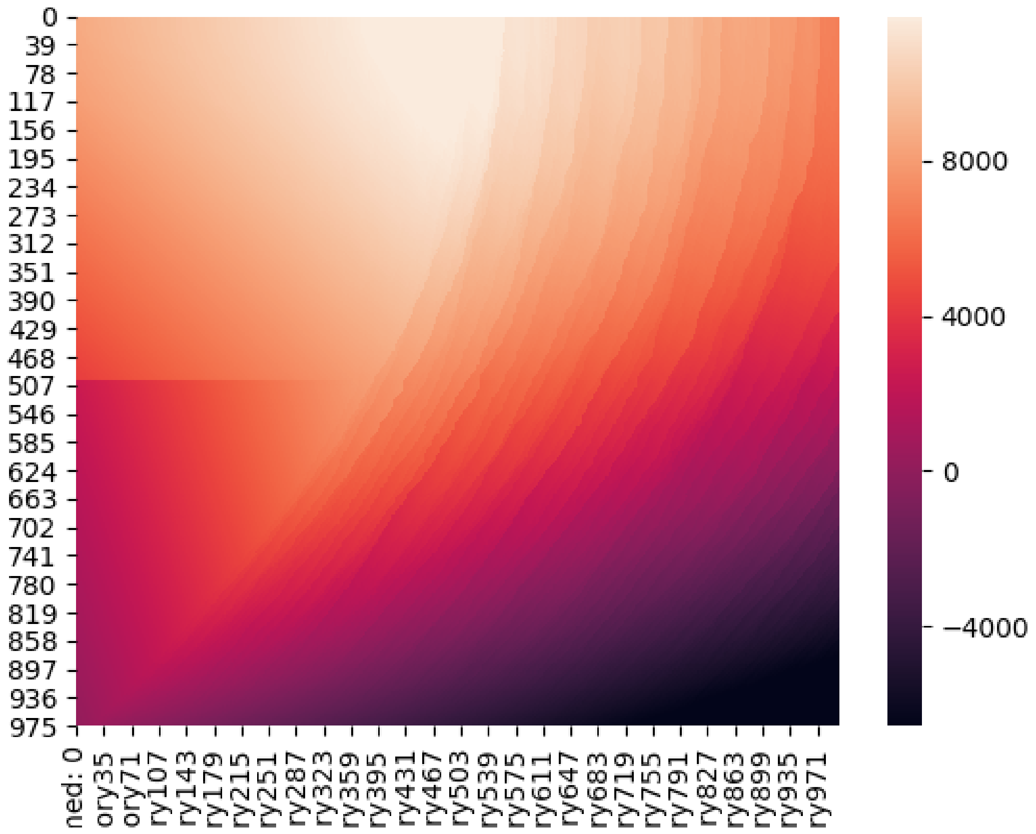

Figure 14: Profit heatmap for each time-inventory state of experiment nr.10 out of 10.

3.3 Order Heatmaps

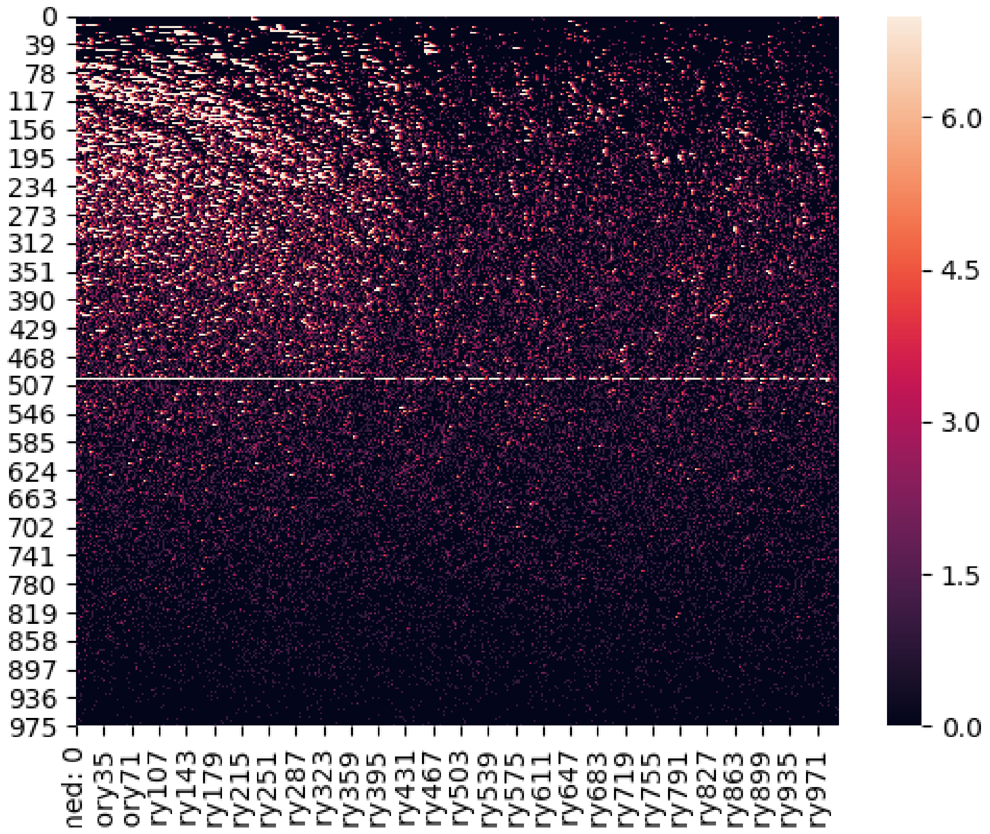

Figure 15: Order heatmap for each time-inventory state of experiment nr.1 out of 10.

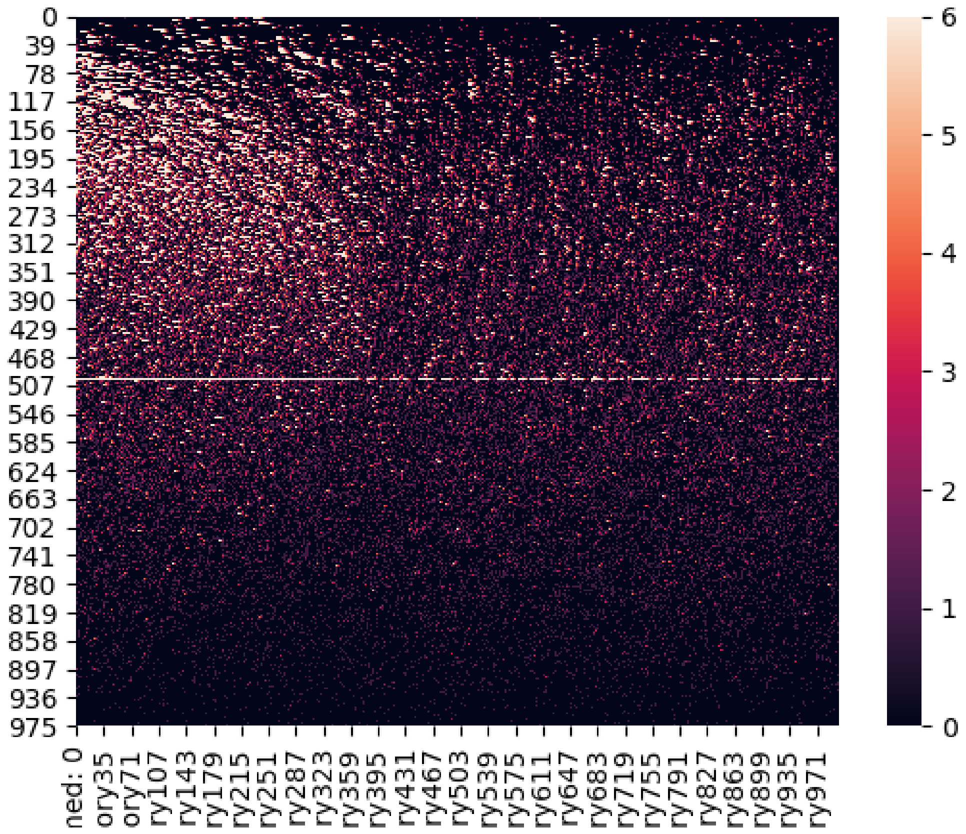

Figure 16: Order heatmap for each time-inventory state of experiment nr.2 out of 10.

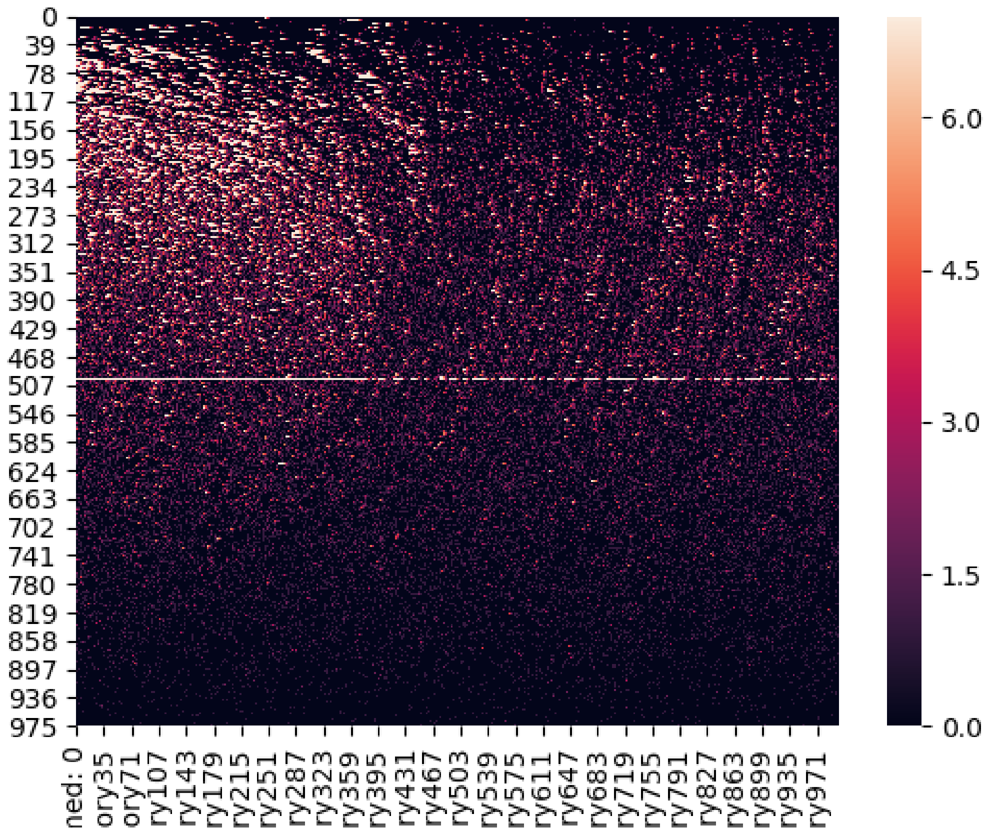

Figure 17: Order heatmap for each time-inventory state of experiment nr.3 out of 10.

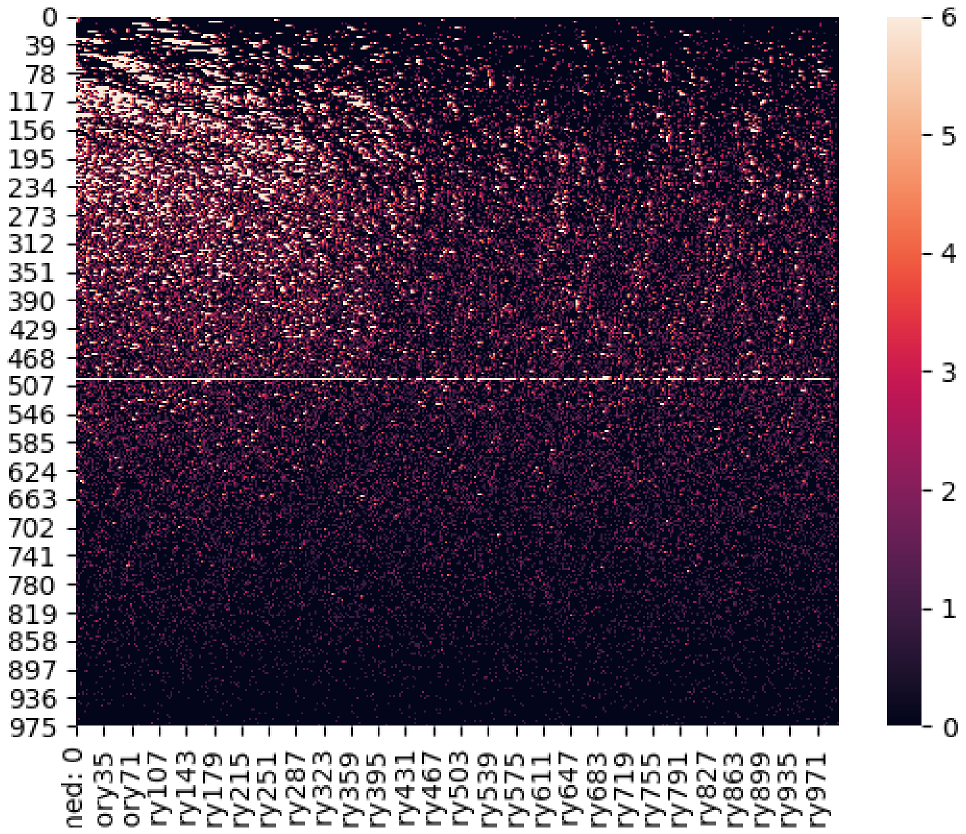

Figure 18: Order heatmap for each time-inventory state of experiment nr.4 out of 10.

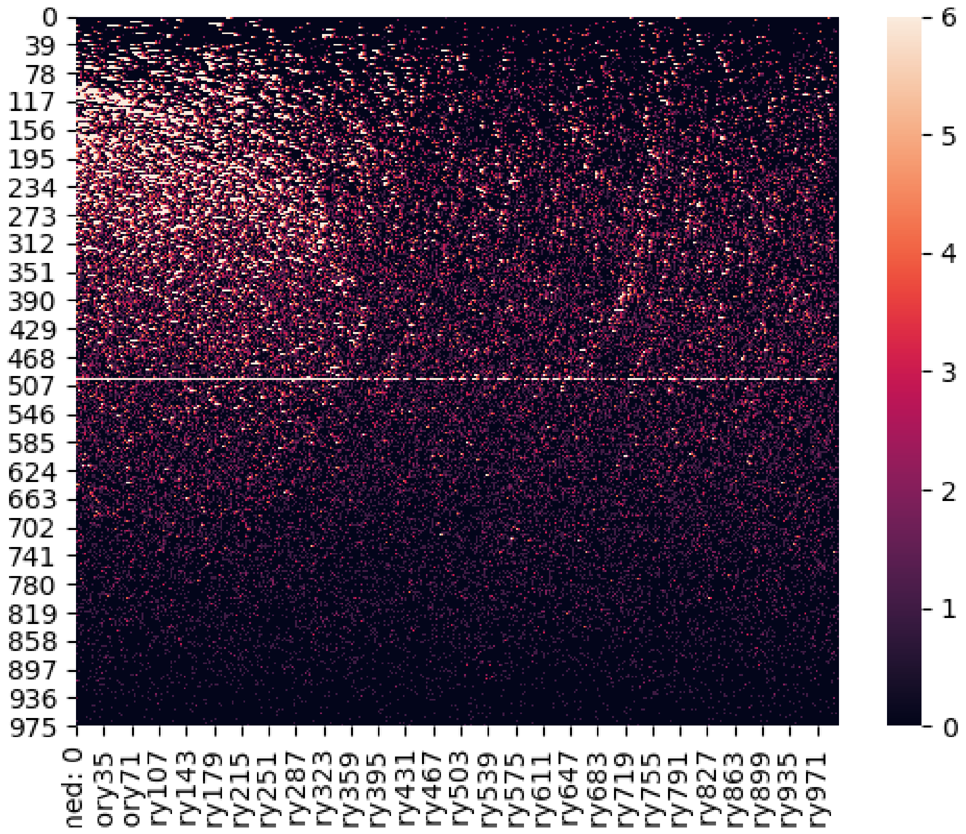

Figure 19: Order heatmap for each time-inventory state of experiment nr.5 out of 10.

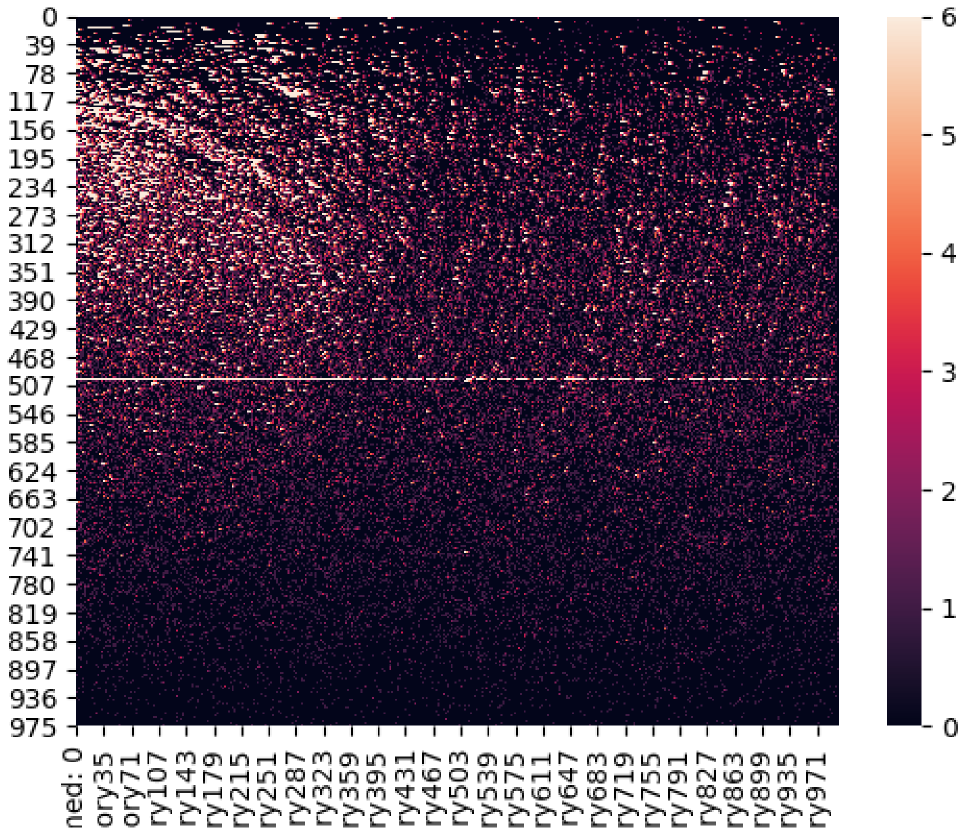

Figure 20: Order heatmap for each time-inventory state of experiment nr.6 out of 10.

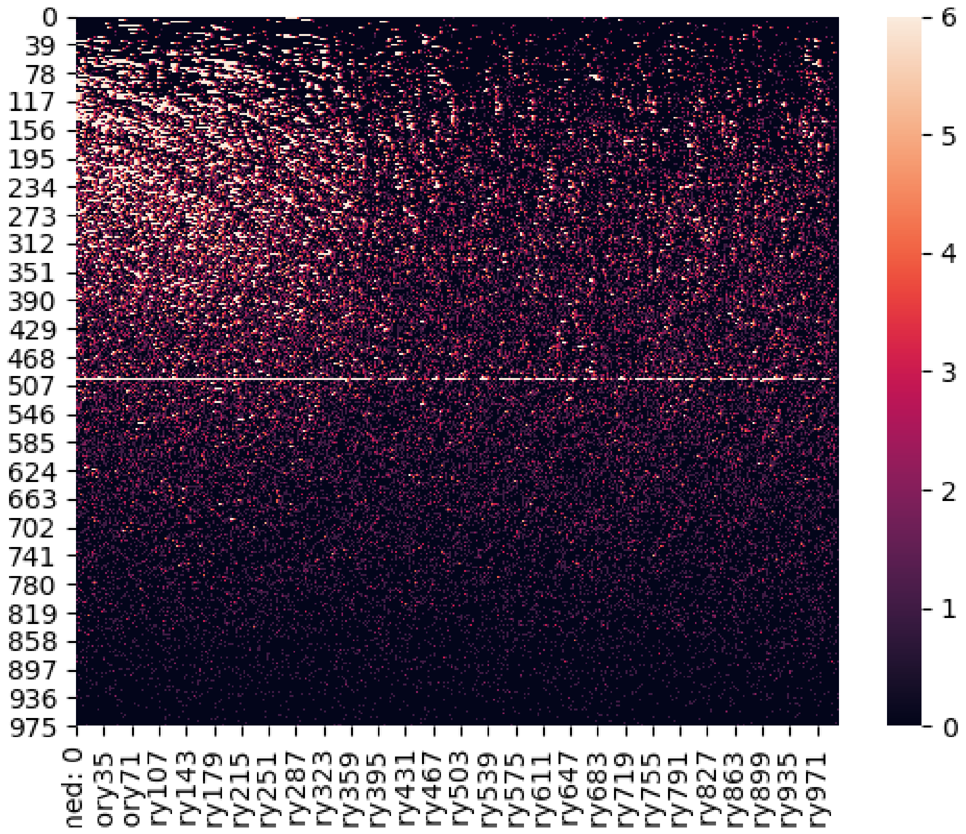

Figure 21: Order heatmap for each time-inventory state of experiment nr.7 out of 10.

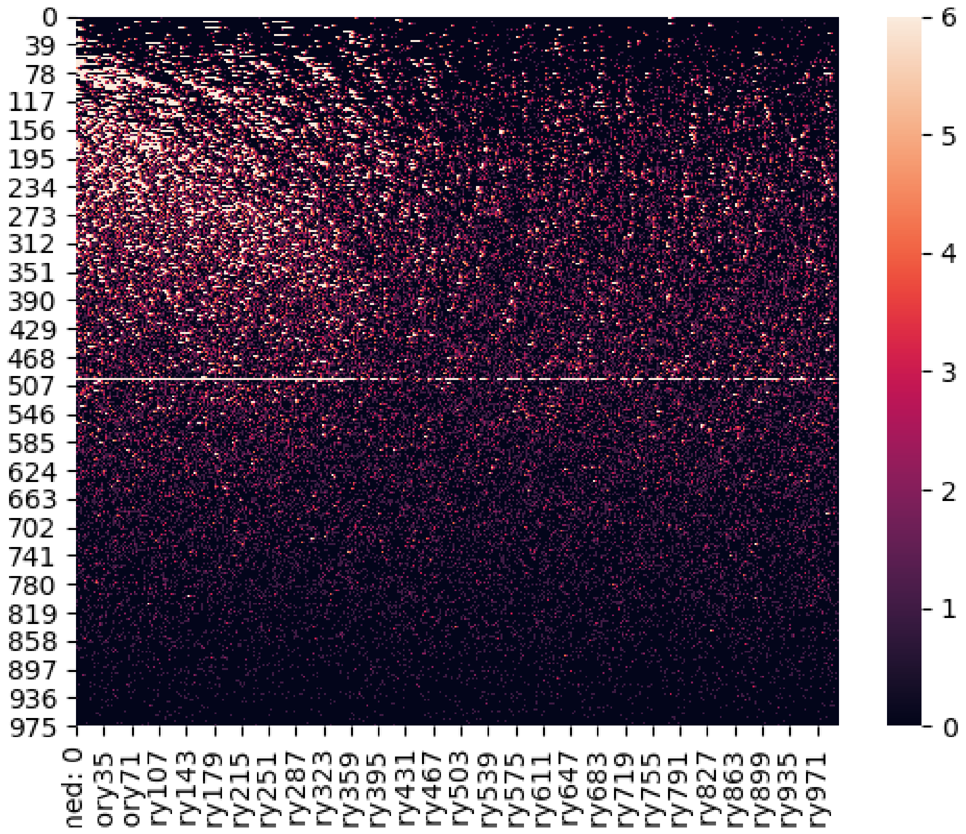

Figure 22: Order heatmap for each time-inventory state of experiment nr.8 out of 10.

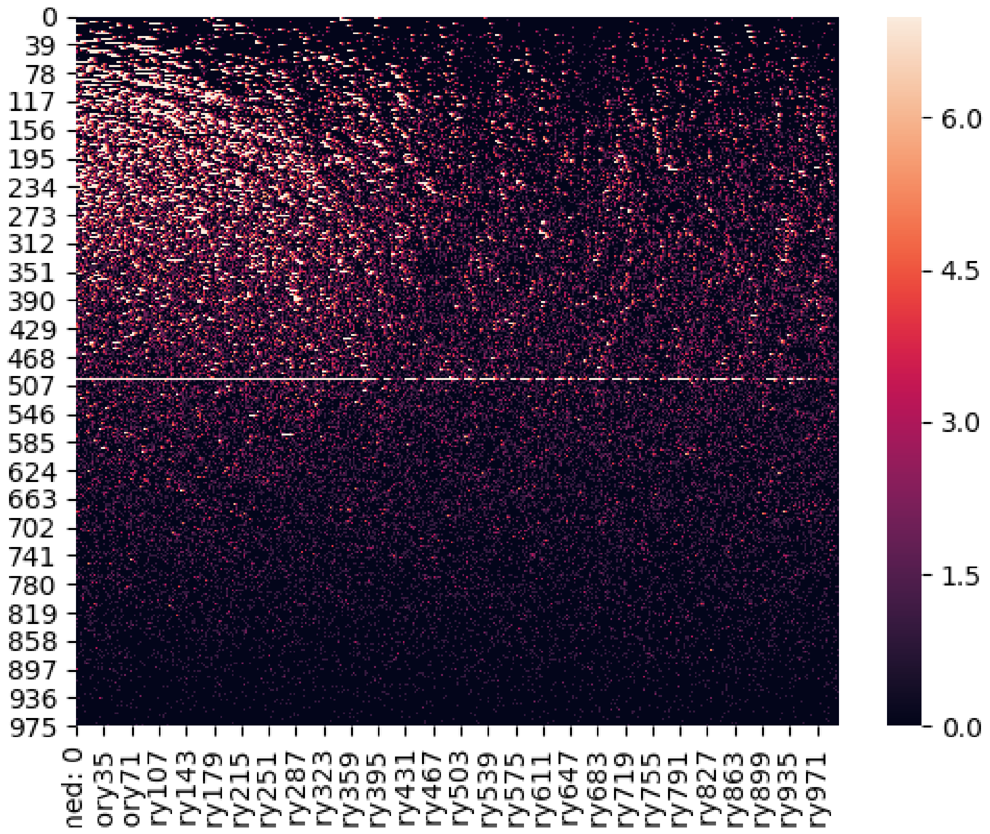

Figure 23: Order heatmap for each time-inventory state of experiment nr.9 out of 10.

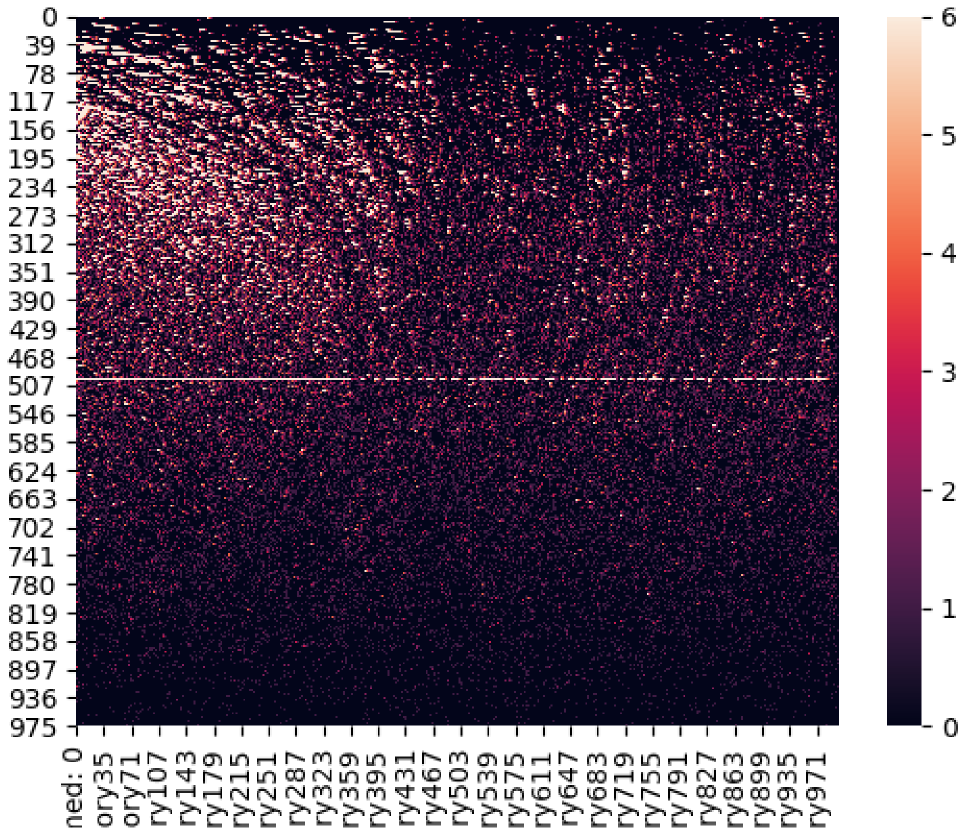

Figure 24: Order heatmap for each time-inventory state of experiment nr.10 out of 10.