ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis Econometrics and Operations Research

Strategic Open Routing in Service Systems with a Shared Station

July 7, 2019

Author:

M.N. van der Meij

Student number: 452989

Supervisor:

MSc C.D. van Oosterom

Second assessor:

MSc J.S.Vester

Abstract

An accurate prediction of the behaviour of strategic individuals in an open routing service system with a shared station can be of great use for many companies, an example is a restaurant with an open buffet. However, customer behaviour is hard to predict and system details tend to differ a lot. In the paper of Arlotto et al. (2019) an open routing system with two stations is simulated. They conclude customers show herding behaviour at the slowest station, to prevent being further back in the slow queue, as this could happen if they visit the faster station first. In this research, we first replicate their simulation and compare the results. In addition, we perform a similar simulation on a three-station subset open routing system with a shared station. Customers in this station visit only two of the three stations, but the shared station is always one of them. We divide the customers into two groups, one group visits one subset of the stations and the other group visits the other subset. The simulation shows individuals tend to herd at the shared station, given high enough service rates at the two specific stations.

Contents

1 Introduction

In many everyday environments, service systems can be found. In these systems, services are provided to customers at one or multiple stations. An example of this is a breakfast buffet, where a customer can independently choose to go to the bread station, beverage station or fruit station in whatever order they decide. As individuals are free to choose their path strategically, they can minimise the time they will spend waiting in the queues for these stations. Given that there are more customers who will be trying to do this, the question arises: What route should a customer choose?

The importance of researching this subject lies in the frequency of the occurrence of an open routing system. As described by Arlotto et al. (2019), these can be found in all sorts of environments such as amusements parks, festivals, shopping centres and buffets, but also in trials of medical research, as described by Baron et al. (2016). In the latter case, the individuals are not free to choose their own route but are given a schedule by a central planner. The central planner can have different objectives such as maximising customer satisfaction or minimising the total service time of the system. If one can get a clearer view of the behaviour of customers in an open routing system and how this compares to the optimal behaviour for a company, one can create a plan to increase customer satisfaction or to decrease the total system time. Furthermore, awareness under individuals can be created to inform them about the choice they should make to minimise their own waiting times as well as improve the cumulative system time.

There has been done previous research on open routing in service networks and several aspects have already been discovered and clarified. This is to be expected, as there is a great variety of different versions of open routing in service networks. We will analyse a three-station system with one shared station between two customer classes. Here, customers arrive simultaneously, but with priorities given to each customer. Furthermore, a numerical research will be performed on the same system, but in this case with stochastic arrivals. Firstly, we will reproduce the simulation performed in Arlotto et al. (2019). As their research focuses on a simulation with two stations with individuals who learn according to the historical average waiting times of both stations, there are still questions left to answer for similar systems with other aspects. Therefore we will extend the research of Arlotto et al. (2019) by analysing a model based on three stations, where a subset of two stations is visited by each individual. In our model, one station will be visited by each individual, while the other two stations will only be attended by part of the individuals. This setup is seen in multiple environments in practice, but yet little research is done for this construction. An example would be a buffet where there are separate stations for the drinks, warm and cold food. As customers are likely to either attend the cold or warm buffet, but visit the drinks station either way. These reasons and applications motivated us to analyse the following research question: "*How do strategic individuals in an open routing system where only a subset of the stations is visited, with a shared station between these subsets, behave?*" which brings the following sub-questions: "*How do service rates influence the behaviour of individuals in an open routing system?*" and "*Do individuals herd at their specific station?*" and finally "*Do individuals herd at the shared station?*".

Motivated by these questions, we first replicate the numerical simulation performed by Arlotto et al. (2019) and compare the achieved results with those of their simulation. We will then set up a simulation with three stations: A, B and C. Customers will visit shared station A and either B or C. The results of this numerical research give us a better understanding of the behaviour of individuals in open routing systems. We find customers tend to herd at the shared station when the service rates at the specific stations are high enough and the system is congested. Furthermore, as the system becomes less congested, customers show a slight decrease in herding behaviour. When the service rates of the specific stations are above that of the shared station, herding behaviour at the shared station is shown for all parameter combinations tested.

In Section 2 the relevant literature will be discussed, in Section 3 the two-station and three-station subset open routing games will be explained in further detail as well as the specifications of the systems we used in the simulations. Section 4 gives an in depth view of the two-station simulation as well as the three-station simulation. The parameters and structure used in the simulations as well as the hypothesis for these simulations will be discussed in this section. In Section 5, the output of the simulations is analysed and compared. Section 6 contains the conclusion of the paper and some final remarks and potential extensions for future research.

2 Literature

The research we do is closely related to literature of different aspects. We will go over some literature done on two-station networks, simulation, herding and finally a paper about multi-station routing. One of the few papers papers incorporating a stochastic network where customers choose the sequence of stations they visit, is Parlaktürk and Kumar (2004). They research a network, with two stations, where a "job" needs two tasks performed on it. Each station has a queue for both Task 1 and Task 2, where Task 1 has a shorter service time. Each station can perform only one task for each individual. As Task 1 is always executed first, the route decides which station performs which task for individuals i. The system planner can choose at each station which queue to serve next.

Depending on the scheduling rule chosen by the system planner, the cumulative service time decreases or increases. This can be caused by a better distribution of the already chosen routes of individuals, or by the differences in routes chosen by individuals, as they may change routes when the system planner implements a different scheduling rule. In the end, the researchers propose a scheduling rule where the self interested behaviour of the customers does not decrease the overall performance of the system. The contrast with the model researched by Arlotto et al. (2019) is that there are two queues at each station, instead of separate stations for each task. Furthermore, as there is only one server at each station, we assume the same serving method, namely first come first serve (FCFS), is applied to all customers for each station.

When we look at the existing literature about simulation-based study of routing schemes, the paper of Pinilla and Prinz (2003) gives a helpful insight for the numerical part of our research. They look into the standard sequential model and use simulation to receive insights in a flexible system. With their example of routing in a coffee shop they find that, when assigning the sequence of tasks dynamically compared to a fixed sequence of tasks, performance can be increased significantly. When we investigate the options to implement these results in our three-station system we see the options are minimal. The ability to determine which station to go next to after attending a station, is in our network not effective since there are only two stations to attend. This makes that there is only one station to attend to after having visited the first station. Therefore the insights of constructing a simulation to obtain empirical results are applicable, but we will not investigate the topic of flexible route choosing in further depth.

A previous research that found herding behaviour under customers is that of Veeraraghavan and Debo (2011, 2009). They looked into two competitive service providers where customers have private information about the quality of each provider. They find herding in cases where service rates are relatively high. As longer queues may insinuate better quality, uninformed individuals will join the longer queue and thus contribute to the herding strategy if they seek to optimise their utility. When comparing the results of the research to those of Arlotto et al. (2019), a similar aspect is finding herding as a equilibrium strategy. The difference occurs when we analyse the incentive behind the herding, as the customers are driven by the service quality and not the time spent in the queues. As the individuals in Arlotto et al. (2019) are assumed to try to minimise their expected time in the queue, starting at the less crowded station will be punished by a longer queue at the second station. Therefore, the both occurrences of herding have different causes.

When considering the three-station subset open routing game, there is the paper of Foss and Chernova (1998) which researches stability of multi-station systems which are partially accessible to each individual. This is a close representation of the idea of a customer visiting a set subset of a system. Foss and Chernova (1998) looks into three different situations where the system service times differ in each situation. They obtain simple stability criteria for two cases and further analyse the third case. An interesting approach is shown by using Markov process and chains to prove the stability criteria. Another aspect of the paper is the use of constant routing policies. Although multiple routing policies are studied, there is always a constant decision rule which does not implement an individuals historical information.

3 Open Routing Systems

Each open routing system has their own respective specifications such as number of stations, connections between the stations, service rates and many more. In this section we will discuss two types of open routing systems: the two-station open routing system and the three-station subset open routing system.

3.1 Two-station open routing game

Our first model, based on the model of Arlotto et al. (2019), is a two-station open routing system. In this model, the customers want to minimise their waiting time while still attending both stations. The stations, station A and station B, each have one queue with one server and nonidentical service rates μ_A and μ_B, respectively. Without a loss of generality, we assume the case of non-equal service rates ($\mu_A < \mu_B$). The customers are free to choose which station to attend first, but have to visit both stations exactly once. A FCFS policy is applied to serve the queues, as this is also maintained in many service environments in practice. The resulting network is shown in Figure 1.

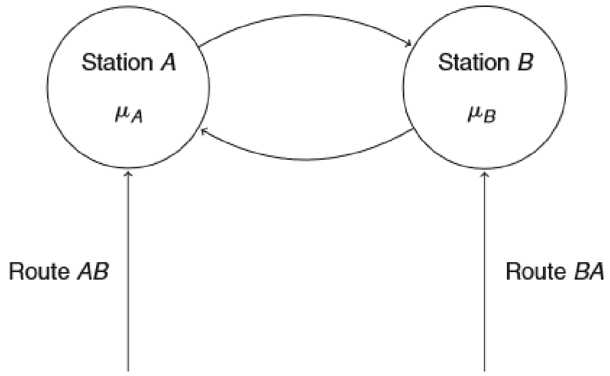

Figure 1: Customers who follow AB will first visit station A and than B, Arlotto et al. (2019)

The paper of Arlotto et al. (2019) gives useful insights about the equilibrium behaviour of customers in a two-station open routing system. In this system, customers choose a route simultaneously, receive a randomised priority and service times are deterministic. It shows that given that there are enough players in the system, herding equilibria are the only pure strategy Nash equilibria. Even when non-strategic customers, who have a pre-determined route independent of the queues, or stochastic service times are introduced, herding is still an equilibrium strategy profile. When customers who only attend one of the two stations are introduced in the game, herding behaviour stays an equilibrium strategy profile. The note has to be made that in this case it is necessary to have enough strategic players in the game compared to the customers who attend only one station. When the game is played sequentially it is shown that herding at station A, the slower station, is the prevailing strategy. The motivation is that when a player first visits station B, his position will be overtaken and he will end more to the back of the slower queue at station A. The individual with the last priority, will join the queue at station B first, as he is last in line for station A already. In the case of a system with more than two stations, herding is still a Nash equilibrium. In this case the herd visits the stations according to the increasing service rates. The customers start at the slowest station and make their way to the fastest serving station. Arlotto et al. (2019) have shown the conditions and boundaries of this specific herding strategy profile, but do not exclude the possibility of other equilibrium strategy profiles existing for a system with more than two stations.

Furthermore, Arlotto et al. (2019) have found that the cumulative system time, the time it takes to serve all customers, is close to the optimal time when herding is applied. Finally, they looked into an example of a non-congested system which reached a steady state. In this system customers arrive over time and the arrival rate is lower than the slowest service rate of the stations. For this example, herding profiles are still equilibria. Although, any other feasible routing profile is also an equilibrium. Therefore herding behaviour is not necessary prevailing in a system that does not overflow over time.

3.2 Three-station subset open routing game

The research of Arlotto et al. (2019) suggests that studying a system with more than two stations, where each customer visits only a subset of the stations, may be interesting. We adopt this idea to a system with three stations, station A, B and C, where each customer has to visit station A and either B or C, depending on the class of the customer. We will split the customers in two classes, S_{ab} and S_{ac}, where individuals in S_{ab} will visit station A and B and those in S_{ac} visit A and C. A visual representation of the two service systems with a shared station is given in Figure 2.

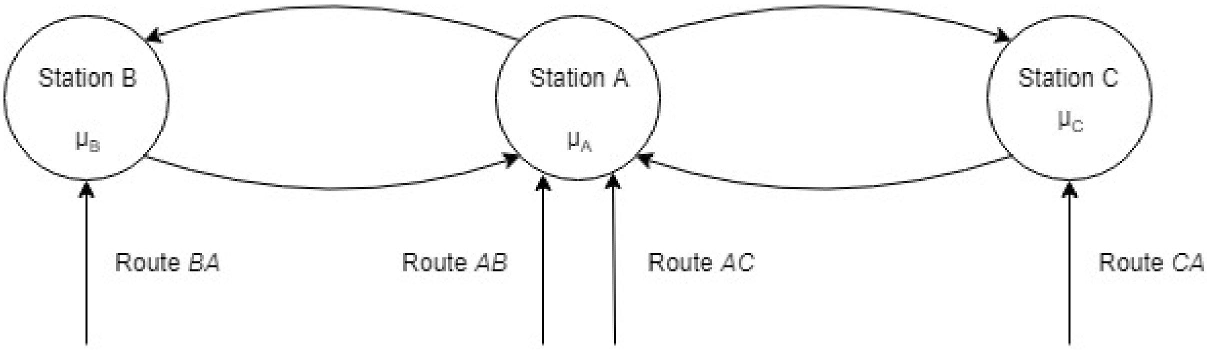

Figure 2: Customers in class S_{ab} will either visit station A first, if they take route AB, or B if they take route BA

The customers are free to choose which station they attend first, but there is still a one queue FCFS policy at each station. In case of equal arrival times of a customer from class S_{ab} and S_{ac} at the same station, class S_{ab} will be served first. This leads to two separate groups of possible routes: AB and BA for the customers in class S_{ab} and route AC and CA for those in S_{ac}. Therefore, multiple herding strategies are possible. The first one would be herding at station A, where every customer first visits station A. One of the other possibilities would be herding at the specific stations. Customers in class S_{ab} and S_{ac} would take route BA and CA, respectively. Furthermore, a possible herding strategy profile would be one where one class starts at station A, while the other class visits their specific station first.

We still assume inequality in service rates. As we consider equally large classes of customers, we set $\mu_B > \mu_C$, without a loss of generality, to prevent repetition of results with different station labels. This gives room for three different set-ups considering the service rates of the station:

- $\mu_A > \mu_B > \mu_C$
- $\mu_B > \mu_A > \mu_C$
- $\mu_B > \mu_C > \mu_A$

which gives different combinations of slowest and fastest stations. This could be an important factor in our multi-station subset open routing game, as the results of Arlotto et al. (2019) have shown us herding at the slowest station is an equilibrium strategy profile. Even though their system structure was different, it may have similarities in outcome.

When all customers arrive at the same time, but priorities are still given uniformly at random, the number of customers N is large enough and the service times are deterministic, there is an equilibrium strategy profile where all customers start at the shared station A.

Proposition 1 (Herding Equilibrium in the Three-Station Subset Open Routing Game). *For $N \geq 2\mu_A/\mu_C + 1$ and $\mu_A < \mu_C < \mu_B$, the open routing game with a shared station has a Nash Equilibrium in which all players visit station A first.*

The proof for this assumption is given in Appendix A and based upon the proof of Proposition 1 in Arlotto et al. (2019). When this situation is analysed for the case with stochastic service times, herding at the shared station is found to be a symmetric Nash equilibrium for all customers in both classes.

Proposition 2 (Nash Equilibrium with Visiting Shared Station A first in Case of Stochastic Service Times). *For $N > 1 + \frac{2\mu_A}{\mu_C} + \frac{\mu_A^2(\sigma_A^2+\sigma_C^2)}{2-\frac{\mu_A}{\mu_C}}$, it is a symmetric Nash Equilibrium for all customers in class S_{ab} and S_{ac} to visit station A first, which implies they follow route AB and AC, respectively.*

The proof for this assumption can be found in Appendix A and is inspired by the proof of Proposition 5 in the paper of Arlotto et al. (2019).

4 Methodology

In this section we will first describe how we replicated the simulation performed by Arlotto et al. (2019) with the two-station open routing system. Secondly, the simulation of the three-station subset open routing game will be explained. The results of these simulations will be discussed in Section 5.

4.1 Two-station simulation

In the paper of Arlotto et al. (2019) the differences of equilibria strategies between congested and non-congested systems are discussed. They state that in systems that are not very busy, herding strategies do not predominate compared to other strategies. This brought them to the following hypothesis: "*Herding occurs when a service system is congested, that is, the arrival rate is higher than the service rates of both stations until the arrival of the last customer*" where the practical benefits of confirming this hypothesis are significant as systems are often congested at the start of their service availability. Furthermore, even though the analytical results given by Arlotto et al. (2019) are theoretically correct, they argue that it is highly unlikely that individuals perform such analysis before deciding which queue to join. Therefore they decided to perform a simulation where individuals learn through repeated rounds in the system. The arrival and service times were both stochastic. The results of the simulation give an insight in the customer behaviour in a two-station open routing system. This information is most interesting for us, as we will perform a simulation with a three-station service system, where customers visit only two stations. Individuals play multiple rounds in the system and choose the route which they believe has the shortest expected waiting time. After the round they learn the waiting time of both the chosen route as well as the other route. After learning this their beliefs update to the empirical average of their own samples for both routes. In case of an equal average waiting time for both routes, a random route will be chosen. For the first round, a random route is chosen as well.

The arrival times are defined by γ and ϕ, which represent the mean and variance of the arrival times. The arrival time of individual $i \in \{1, ..., N\}$ will be assumed to be uniformly distributed on $[i\gamma - \phi, i\gamma + \phi]$. This leads to mutually independent arrivals of different customers, but leaves the opportunity of different priorities if ϕ is large enough, as overlap may occur when $\phi > \gamma/2$. The service rates of the stations are defined as μ_A and μ_B for station A and B respectively. These are taken to be distributed exponentially, where station B is assumed to be the slower station with fixed $\mu_B = 1$ and $\mu_A < 1$. The number of individuals N will stay fixed at $N = 50$. We will simulate all combinations of $\gamma \in \{0.001, 0.1, 0.25, 0.5, 0.75, 1\}$, $\phi \in \{0, 0.25, 0.5, 0.75, 1\}$ and $\mu_A \in \{0.1, 0.25, 0.5, 0.75\}$ which leads to a total of 120 parameter sets. For each parameter set 100 independent trials are performed, with each up to 250 rounds of play in each trial. The trial will be stopped whenever all individuals have chosen the same route 50 times, to decrease computation time. This condition will bias the results a little bit towards herding behaviour, as the round stops whenever complete herding occurs for 50 routes.

However, the effect will be small as it is unlikely individuals will switch routes when all individuals choose the same route for such a period. The largest expected bias will be created by the last customer, as is may be efficient for her not to herd. The gain in computation time is expected to be of a factor 10, which is based on a test run for comparison.

The results of this simulation will be compared to those of Arlotto et al. (2019), where the final null hypothesis will be: "*The number of customers choosing route AB in the final round of the simulation has the same distribution as that of the simulation performed by Arlotto et al. (2019)*". To test this, we will first test the equality of the mean number of customers choosing AB in the last round for each parameter combination, for our simulation and the one performed by Arlotto et al. (2019). The two-sample t-test will be used to test this equality, with the null hypothesis: "*The average number of customers choosing route AB at the end of the trial, for this parameter set, is equal for both simulations*", where this test will be performed for each parameter set shown in Arlotto et al. (2019). The test statistic will be calculated as follows:

$$T_i = \frac{\bar{X}_i - \bar{Y}_i}{s(X_i)} \sqrt{\frac{N}{2}}, \tag{1}$$

where X_i is the average of all trials for parameter set i, Y_i the average given by Arlotto et al. (2019), $s(X_i)$ the sample standard deviation of the trails for parameter set i and N the number of trials, which is 100 in both simulations. This also brings that we use 198 degrees of freedom in our tests. The results of these lower stage tests, which use a 95% confidence level, will be used as inputs for a binomial test to confirm or deny the main hypothesis. Even though the Jarque-Bera test rejects the null hypothesis of Normality for the distribution of the number of individuals choosing route AB in each trial, the averages of these trails may still be normally distributed. This, combined with the Central Limit Theorem makes that we can still assume normality and therefore apply the two-sample t-test.

4.2 Three-station subset simulation

For the same reasons as above, a simulation of the three-station subset open routing game is constructed where the hypothesis: "*When the system is congested, herding occurs at either the shared station A, or at each class' specific station*", is tested. Considering the parameter sets for this system, we have $\gamma \in \{0.001, 0.1, 0.25, 0.5, 0.75, 1\}$, $\phi \in \{0, 0.25, 0.5, 0.75, 1\}$ and $(\mu_B, \mu_C) \in \{$(0.25,0.1), (0.75,0.5), (1.25,0.75), (1.75,0.25), (1.25,1.1), (1.75,1.5)$\}$. This gives for each combination of μ's described in Section 3.2, two different situations with varying values for μ_B and μ_C. The total number of parameter combinations is 180. For each of these parameter combinations 100 independent trials will be executed, where individuals play up to 250 rounds in each trial.

5 Results

We will discuss the results from the two-station simulation in Subsection 5.1 and those of the three-station subset simulation in Subsection 5.2. For both simulations we analyse the number of people who choose station A as their first station to visit and for the three-station simulation we will analyse the number of individuals attending A first per class as well. Furthermore, the results of the two-station simulation will be compared to those of Arlotto et al. (2019).

5.1 Two-station simulation outcome

The results of the simulation with $\gamma = 0.001$ and $\phi = 0$ are shown for multiple values of μ_A in Figure 3. This shows the results of the system where all customers arrive in a set order, as $\phi = 0$, and with very little time between arrivals, as $\gamma = 0.001$. The situation closely resembles the analytically researched system in Arlotto et al. (2019), in which all customers are all present at the start and choose their routes sequentially. This differs from the simulation performed, as individuals cannot observe each others decisions but choose their route based on the historical waiting times. It can be seen the number of trails ending with all individuals choosing route AB increases as μ_A increases. Furthermore, there are no occurrences where less than 40 customers choose route AB, which is in line with the statements made by Arlotto et al. (2019), even though the conditions of the system differ from those of system the propositions are based on. The herding at station A can be explained by customers experience from previous rounds, where they encountered a longer waiting time if they attended station B first due to being overtaken at station A by later arrivals.

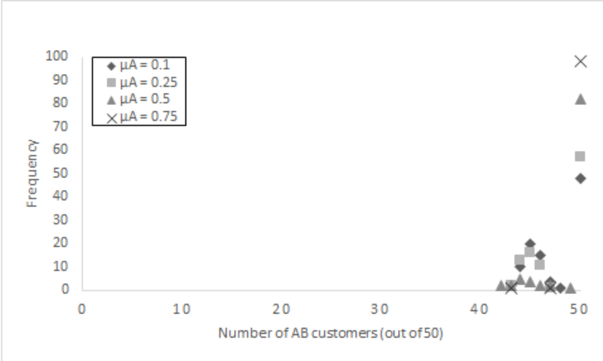

Figure 3: Frequency Chart for Number of AB Customers in the Final Round (100 trials in total), for $\gamma = 0.001$ and $\phi = 0$

In Table 1 and Table 2, the resulting number of individuals choosing route AB in the final round is shown. The γ values chosen for these tables represent arrivals more spread out over time, instead of the $\gamma = 0.001$. This is motivated based on the application in practice, since $\gamma = 0.001$ would mean each individual arrives almost at the same time, which is unlikely in a real world scenario. We fix μ_A at 0.75 and 0.5 for the tables and vary γ and ϕ between zero and one. Because herding never seems to arise for route BA, for any of the parameter sets in the tables, the number of customers who choose route AB can be used as a measure for herding. The right side of the tables give the values for the first quartile of the results.

We note the number of final round choices for AB seems to decrease as γ increases. This can be explained due to more idle time at the stations. When inter-arrival times increase, the probability of a customer arriving at the service system when there are no busy stations increases. As this would bring equal waiting times for both routes, more customers choose a route randomly, which leads to more customers taking route BA. There is clear herding behaviour for all parameter settings, given $\gamma \geq 0.1$, since all average results are over 42, which represents 84% of the individuals, and for $\mu_A = 0.5$ the averages are even over 48, which accounts for 96% of the customers. Furthermore, there seems to be a slight difference due to the decreasing value of μ_A, when comparing Table 2 and Table 1. The results of the simulation with $\mu_A = 0.5$ tend to be slightly higher, which would be in line with the theory of increasing herding behaviour when the differences in service rates increase as described by Arlotto et al. (2019). When looked into these differences in more detail, we see a smaller decrease as γ increases compared to the decrease in Table 1. This would be in line with the theory of decreasing herding behaviour as the system is less congested due to more equilibrium strategy profiles existing in a non-congested system. With slower service rates, 0.5 instead of 0.75, the system is more congested for equal γ values. Therefore, herding behaviour appears more than in Table 1.

Sample mean						Sample first quartile					
	ϕ						ϕ				
γ	0	0.25	0.5	0.75	1	γ	0	0.25	0.5	0.75	1
0.1	49.25	49.37	49.45	49.52	49.41	0.1	50	50	50	50	50
0.25	49.19	49.36	49.25	49.31	49.41	0.25	50	50	49	50	50
0.5	48.8	48.87	48.88	48.64	48.96	0.5	49	49	49	48	49
0.75	46.99	47.9	48.3	48.14	47.98	0.75	46.75	48	47.75	47	47.75
1	43.71	43.8	42.59	43	42.93	1	41	41	41	40	40.75

Table 1: Average number of AB customers (out of 50) in the final round, with μ_A=0.75

Sample mean						Sample first quartile					
	ϕ						ϕ				
γ	0	0.25	0.5	0.75	1	γ	0	0.25	0.5	0.75	1
0.1	49.56	49.49	49.52	49.63	49.6	1	50	49	49	50	50
0.25	49.09	49.58	49.4	49.82	49.35	0.75	49	50	50	50	50
0.5	49.29	49.43	49.09	49.38	49.51	0.5	50	50	50	50	50
0.75	49.11	49.1	49.17	48.98	48.99	0.25	50	50	50	50	50
1	48.65	48.47	48.69	49.05	48.65	0.1	49.75	46.75	49.75	50	49

Table 2: Average number of AB customers (out of 50) in the final round, with μ_A=0.5

When we compare the results of our simulation to the results of Arlotto et al. (2019), which are given in Appendix B, we test whether the number of AB customers in the final round differ significantly by means of a two-sample t-test. At a 5% significance level, the results of only 5 parameter sets are not rejected for the null hypothesis of equality. When we test the number of accepted null hypothesis in our main test with null hypothesis:*"The number of customers choosing route AB in the final round of the simulation has the same distribution as that of the simulation performed by Arlotto et al. (2019)"* we use a Binomial test with 5 as the number of successful trials, 50 as the total number of trials and 95% as the probability of success. This leaves space for a 5% error margin which is required since we work with simulation results where random numbers are involved. In case of a 100% success probability it would mean all parameter sets needed an accepted null hypothesis. The result of the Binomial test gives a probability smaller than 0.001, which makes that we reject the null hypothesis. The differences in results may occur due to decisions made within the simulation, for example the cut-off condition as described in Subsection 4.1.

Another reason may be a difference in simulation structure, which could be caused by an unspecified detail in the paper of Arlotto et al. (2019).

5.2 Three-station subset routing behaviour

Figure 4 shows the results of the three-station open routing game simulation, where the empirical frequencies for the number of individuals who choose to visit station A first are represented, for three combinations of (μ_B, μ_C). The values of γ and ϕ are set to 0.001 and 0, respectively. Similar to Figure 3 in Section 5.1, this closely represents the scenario of all individuals arriving simultaneously, but with different priorities. If we compare the simulation results to Proposition 2, which states herding at the shared station is an equilibrium strategy profile under certain conditions, similar outcomes are found. As the number of individuals visiting A first is concentrated around 50 for all three combinations of (μ_B,μ_c). Furthermore, as μ_B and μ_c increase, the number of trials resulting in all customers choosing to visit A first increases as well. This could imply the incentive for herding behaviour is dependent on the service rates for the other stations. Another dependency of this can be found in the conditions for N in Proposition 1 and 2, as the number of individuals necessary for herding at station A to be an equilibrium strategy profile decreases as μ_C increases.

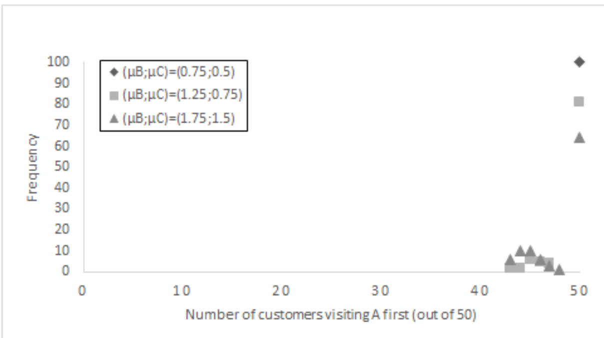

Figure 4: Frequency Chart for Number of Customers starting at station A in the Final Round (100 trials in total)

In Table 3 and Table 4, the results of the three-station subset simulation are shown for μ_B and μ_C equal to (0.25, 0.1) and (1.75, 0.25), respectively. When analysing the results in Table 3, there seems to be no herding at station A, but at the specific stations for $\gamma <$ 0.5. There is still an increase in value as γ increases from 0.1 to 0.25, but there remains a clear sign of herding at the specific stations. For $\gamma > 0.25$, the individuals from class S_{ac} herd at station A, while the other class still starts at their specific station. For all other combinations of parameters, the results are comparable to those shown in Table 4. There is a strong trend of herding behaviour for each $\gamma \geq 0.1$ as well as for ϕ in [0,1]. When we compare the results of the numerical analysis with Proposition 2, we note that they are in line with each other even though the arrivals are stochastic instead of at the same time.

Sample mean						Sample first quartile					
	ϕ						ϕ				
γ	0	0.25	0.5	0.75	1	γ	0	0.25	0.5	0.75	1
0.1	3.89	4.04	4.17	4.58	4.33	0.1	3	3	3	4	4
0.25	7.8	7.88	8.06	8.03	8.36	0.25	7	7	7	7	8
0.5	24.14	24.67	24.93	24.85	24.81	0.5	23	24	25	25	24.75
0.75	25.76	25.64	25.66	25.75	25.72	0.75	25	25	25	25	25
1	26.86	26.8	26.69	26.78	26.72	1	26	26	26	26	26

Table 3: Average number of customers visiting A first (out of 50) in the final round, with μ_B=0.25 and μ_C=0.1

Sample mean						Sample first quartile					
	ϕ						ϕ				
γ	0	0.25	0.5	0.75	1	γ	0	0.25	0.5	0.75	1
0.1	49.73	49.86	49.73	49.78	49.74	0.1	50	50	50	50	50
0.25	49.98	49.94	49.9	49.91	49.92	0.25	50	50	50	50	50
0.5	49.95	49.95	49.96	49.97	49.97	0.5	50	50	50	50	50
0.75	48.83	49.04	48.9	49.11	49.1	0.75	48	48	48	48	48
1	45.27	45.38	45.3	45.25	45.03	1	44	44	44	44	44

Table 4: Average number of customers visiting A first (out of 50) in the final round, with μ_B=1.75 and μ_C=0.25

A separate representation of the individuals per class attending station A first is given in Table 5 and Table 6 for (μ_B, μ_C)=(0.25,0.1) and (μ_B, μ_C)=(1.75,0.25), respectively. In Table 5 a slight increase in customers from S_{ab} visiting A first can be seen as γ increases, while for customers in class S_{ac} this seems to be a much stronger increase, as it ranges from 3.09 to 25 for $\phi = 0$. Furthermore, there is a clear difference between the behaviour of the two classes. Class S_{ab} shows clear herding behaviour at station B, as there are less than two customers on average choosing route AB for all parameter sets, while class S_{ac} tends to shift from visiting station C first, to herding at station A as γ increases.

Table 6 gives a representation of a (μ_B, μ_C) set where for all values of γ and ϕ individuals from both classes show a clear herding behaviour. As γ increases, the number of individuals in class S_{ab} visiting A first decreases, which is as expected as the system becomes less busy. For the individuals in class S_{ac} this is not the case, as μ_C is lower than μ_B. This causes station C to remain busy even though the increasing inter-arrival times.

Class S_{ab}						Class S_{ac}					
	ϕ						ϕ				
γ	0	0.25	0.5	0.75	1	γ	0	0.25	0.5	0.75	1
0.1	0.09	0.08	0.08	0.13	0.14	0.1	3.8	3.96	4.09	4.45	4.19
0.25	0.07	0.02	0.03	0.04	0.01	0.25	7.73	7.86	8.03	7.99	8.35
0.5	0.24	0.27	0.36	0.33	0.41	0.5	23.9	24.4	24.57	24.52	24.4
0.75	0.77	0.64	0.66	0.75	0.72	0.75	24.99	25	25	25	25
1	1.86	1.8	1.69	1.78	1.72	1	25	25	25	25	25

Table 5: Average number of customers visiting A first per class in the final round, with μ_B=0.25 and μ_C=0.1

Class S_{ab}						Class S_{ac}					
	ϕ						ϕ				
γ	0	0.25	0.5	0.75	1	γ	0	0.25	0.5	0.75	1
0.1	24.81	24.86	24.78	24.81	24.76	0.1	24.92	25	24.95	24.97	24.98
0.25	24.98	24.94	24.9	24.91	24.92	0.25	25	25	25	25	25
0.5	24.95	24.95	24.96	24.97	24.97	0.5	25	25	25	25	25
0.75	23.83	24.04	23.9	24.11	24.1	0.75	25	25	25	25	25
1	20.27	20.38	20.3	20.25	20.03	1	25	25	25	25	25

Table 6: Average number of customers visiting A first per class in the final round, with μ_B=1.75 and μ_C=0.25

6 Conclusion

Firstly, a replication of the two-station open routing game simulation in Arlotto et al. (2019) is performed. These results are compared and it is found that the average number of customers choosing route AB in our simulation follows a different distribution than that of the results of the simulation performed by Arlotto et al. (2019). Analyses of the simulation results shows that herding behaviour appears as an equilibrium strategy when the arrival rate is higher than the lowest service rate. As the inter-arrival times increase and thus the system becomes less busy, customers tend to spread out over both routes.

Therefore, it is suggested herding occurs when a service system is congested.

Secondly, a numerical research is performed on a three-station subset open routing game, where customer visit one specific station and a shared station. The results of the simulation performed with this system show us the influence of service rates, inter-arrival times and changes in priorities on the routing behaviour. With these results we can answer the question "*How do strategic individuals in an open routing system where only a subset of the stations is visited, with a shared station between these subsets, behave?*" We see when the service rates of the two specific stations are sufficiently large compared to that of the shared station, customers of both groups herd at the shared station. When the service rates are not sufficiently large, customers herd at their specific station, or the case arises where one class herds at their specific station and the other class at the shared station. Another parameter of influence on the behaviour of customers is the inter-arrival times, as it seems a less congested system gives room for other strategy profiles and customers show a little less herding behaviour. These findings are in line with Proposition 2, which states an open routing game with a shared station has a Nash Equilibrium in which all players visit shared station A first. Although, it has to be noted that there are differences in system conditions, as the proposition holds for systems with simultaneous arrivals while the simulation works with simultaneous as well as sequential arrivals.

The differences between these two systems could be an interesting subject for future research, where the similarities and differences in analytical propositions could be further researched. This may lead to a better practical application of the propositions made by Arlotto et al. (2019), for the two-station system and our own propositions for the three-station subset system.

Another example for future research would be a further elaboration on the idea of a subset open routing game. This could be by the addition of another subset, which would imply each station is shared by two classes, or the implementation of more stations while maintaining only one shared station. A further understanding of these situations could widen the practical application of these results, as many practical situations only slightly differ from each other. Therefore a more general statement about herding behaviour would be of great practical value.

Finally, as already stated by Arlotto et al. (2019), an analyses with varying utility for different customers could give a more realistic result. As customers may have a more complex opinion about waiting times and experience waiting in queues differently. For example, some may prefer waiting equally long at both stations over waiting for a longer time at one station and being served at the second station immediately. Even if the total waiting time is equal for both situations.

An improvement for future research would be the inclusion of a wider range of service rates, inter-arrival times and priority variance. Furthermore, if practically possible, the cut-off restriction used in the simulation could be dropped. It could also be interesting to include varying types of learning rules, instead of only letting individuals decide based on the historical waiting times. This could show how routing behaviour differs between different assumed learning rules.

References

A. Arlotto, A. E. Frazelle, and Y. Wei. Strategic open routing in service networks. *Management Science*, 65(2):735–750, 2019.

O. Baron, O. Berman, D. Krass, and J. Wang. Strategic idleness and dynamic scheduling in an open-shop service network: Case study and analysis. *Manufacturing & Service Operations Management*, 19(1):52–71, 2016.

S. Foss and N. Chernova. On the stability of a partially accessible multi-station queue with state-dependent routing. *Queueing Systems*, 29(1):55–73, 1998.

J. Kingman. Some inequalities for the queue gi/g/1. *Biometrika*, 49(3/4):315–324, 1962.

A. Müller and D. Stoyan. *Comparison methods for stochastic models and risks*, volume 389. Wiley New York, 2002.

A. K. Parlaktürk and S. Kumar. Self-interested routing in queueing networks. *Management Science*, 50(7):949–966, 2004.

J. M. Pinilla and F. B. Prinz. Lead-time reduction through flexible routing: application to shape deposition manufacturing. *International Journal Of Production Research*, 41 (13):2957–2973, 2003.

S. Veeraraghavan and L. Debo. Joining longer queues: Information externalities in queue choice. *Manufacturing & Service Operations Management*, 11(4):543–562, 2009.

S. K. Veeraraghavan and L. G. Debo. Herding in queues with waiting costs: Rationality and regret. *Manufacturing & Service Operations Management*, 13(3):329–346, 2011.

A Analytical proofs

Proof of Proposition 1. Suppose $\mu_A < \mu_C < \mu_B$ and $N \geq \frac{2\mu_A}{\mu_C} + 1$, then whenever a specific station (B or C) is idle, it will never have a queue again, since the service rate of station A is the lowest of all three. This also implies that whenever station A has started serving customers, it will not be idle until all customers are served. Furthermore, as the priorities of customers are drawn uniformly at random, the game is symmetric given the class of the customer. This means customer i can be considered, where i represents an arbitrary player index as long as the player remains in the same class.

Assume all individuals from class S_{ab} and S_{ac} start at station A, than customer i with priority j from class S_{ab} will have to wait for $j-1$ customers to be served before being served herself. When leaving station A, they will have immediate service at station B. Thus, customer i with priority j from class S_{ab} has total service time $Q^A_{ab}(j)$ given by

$$Q^A_{ab}(j) = \frac{j}{\mu_A} + \frac{1}{\mu_B}, j = 1, ..., N. \tag{2}$$

Let $T_{ab}(1, m, k)$ be the expected time in the system for customer i, from class S_{ab} who attends station A first, as well as m other customers from class S_{ab} and k customers from class S_{ac}. $T_{ab}(0, m, k)$ is the expected system time if customer i chooses to attend station B first, and $m + k$ customers attend A first. Due to the uniformly random priorities, the expected waiting time for customer i is

$$T_{ab}(1, \frac{N}{2} - 1, \frac{N}{2}) = \sum_{j=1}^{N} \frac{1}{N} Q^A_{ab}(j) = \sum_{j=1}^{N} \frac{1}{N}(\frac{j}{\mu_A} + \frac{1}{\mu_B}) = \frac{1}{\mu_B} + \frac{N+1}{2\mu_A}, \tag{3}$$

where the third step contains the constant occupancy of station A in $\frac{j}{\mu_A}$ and the lack off a queue forming at station B, as this station is faster than station B, in $\frac{1}{\mu_B}$. If the individual deviates from visiting route A first, they will be last in line at station A. Therefore, $T_{ab}(0, \frac{N}{2} - 1, \frac{N}{2})$ is deterministic and given by

$$T_{ab}(0, \frac{N}{2} - 1, \frac{N}{2}) = \frac{1}{\mu_B} + (\frac{N}{\mu_A} - \frac{1}{\mu_B}) = \frac{N}{\mu_A}, \tag{4}$$

where the $(\frac{N}{\mu_A} - \frac{1}{\mu_B})$ represents the time spent waiting for $N - 1$ customers at station A and being served herself at A, minus the time spent at B. As station A never gets idle once service has started until all customers are served, these two values can be deducted from each other. This implies due to the assumptions of $N \geq \frac{2\mu_A}{\mu_C} + 1$ and $\mu_A < \mu_C < \mu_B$ that

$$\frac{1}{\mu_B} + \frac{N+1}{2\mu_A} \leq \frac{1}{\mu_C} + \frac{N+1}{2\mu_A} \leq \frac{N}{\mu_A}. \tag{5}$$

Therefore, there is no incentive for customer i from class S_{ab} to deviate from taking route AB, as $T_{ab}(1, \frac{N}{2}-1, \frac{N}{2}) \leq T_{ab}(0, \frac{N}{2}-1, \frac{N}{2})$.

For a customer i in class S_{ac}, $T_{ac}(1, \frac{N}{2}, \frac{N}{2}-1)$ and $T_{ac}(0, \frac{N}{2}, \frac{N}{2}-1)$ can be found by a similar approach and are given by

$$T_{ac}(1, \frac{N}{2}, \frac{N}{2}-1) = \frac{1}{\mu_C} + \frac{N+1}{2\mu_A} \quad and \quad T_{ac}(0, \frac{N}{2}, \frac{N}{2}-1) = \frac{N}{\mu_A}. \tag{6}$$

Therefore, due to the assumptions of $N \geq \frac{2\mu_A}{\mu_C} + 1$ and $\mu_A < \mu_C < \mu_B$,

$$\frac{1}{\mu_C} + \frac{N+1}{2\mu_A} \leq \frac{N}{\mu_A}. \tag{7}$$

This means there is no incentive for customer i from class S_{ac} to change to route CA. This brings that we have a Nash Equilibrium at station A.

Proof of Proposition 2. Assume equal variance of the service rates for station B and C ($\sigma_B^2 = \sigma_C^2$), $\mu_A < \mu_C < \mu_B$ and that all customers from class S_{ab} and S_{ac} visit station A first. This makes that stations B and C are empty until the first departure towards the specific station. As all customers visit station A first and priorities between customers from class S_{ab} and S_{ac} are distributed alternately, station B behaves like a $GI/GI/1$ queueing system with arrival rate $\frac{\mu_A}{2}$ and service rate μ_B. The same holds for station C with arrival rate $\frac{\mu_A}{2}$ and service rate μ_C. Note that as $\mu_A < \mu_C < \mu_B$, both systems would be stable.

We let $F_{W^B_{0,ab}}$ be the distribution function for a random variable, independent of the arrival and service processes, which may alter the initial state of the queueing system. $W^B_{k,ab}$, $k \geq 1$, denotes the waiting time the k-th departure of class S_{ab} from station A experiences at station B. Here, $F_{W^B_{k,ab}}$ is the distribution function of $W^B_{k,ab}$. As with probability 1, $W^B_{0,ab} = 0$ as well as $W^B_{1,ab} = 0$, since station B starts empty, we can state $F_{W^B_{1,ab}}$ stochastically dominates $F_{W^B_{0,ab}}$. We denote this as

$$F_{W^B_{0,ab}} \leq_{st} F_{W^B_{1,ab}}. \tag{8}$$

To define the stationary waiting time distribution function for a $GI/GI/1$ queueing system, we use $F_{W^B_{\infty,ab}}$. As stated in Müller and Stoyan (2002) by Theorem 6.2.1, it holds that

$$F_{W^B_{k,ab}} \leq_{st} F_{W^B_{\infty,ab}} \quad \forall\, k = 1, 2, ... \tag{9}$$

The current strategy of the customer, with all other customers visiting station A first as well, gives her priority k at station A, where $k = 1, .., N$ each has probability $\frac{1}{N}$.

Therefore, the conditional expected time in the system, $\mathbb{E}[Q^A_{ab}|k]$, is given by

$$\mathbb{E}[Q^A_{ab}|k] = \frac{k}{\mu_A} + \mathbb{E}[W^B_{k,ab}] + \frac{1}{\mu_B}. \tag{10}$$

Which is constructed by the expected waiting and service time at station A, the expected waiting time at station B and the expected service time at station B. Equation (9) implies that $\mathbb{E}[W^B_{k,ab}] \leq \mathbb{E}[W^B_{\infty,ab}]$. Therefore, we can state

$$\mathbb{E}[Q^A_{ab}|k] \leq \frac{k}{\mu_A} + \mathbb{E}[W^B_{\infty,ab}] + \frac{1}{\mu_B} \leq \frac{k}{\mu_A} + \frac{\mu_A(\sigma_A^2 + \sigma_B^2)}{4(1 - \frac{\mu_A}{2\mu_B})} + \frac{1}{\mu_B}. \tag{11}$$

The final inequality follows from bounds for the steady-state expected waiting time in a queue for a $GI/GI/1$ queue as given by Kingman (1962), with $\frac{\mu_A}{2}$ as arrival rate for station B. This leads to expected system time

$$\mathbb{E}[Q^A_{ab}] \leq \frac{1}{N}\sum_{k=1}^{N}(\frac{k}{\mu_A} + \frac{\mu_A(\sigma_A^2 + \sigma_B^2)}{4(1 - \frac{\mu_A}{2\mu_B})} + \frac{1}{\mu_B}) = \frac{N+1}{2\mu_A} + \frac{\mu_A(\sigma_A^2 + \sigma_B^2)}{4(1 - \frac{\mu_A}{2\mu_B})} + \frac{1}{\mu_B}. \tag{12}$$

When a customer from class S_{ab} deviates, she will be last in line at station A. Therefore, her expected total system time, $\mathbb{E}[Q^B_{ab}]$, is at least $\frac{N}{\mu_A}$. Finally, as $N > 1 + \frac{2\mu_A}{\mu_C} + \frac{\mu_A^2(\sigma_A^2+\sigma_B^2)}{2-\frac{\mu_A}{\mu_B}}$ and $\mu_A < \mu_C < \mu_B$, it follows

$$\begin{aligned}
&\frac{2\mu_A}{\mu_C} + \frac{\mu_A^2(\sigma_A^2 + \sigma_B^2)}{2 - \frac{\mu_A}{\mu_B}} < N - 1 \\
\rightarrow &\frac{1}{\mu_C} + \frac{\mu_A(\sigma_A^2 + \sigma_B^2)}{2(2 - \frac{\mu_A}{\mu_B})} < \frac{1}{\mu_B} + \frac{\mu_A(\sigma_A^2 + \sigma_B^2)}{4(1 - \frac{\mu_A}{2\mu_B})} < \frac{N-1}{2\mu_A} \\
\rightarrow &\mathbb{E}[Q^A_{ab}] \leq \frac{N+1}{2\mu_A} + \frac{\mu_A(\sigma_A^2 + \sigma_B^2)}{4(1 - \frac{\mu_A}{2\mu_B})} + \frac{1}{\mu_B} < \frac{N}{\mu_A} \leq \mathbb{E}[Q^B_{ab}]
\end{aligned}$$

Therefore, a customers expected system time is shorter if she chooses route AB over BA and thus is there no incentive for her to deviate.

When considering a customer from class S_{ac} the expected system time for both routes can be found in a similar way and are given by

$$\mathbb{E}[Q^A_{ac}] \leq \frac{N+1}{2\mu_A} + \frac{\mu_A(\sigma_A^2 + \sigma_C^2)}{4(1 - \frac{\mu_A}{2\mu_C})} + \frac{1}{\mu_C} \quad \textit{and} \quad \mathbb{E}[Q^B_{ac}] \geq \frac{N}{\mu_A}. \tag{13}$$

With these expected total system times and the condition for N, it follows

$$\mathbb{E}[Q^A_{ac}] \leq \frac{N+1}{2\mu_A} + \frac{\mu_A(\sigma_A^2 + \sigma_C^2)}{4(1 - \frac{\mu_A}{2\mu_C})} + \frac{1}{\mu_C} \leq \frac{N}{\mu_A} \leq \mathbb{E}[Q^B_{ac}].$$

This implies there is no incentive for a customer in class S_{ac} to deviate from route AC to route CA when all other customers visit station A first. This makes that we can conclude it is a Nash Equilibrium for all customers in both classes to visit station A first.

B Past Results

Sample mean						Sample first quartile					
	ϕ						ϕ				
γ	0	0.25	0.5	0.75	1	γ	0	0.25	0.5	0.75	1
0.1	46.22	46.61	45.42	47.45	49.2	0.1	44	45	45	47	49
0.25	46.55	45.83	46.22	46.07	45.72	0.25	45	44	44.75	44	44
0.5	45.2	45.27	45	44.64	44.32	0.5	43	43	42	42	42
0.75	41.68	42.7	41.59	41.73	42.46	0.75	39.75	41	38.75	40	40.75
1	35.9	37.38	36.47	36.23	38.17	1	33.75	34.75	34	34	35

Table 7: Summary statistic for Number of AB Customers in the Final Round,with μ_A =0.75 by Arlotto et al. (2019)

Sample mean						Sample first quartile					
	ϕ						ϕ				
γ	0	0.25	0.5	0.75	1	γ	0	0.25	0.5	0.75	1
0.1	48.47	48.53	49.98	50	50	0.1	48	48	50	50	50
0.25	48.34	48.39	47.81	48.61	49.18	0.25	48	48	47	48	49
0.5	48.05	48.07	48.3	48.1	47.82	0.5	47	47	48	47.75	47
0.75	47.88	47.9	47.81	47.8	47.76	0.75	47	47	47	47	47
1	47.48	47.42	47.47	47.52	47.33	1	47	47	47	47	47

Table 8: Summary statistic for Number of AB Customers in the Final Round,with μ_A =0.5 by Arlotto et al. (2019)

C Code

Two-Station simulation code

```
import numpy as np
import pandas as pd

#import sys

import itertools
#from multiprocessing.dummy import Pool as ThreadPool

#from customQueue import Queue
#from iteration import Iteration

max_trails = 100
max_rounds = 250
```

```
n = 50
count=-1

gamma_params = np.array([0.001, 0.1, 0.25, 0.5, 0.75, 1])
#gamma_params = np.array([0.1, 0.25, 0.5, 0.75, 1])
#gamma_params = np.array([0.25, 0.5, 0.75, 1])
#gamma_params = np.array([10])

phi_params = np.array([0, 0.25, 0.5, 0.75, 1])
mu_a = np.array([0.1, 0.25, 0.5, 0.75])
#mu_a = np.array([0.5, 0.75])
skip=0

params = itertools.product(gamma_params, phi_params, mu_a)
final_results = pd.DataFrame(index=range(max_trails), columns=range(len(gamma_params)*len(phi_params)*len(mu_a)))
#iteration = Iteration()
#pool = ThreadPool(6)
#results = pool.starmap(iteration.iteration, zip(params, itertools.repeat(max_trails), itertools.repeat(max_rounds), itertools.repeat(n)))
#pool.close()
#pool.join()
#final_results = results
for param in params:
    skip +=1
    if skip<=5:
        continue
    count+=1
    individuals = pd.DataFrame(index=range(n),columns=["person id","arrivaltime","stations visited", "current route", "avg time AB", "avg time BA"])
    individuals['person id'] = range(n)
    for trail in range(max_trails):
        #print("trail "+str(trail))
        individuals['current route'] = np.random.choice(a=[True, False], size=(n,1))
        individuals['avg time AB'] = np.zeros(n)
        individuals['avg time BA'] = np.zeros(n)
        stationary=0
        prev_stat=0

        for nRound in range(max_rounds):
            print(str(param)+" - trail "+str(trail)+" - round "+str(nRound))
            if nRound>1:
                ab_time = np.copy(individuals['avg time AB'])
                ba_time = np.copy(individuals['avg time BA'])
                individuals['current route'] = ab_time <= ba_time  # if true, route AB is chosen
                individuals['current route'][ab_time == ba_time] = np.random.choice(a=[True, False], size=len(individuals['current route'][ab_time == ba_time]))

            incr_vec = np.arange(n)+1
            lower = incr_vec * param[0] - param[1]
            upper = incr_vec * param[0] + param[1]
            arrival_times = np.random.choice(a=upper-lower,size=(n,))+lower
            individuals['arrivaltime']=arrival_times

            #serv_timesA = np.random.exponential(scale=1/param[2],size=n)
            serv_timesA = np.random.exponential(scale=1/param[2],size=n)

            serv_timesB = np.random.exponential(scale=1, size=n)


            waitingtime_ab = np.zeros(n)
            waitingtime_ba = np.zeros(n)

            arrivals_station1 = np.copy(arrival_times)

            departure_stationA = np.zeros(n)
            departure_stationB = np.zeros(n)

            departure_stationAFic = np.zeros(n)
            departure_stationBFic = np.zeros(n)

            first_availabilityA = np.zeros(n)
            first_availabilityB = np.zeros(n)

            individuals['stations visited']=0
            stations_visitedFic=np.zeros(n)
```

```
        while sum(individuals['stations visited']) <100:

            if sum(arrivals_station1)==0:
                min_arrival=100000
            else:
                min_arrival = np.min(arrivals_station1[np.nonzero(arrivals_station1)])

            if sum(departure_stationA)==0:
                min_departureA=100000
            else:
                min_departureA = np.min(departure_stationA[np.nonzero(departure_stationA)])

            if sum(departure_stationB)==0:
                min_departureB=100000
            else:
                min_departureB = np.min(departure_stationB[np.nonzero(departure_stationB)])

            #fictitious play
            if sum(departure_stationAFic)==0:
                min_departureAFic=100000
            else:
                min_departureAFic = np.min(departure_stationAFic[np.nonzero(departure_stationAFic)])

            if sum(departure_stationBFic)==0:
                min_departureBFic=100000
            else:
                min_departureBFic = np.min(departure_stationBFic[np.nonzero(departure_stationBFic)])

            #choose next event

            if  min_arrival <= min_departureA and min_arrival <= min_departureB and min_arrival <= min_departureAFic and min_arrival <= min_departureBFic : # or np.isnan(min_departureB)):
                current_time = min_arrival
                id = np.where(arrivals_station1==min_arrival)[0][0]
                event_type=0 #first arrival
                arrivals_station1[id]=0

            elif min_departureA <= min_departureB and min_departureA <= min_departureAFic and min_departureA <= min_departureBFic: # or np.isnan(min_departureB):
                current_time = min_departureA
                id = np.where(departure_stationA==min_departureA)[0][0]
                event_type=1 #departure from A
                departure_stationA[id]= 0

            elif min_departureB <= min_departureAFic and min_departureB <= min_departureBFic:
                current_time = min_departureB
                id = np.where(departure_stationB==min_departureB)[0][0]
                event_type=2 #departure from B
                departure_stationB[id] = 0

            elif min_departureAFic <= min_departureBFic:
                current_time = min_departureAFic
                id = np.where(departure_stationAFic == min_departureAFic)[0][0]
                event_type=3 #departure from fictitious A
                departure_stationAFic[id]=0
            else:
                current_time = min_departureBFic
                id = np.where(departure_stationBFic == min_departureBFic)[0][0]
                event_type=4 #departure from fictitious B
                departure_stationBFic[id]=0

            curr_time_vec = np.full(n,current_time)

            #execute event
            if(event_type==0):      #first arrival event
                id_route = individuals.ix[id,'current route']

                if id_route:            #individuals visits A first => compute departure from A
                    departure_stationA[id] = np.maximum.reduce([np.maximum.reduce(departure_stationA),current_time]) + serv_timesA[id]
```

```
                        departure_stationBFic[id] = np.maximum.reduce([np.maximum.reduce(departure_stationB),current_time]) + serv_timesB[id]
                        first_availabilityA[:id] = np.maximum.reduce([curr_time_vec[:id],first_availabilityA[:id]]) + serv_timesA[id]
                        first_availabilityA[id+1:] = np.maximum.reduce([curr_time_vec[id+1:],first_availabilityA[id+1:]]) + serv_timesA[id]
                    else:       #individuals visits B first => compute departure from B
                        departure_stationB[id] = np.maximum.reduce([np.maximum.reduce(departure_stationB),current_time]) + serv_timesB[id]
                        departure_stationAFic[id] = np.maximum.reduce([np.maximum.reduce(departure_stationA),current_time]) + serv_timesA[id]
                        first_availabilityB[:id] = np.maximum.reduce([curr_time_vec[:id],first_availabilityB[:id]]) + serv_timesB[id]
                        first_availabilityB[id+1:] = np.maximum.reduce([curr_time_vec[id+1:],first_availabilityB[id+1:]]) + serv_timesB[id]

                elif(event_type==1):    #departure from station A
                    individuals.ix[id,'stations visited'] += 1

                    if individuals.ix[id,'stations visited']==2:
                        waitingtime_ba[id]=current_time
                    else:
                        departure_stationB[id] = np.maximum.reduce([np.maximum.reduce(departure_stationB),current_time]) + serv_timesB[id]
                        first_availabilityB[:id] = np.maximum.reduce([curr_time_vec[:id],first_availabilityB[:id]]) + serv_timesB[id]
                        first_availabilityB[id+1:] = np.maximum.reduce([curr_time_vec[id+1:],first_availabilityB[id+1:]]) + serv_timesB[id]

                elif(event_type==2):    #departure from station B
                    individuals.ix[id,'stations visited'] += 1

                    if individuals.ix[id,'stations visited']==2:
                        waitingtime_ab[id]=current_time
                    else:
                        departure_stationA[id] = np.maximum.reduce([np.maximum.reduce(departure_stationA),current_time]) + serv_timesA[id]
                        first_availabilityA[:id] = np.maximum.reduce([curr_time_vec[:id],first_availabilityA[:id]]) + serv_timesA[id]
                        first_availabilityA[id+1:] = np.maximum.reduce([curr_time_vec[id+1:],first_availabilityA[id+1:]]) + serv_timesA[id]

                elif(event_type==3): #departure from fictitious station A
                    stations_visitedFic[id] +=1

                    if stations_visitedFic[id]==2:
                        waitingtime_ba[id]=current_time
                    else:
                        departure_stationBFic[id] = first_availabilityB[id] + serv_timesB[id]

                elif(event_type==4): #departure from fictitious station B
                    stations_visitedFic[id] +=1

                    if stations_visitedFic[id]==2:
                        waitingtime_ab[id]=current_time
                    else:
                        new_departuretime = first_availabilityA[id] + serv_timesA[id]
                        departure_stationAFic[id] = new_departuretime


            waitingtime_ab = waitingtime_ab - individuals['arrivaltime']
            waitingtime_ba = waitingtime_ba - individuals['arrivaltime']
            individuals['avg time AB'] = ((nRound)*individuals['avg time AB']+waitingtime_ab)/(nRound+1)
            individuals['avg time BA'] = ((nRound)*individuals['avg time BA']+waitingtime_ba)/(nRound+1)
            if np.sum(individuals['current route'])==prev_stat or all(individuals['current route']==True) or all(individuals['current route']==False) :
                stationary +=1
            prev_stat = np.sum(individuals['current route'])
            if stationary >= 50:
                break
        final_results.ix[trail,count]=sum(individuals['current route'])
#final_results.to_csv(r'C:\Users\mees_\Documents\Thesis QM\data\results_total.csv')
```

Three-Station Simulation

```python
import numpy as np
import pandas as pd

#import sys

import itertools
#from multiprocessing.dummy import Pool as ThreadPool

#from customQueue import Queue
#from iteration import Iteration

max_trails = 100
max_rounds = 250
n_ab = 25
n_ac = 25
n= n_ab + n_ac
count=-1

gamma_params = np.array([0.001, 0.1, 0.25, 0.5, 0.75, 1])
#gamma_params = np.array([0.1, 0.25, 0.5, 0.75, 1])
phi_params = np.array([0, 0.25, 0.5, 0.75, 1])
mu_a = 1
mu_b_c_params = np.array([(0.25,0.1),(0.75,0.5),(1.25,0.75),(1.75,0.25),(1.25,1.1),(1.75,1.5)])
#mu_b_c_params = np.array([(0.25,0.1),(1.75,0.25)])


params = itertools.product(gamma_params, phi_params, mu_b_c_params)
final_results = pd.DataFrame(index=range(max_trails), columns=range(len(gamma_params)*len(phi_params)*len(mu_b_c_params)))
final_results_ab = pd.DataFrame(index=range(max_trails), columns=range(len(gamma_params)*len(phi_params)*len(mu_b_c_params)))
final_results_ac = pd.DataFrame(index=range(max_trails), columns=range(len(gamma_params)*len(phi_params)*len(mu_b_c_params)))

#iteration = Iteration()
#pool = ThreadPool(6)
#results = pool.starmap(iteration.iteration, zip(params, itertools.repeat(max_trails), itertools.repeat(max_rounds), itertools.repeat(n)))
#pool.close()
#pool.join()
#final_results = results

for param in params:
    count+=1
    individuals = pd.DataFrame(index=range(n),columns=["person id","arrivaltime","stations visited",
                                                        "current route", "avg time A first", "avg time A last", "subset"])
    individuals['person id'] = range(n)
    individuals.ix[:n_ab,'subset']=True #subset is true if subset ab is assigned
    individuals.ix[n_ab:,'subset']=False #subset is false if subset ac is assigned

    for trail in range(max_trails):
        #print("trail "+str(trail))
        individuals['current route'] = np.random.choice(a=[True, False], size=(n,1))
        individuals['avg time A first'] = np.zeros(n)
        individuals['avg time A last'] = np.zeros(n)
        stationary=0
        prev_stat=0

        for nRound in range(max_rounds):
            print(str(param)+" - trail "+str(trail)+" - round "+str(nRound))
            if nRound>1:
                afirst_time = np.copy(individuals['avg time A first'])
                alast_time = np.copy(individuals['avg time A last'])
                individuals['current route'] = afirst_time <= alast_time  # if true, station A is visited first
                individuals['current route'][afirst_time == alast_time] = np.random.choice(a=[True, False],size=len(individuals['current route'][afirst_time == alast_time]))

            incr_vec_ab = 2*np.arange(n_ab)+1
            incr_vec_ac = 2*np.arange(n_ac)+2
            incr_vec = np.concatenate((incr_vec_ab,incr_vec_ac))
            lower = incr_vec * param[0] - param[1]
            upper = incr_vec * param[0] + param[1]
            arrival_times = np.random.choice(a=upper-lower,size=(n,))+lower
            individuals['arrivaltime']=arrival_times
```

```
        serv_timesA = np.random.exponential(scale=1/mu_a, size=n)
        serv_timesB = np.random.exponential(scale=1/param[2][0], size=n_ab)
        serv_timesC = np.random.exponential(scale=1/param[2][0], size=n_ac)

        waitingtime_ab = np.zeros(n_ab)
        waitingtime_ba = np.zeros(n_ab)
        waitingtime_ac = np.zeros(n_ac)
        waitingtime_ca = np.zeros(n_ac)

        arrivals_station1 = np.copy(arrival_times)

        departure_stationA = np.zeros(n)
        departure_stationB = np.zeros(n_ab)
        departure_stationC = np.zeros(n_ac)

        departure_stationAFic = np.zeros(n)
        departure_stationBFic = np.zeros(n_ab)
        departure_stationCFic = np.zeros(n_ac)

        first_availabilityA = np.zeros(n)
        first_availabilityB = np.zeros(n_ab)
        first_availabilityC = np.zeros(n_ac)

        individuals['stations visited']=0
        stations_visitedFic=np.zeros(n)


        while sum(individuals['stations visited']) <100:

            if sum(arrivals_station1)==0:
                min_arrival=100000
            else:
                min_arrival = np.min(arrivals_station1[np.nonzero(arrivals_station1)])

            if sum(departure_stationA)==0:
                min_departureA=100000
            else:
                min_departureA = np.min(departure_stationA[np.nonzero(departure_stationA)])

            if sum(departure_stationB)==0:
                min_departureB=100000
            else:
                min_departureB = np.min(departure_stationB[np.nonzero(departure_stationB)])

            if sum(departure_stationC)==0:
                min_departureC=100000
            else:
                min_departureC = np.min(departure_stationC[np.nonzero(departure_stationC)])

            #fictitious play
            if sum(departure_stationAFic)==0:
                min_departureAFic=100000
            else:
                min_departureAFic = np.min(departure_stationAFic[np.nonzero(departure_stationAFic)
                    ])

            if sum(departure_stationBFic)==0:
                min_departureBFic=100000
            else:
                min_departureBFic = np.min(departure_stationBFic[np.nonzero(departure_stationBFic)
                    ])

            if sum(departure_stationCFic)==0:
                min_departureCFic=100000
            else:
                min_departureCFic = np.min(departure_stationCFic[np.nonzero(departure_stationCFic)
                    ])


            #choose next event
            if  min_arrival <= min_departureA and min_arrival <= min_departureB and min_arrival <=
                 min_departureC and min_arrival <= min_departureAFic and min_arrival <=
                 min_departureBFic and min_arrival <= min_departureCFic: # or np.isnan(
                 min_departureB)):
                current_time = min_arrival
```

```
            id = np.where(arrivals_station1==min_arrival)[0][0]
            event_type=0 #first arrival
            arrivals_station1[id]=0

        elif min_departureA <= min_departureB and min_departureA <= min_departureC and min_departureA <= min_departureAFic and min_departureA <= min_departureBFic and min_departureA <= min_departureCFic: # or np.isnan(min_departureB):
            current_time = min_departureA
            id = np.where(departure_stationA==min_departureA)[0][0]
            event_type=1 #departure from A
            departure_stationA[id]= 0

        elif min_departureB <= min_departureC and min_departureB <= min_departureAFic and min_departureB <= min_departureBFic and min_departureB <= min_departureCFic:
            current_time = min_departureB
            id = np.where(departure_stationB==min_departureB)[0][0]
            event_type=2 #departure from B
            departure_stationB[id] = 0

        elif min_departureC <= min_departureAFic and min_departureC <= min_departureBFic and min_departureC <= min_departureCFic:
            current_time = min_departureC
            id = np.where(departure_stationC == min_departureC)[0][0] +n_ab
            event_type=3 #departure from C
            departure_stationC[id-n_ab]=0

        elif min_departureAFic <= min_departureBFic and min_departureAFic <= min_departureCFic:
            current_time = min_departureAFic
            id = np.where(departure_stationAFic == min_departureAFic)[0][0]
            event_type=4 #departure from fictitious A
            departure_stationAFic[id]=0

        elif min_departureBFic <= min_departureCFic:
            current_time = min_departureBFic
            id = np.where(departure_stationBFic == min_departureBFic)[0][0]
            event_type=5 #departure from fictitious B
            departure_stationBFic[id]=0

        else:
            current_time = min_departureCFic
            id = np.where(departure_stationCFic == min_departureCFic)[0][0] +n_ab
            event_type=6 #departure from fictitious C
            departure_stationCFic[id-n_ab]=0

        curr_time_vec = np.full(n,current_time)

        #execute event
        if(event_type==0):      #first arrival event
            id_route = individuals.ix[id,'current route'] # if true, visit A first
            id_subset = individuals.ix[id,'subset'] # if true, subset is AB

            if id_route and id_subset:            #individuals visits A first => compute departure from A and fictitious B
                departure_stationA[id] = np.maximum.reduce([np.maximum.reduce(departure_stationA),current_time]) + serv_timesA[id]
                departure_stationBFic[id] = np.maximum.reduce([np.maximum.reduce(departure_stationB),current_time]) + serv_timesB[id]
                first_availabilityA[:id] = np.maximum.reduce([curr_time_vec[:id], first_availabilityA[:id]]) + serv_timesA[id]
                first_availabilityA[id+1:] = np.maximum.reduce([curr_time_vec[id+1:], first_availabilityA[id+1:]]) + serv_timesA[id]

            elif not id_route and id_subset:       #individuals visits B first => compute departure from B and fictitious A
                departure_stationB[id] = np.maximum.reduce([np.maximum.reduce(departure_stationB),current_time]) + serv_timesB[id]
                departure_stationAFic[id] = np.maximum.reduce([np.maximum.reduce(departure_stationA),current_time]) + serv_timesA[id]
                first_availabilityB[:id] = np.maximum.reduce([curr_time_vec[:id], first_availabilityB[:id]]) + serv_timesB[id]
                first_availabilityB[id+1:] = np.maximum.reduce([curr_time_vec[id+1+n_ab:], first_availabilityB[id+1:]]) + serv_timesB[id]

            elif id_route and not id_subset:        #individuals visits A first => compute departure from A and fictitious C
                departure_stationA[id] = np.maximum.reduce([np.maximum.reduce(
```

```
                    departure_stationA),current_time]) + serv_timesA[id]
                    departure_stationCFic[id-n_ab] = np.maximum.reduce([np.maximum.reduce(departure_stationC),current_time]) + serv_timesC[id-n_ab]
                    first_availabilityA[:id] = np.maximum.reduce([curr_time_vec[:id],first_availabilityA[:id]]) + serv_timesA[id]
                    first_availabilityA[id+1:] = np.maximum.reduce([curr_time_vec[id+1:],first_availabilityA[id+1:]]) + serv_timesA[id]

                else:   #individuals visits C first => compute departure from C and fictitious A
                    departure_stationC[id-n_ab] = np.maximum.reduce([np.maximum.reduce(departure_stationC),current_time]) + serv_timesC[id-n_ab]
                    departure_stationAFic[id] = np.maximum.reduce([np.maximum.reduce(departure_stationA),current_time]) + serv_timesA[id]
                    first_availabilityC[:id-n_ab] = np.maximum.reduce([curr_time_vec[:id-n_ab],first_availabilityC[:id-n_ab]]) + serv_timesC[id-n_ab]
                    first_availabilityC[id-n_ab+1:] = np.maximum.reduce([curr_time_vec[id+1:],first_availabilityC[id-n_ab+1:]]) + serv_timesC[id-n_ab]

            elif(event_type==1):    #departure from station A
                individuals.ix[id,'stations visited'] += 1
                id_subset = individuals.ix[id,'subset'] # if true, subset is AB

                if individuals.ix[id,'stations visited']==2:
                    if id_subset:
                        waitingtime_ba[id]=current_time
                    else:
                        waitingtime_ca[id-n_ab] =current_time

                else:
                    if id_subset:
                        departure_stationB[id] = np.maximum.reduce([np.maximum.reduce(departure_stationB),current_time]) + serv_timesB[id]
                        first_availabilityB[:id] = np.maximum.reduce([curr_time_vec[:id],first_availabilityB[:id]]) + serv_timesB[id]
                        first_availabilityB[id+1:] = np.maximum.reduce([curr_time_vec[id+1+n_ab:],first_availabilityB[id+1:]]) + serv_timesB[id]
                    else:
                        departure_stationC[id-n_ab] = np.maximum.reduce([np.maximum.reduce(departure_stationC),current_time]) + serv_timesC[id-n_ab]
                        first_availabilityC[:id-n_ab] = np.maximum.reduce([curr_time_vec[:id-n_ab],first_availabilityC[:id-n_ab]]) + serv_timesC[id-n_ab]
                        first_availabilityC[id-n_ab+1:] = np.maximum.reduce([curr_time_vec[id+1:],first_availabilityC[id-n_ab+1:]]) + serv_timesC[id-n_ab]

            elif(event_type==2):    #departure from station B
                individuals.ix[id,'stations visited'] += 1

                if individuals.ix[id,'stations visited']==2:
                    waitingtime_ab[id]=current_time
                else:
                    departure_stationA[id] = np.maximum.reduce([np.maximum.reduce(departure_stationA),current_time]) + serv_timesA[id]
                    first_availabilityA[:id] = np.maximum.reduce([curr_time_vec[:id],first_availabilityA[:id]]) + serv_timesA[id]
                    first_availabilityA[id+1:] = np.maximum.reduce([curr_time_vec[id+1:],first_availabilityA[id+1:]]) + serv_timesA[id]

            elif(event_type==3):    #departure from station C
                individuals.ix[id,'stations visited'] += 1

                if individuals.ix[id,'stations visited']==2:
                    waitingtime_ac[id-n_ab]=current_time
                else:
                    departure_stationA[id] = np.maximum.reduce([np.maximum.reduce(departure_stationA),current_time]) + serv_timesA[id]
                    first_availabilityA[:id] = np.maximum.reduce([curr_time_vec[:id],first_availabilityA[:id]]) + serv_timesA[id]
                    first_availabilityA[id+1:] = np.maximum.reduce([curr_time_vec[id+1:],first_availabilityA[id+1:]]) + serv_timesA[id]

            elif(event_type==4): #departure from fictitious station A
                stations_visitedFic[id] +=1
                id_subset = individuals.ix[id,'subset'] # if true, subset is AB

                if stations_visitedFic[id]==2:
                    if id_subset:
```

```
                            waitingtime_ba[id]=current_time
                        else:
                            waitingtime_ca[id-n_ab]=current_time

                    else:
                        if id_subset:
                            departure_stationBFic[id] = first_availabilityB[id] + serv_timesB[id]
                        else:
                            departure_stationCFic[id-n_ab] = first_availabilityC[id-n_ab] +serv_timesC[id-n_ab]

                elif(event_type==5): #departure from fictitious station B
                    stations_visitedFic[id] +=1

                    if stations_visitedFic[id]==2:
                        waitingtime_ab[id]=current_time
                    else:
                        departure_stationAFic[id] = first_availabilityA[id] + serv_timesA[id]

                elif(event_type==6): #departure from ficitious station C
                    stations_visitedFic[id] +=1

                    if stations_visitedFic[id]==2:
                        waitingtime_ac[id-n_ab]=current_time
                    else:
                        departure_stationAFic[id] = first_availabilityA[id] + serv_timesA[id]



            waitingtime_ab = waitingtime_ab - individuals.ix[:n_ab-1,'arrivaltime']
            waitingtime_ba = waitingtime_ba - individuals.ix[:n_ab-1,'arrivaltime']
            waitingtime_ac = waitingtime_ac - individuals.ix[n_ab:,'arrivaltime']
            waitingtime_ca - waitingtime_ca - individuals.ix[n_ab:,'arrivaltime']

            individuals['avg time A first'] = ((nRound)*individuals['avg time A first']+np.concatenate((waitingtime_ab,waitingtime_ac)))/(nRound+1)
            individuals['avg time A last']  = ((nRound)*individuals['avg time A last'] +np.concatenate((waitingtime_ba,waitingtime_ca)))/(nRound+1)

            if np.sum(individuals['current route'])==prev_stat:
                stationary +=1
            prev_stat = np.sum(individuals['current route'])
            if all(individuals['current route']==True) or all(individuals['current route']==False) or stationary >= 50:
                break
        final_results.ix[trail,count]=sum(individuals['current route'])
        final_results_ab.ix[trail,count]=sum(individuals.ix[:n_ab-1,'current route'])
        final_results_ac.ix[trail,count]=sum(individuals.ix[n_ab:,'current route'])

#final_results.to_csv(r'C:\Users\mees_\Documents\Thesis QM\data\results_total.csv')
```