Simulating a Real-World Taxi Cab Company using a Multi Agent-Based Model

Matz Johansson Bergström
Department of Computer Science
Göteborgs Universitet
Email:matz.johansson@gmail.com or matz.johansson@chalmers.se

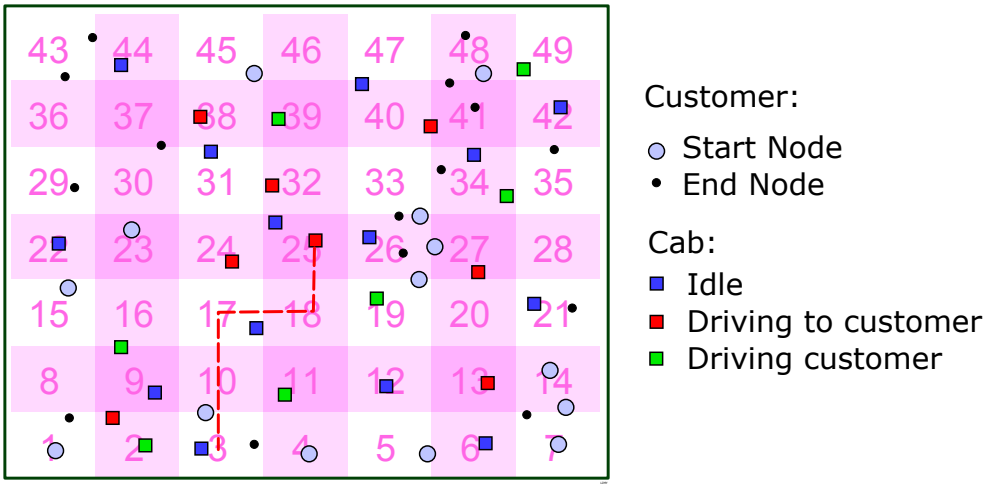

Figure 1: *A screenshot of the Taxi Simulator. The taxi cabs are indicated by color depending on their status. The zones are colored in a pattern so they are easier to see.*

Abstract

In this article we propose a novel multi-agent approach to simulate the behavior of a real-world taxi cab company. The focus of this article is to minimize the number of bailed customers by examining other rules and compare their performance to the existing system.

With our proposed rules we reduce the number of bailed customers, on average, by 40% and as a result of this, we reduce the waiting time also on average by 40%.

Keywords: Multi-agent, simulation, taxi system

1 Introduction

OBS: If not otherwise stated, all assumptions and data were gathered from a current employee of Taxi Göteborg.

Taxi systems are governed by a dispatch system, communicating and assigning customers to cabs.

The cab company we simulate is Taxi Göteborg. Their dispatch system works by following a set of rules to assign taxi cabs to customers based on a zone partitioning scheme of the city of Gothenburg (see Figure 2). Cabs are tracked using GPS and as soon as a cab enters a zone it is queued in that zone.

Figure 2: *The Zone partitioning of Gothenburg, courtesy of Taxi Göteborg. Note that the area of individual zones is changing with the distance to the center of the city.*

We believe that the most important aspect of the dispatch system is the presence of bailed customers. If a customer is assigned a cab, it is not obligated to drive with that cab. If the customer find another cab, it is very common that it will drive off with the other cab.

As seen, using real data, this probability tend to increase over time.

Most of our approximations are built using first-hand information from a current employee at Taxi Göteborg. On average, 10% of the customers during one session will bail on a cab.

Although the exact details of how the dispatcher system works is not readily available to the public, we managed to get enough information and estimates to build a simulation producing plausible outputs.

The simulation model is used to test different dispatch rules and to evaluate their performance with respect to minimizing service objectives (see next section).

The objective to reduce waiting time in customers is even more important today because of the increase of competing cab companies. In recent years, several taxi companies has emerged, such as *Taxi Kedjan*, *Taxi Kurir*, *Mini Taxi*, *Easy Cab* (flygtaxi) [pri]. The additional competition makes it more important that the dispatch system is efficient.

1.1 Background

Taxi Göteborg use a set of internal rules to dispatch cabs to their customers. Since the early 2000, the prevalence of mobile phones has increased. As a result, the calls to the taxi cab dispatch are mostly made outdoors. Because of this, the customers are visible to other cabs which greatly increases the possibility of a customer fetching another cab, instead of waiting for the designated one.

Taxi Göteborg prior to circa 2000 did not use GPS. Because of this, the chauffeurs had to type in the zone they were in via a computer each time they entered a zone. This system made it possible for the chauffeurs to cheat and enter a more popular zone, giving them a better position and to "play the odds". Today, this is impossible due to the introduction of the GPS. The chauffeurs are left in the hands of the dispatch system.

We believe that due to this shift of power from taxi company to customers, Taxi Göteborg needs a more efficient dispatch system, reducing the customer waiting time.

We also need to take into account that the number of cabs from rivaling companies also has increased in the last couple of years. The perception may be that Taxi Göteborg give their employees a fixed salary, which is false, making the customer less likely to notice if the company it calls to is the same as it leaves with.

This is at the core of the simulation, showing that the dispatch system is not well-suited for the fast moving IT solutions of tomorrow.

2 Related work

Much research on dispatch modeling and simulation are focused on frameworks and generic dispatch policies. Many papers are covering advanced techniques such as [Aamena Alshamsi and Rahwan 2009]. In [Aamena Alshamsi and Rahwan 2009] they propose a self-organization technique to change the shape of the dispatch areas as the density of traffic changes. This is however outside of the scope of this article.

In [Krishnan 2008] they simulate a dispatch system written in Python. The simulation aims to reduce the waiting time of customers using *vehicle anticipation*. The anticipation method is important because this will place the cab where the customer is likely to be, before the customer shows up. However their paper does not simulate bailing customers, see page 14 in [Krishnan 2008], which is what we want.

However, in our research we will not take customer anticipation into consideration.

Another detail that [Krishnan 2008] does is to use identical cities when comparing performance of the simulation. We adopted this idea and in our simulation we create a list of customers with their positions and run the simulation with that same list for all consecutive tests (not when we change hotspot).

As far as we know, no paper has created a simulation using hotspots and *bailing customers*.

In the next section we will briefly cover the way we created the city on which we simulate on.

3 The City

In our simulation we create a city as a lattice, or a regular grid of nodes. To break up symmetries we have removed edges, according to the following simple construction rules:

1. The perimeter of the city has edges around.
2. For each node we place at least two adjacent edges.

It is easy to see that these two rules guarantee that the city will never create islands of nodes that cannot be reached.

Because the grid is regular, we can effectively store the distances as a sparse matrix. In our simulation the number of nodes is 151, each edge is 50 meters (7.5 km across).

To approximate the city zone partitioning (see Figure 2), we use a grid to divide the city, as in Figure 3. We believe this approximation is relatively accurate because the density of customers will be the same for the smaller zones (near city core) as the large zones (in the outer limits).

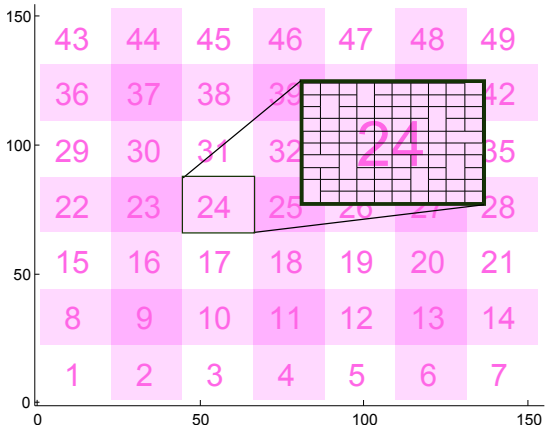

Figure 3: *Visualization of the city using 22801 nodes and partitioned into 49 equally sized zones. The zoomed in zone shows the individual edges. Please note that some edges are intentionally missing.*

We do not believe that the size of the city is important, as long as the cab speed and customer waiting times are comparable to our real-world data.

The next section will show all the approximations and omissions and some arguments for them.

4 Approximations

The following are approximations used in the project to make it computationally feasible. We believe these extra details do not add any extra information that affect the outcome of the simulation in any significant way.

1. Traffic lights are ignored
2. Uniform distribution of customers' start cites
3. Cars can drive through each other
4. Unlimited gas
5. All cars drive at the same speeds (35km/h)
6. No one-way streets
7. Each cab knows the shortest distance between any points
8. A customer will be ready for a cab when it calls

The first item is more of an optimization. If we agree that traffic lights are more of less random and since they affect all the cabs in the simulation there should be no difference if we remove them from the simulation as they will be cancelled out after many runs of the simulation anyway.

The start coordinates are uniformly randomly distributed. For some tests however, where we wish to model high flow of customers we use so called *hotspots*. In these places the probability that a customer is driving from and/or to is increased, which we will see next.

4.1 Simulating customer behavior

According to our statistics, the travel distances are *gamma distributed*. Practically, this means that a distance is generated from a gamma distributed random function, see Figure 4. The histogram was created using over 350 samples.

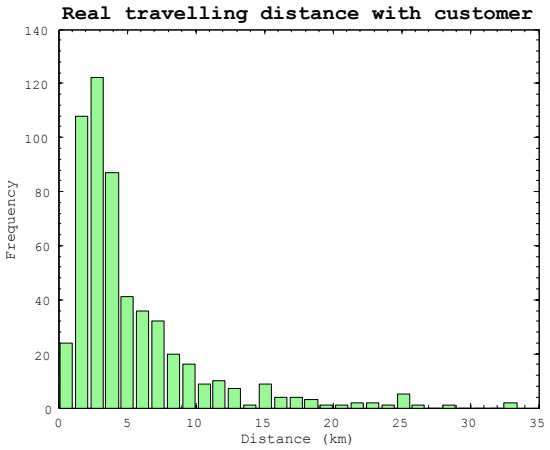

Figure 4: *Histogram of distances driven with customers. Note the characteristic tail to the right as far distances are less likely to occur.*

In our simulation we base our customers' choice of destination on a *gamma distribution*. We first pick a gamma distributed random distance, and collect the nodes at this distance. We will also simulate the way a popular starting and target node (we call this a "hotspot") affect the results. In our simulation we have chosen a hotspot at the actual center of the city, where the distribution of shopping malls are high and people are likely to travel to and from.

We can control the probability of customers spawning from the hotspot by setting a variable we call "source" and the target is controlled by the "sink". In Figure 5 we can see the frequency of the target nodes given different source and sink probabilities.

As the probability of the hotspot changes, we get different patterns of frequency where the customers end up. This knowledge might be interesting in customer anticipation.

Especially notice the diamond shape created when source = 1. The reason we do not get a round shape is because of how we measure distance on a grid. The distance function on a grid is popular called "taxi distance" or "taxi norm". In our simulation we cannot compute distance using the taxi norm, because some edges are removed.[1]

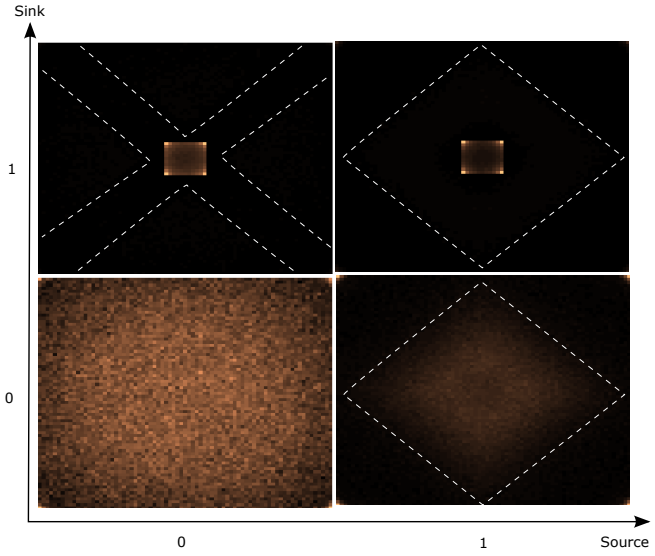

Figure 5: *Four renders of the frequencies of the nodes of the city where customers wish to travel to (sink), using the hotspot idea. The pixels might be difficult to discern from the background so we added some dashed lines.*

As said before, a cab driver will, on average, get 10% bailed customers. This information is also based on real data. As we will see later, our simulation will also produce about 10% bailed customers for the Taxi Göteborg rule.

4.2 Simulating chauffeur behavior

In order to simulate the behavior of the chauffeurs accurately, we introduce a set of rules

Spreading behavior: If a cab is in a zone with more than 5 cabs ahead of him and another zone will give the cab a better placement, the cab will drive to that zone after 10 minutes. This rule is a modification of a real-world rule where you might wait 30 minutes.

Chauffeurs are able to see which position it is in the current zone and also via a computer system which gives the number of cabs in other zones.

Deny customer: If a customer is too far away, the cab will deny the customer.

Bail punishment In the event that a customer bails on a cab, the chauffeur must wait. The waiting time depends on whether the cus-

[1] In earlier versions of the simulation we also planned on adding one-way streets, which affects the distance patterns.

tomer has a mobile phone or not. based on statistics, a customer is accessible via mobile phone in 70% of the time. The other 30% the chauffeur must wait 10 minutes as per Taxi Göteborg rules.

5 The code and problems encountered

The simulation was written in Matlab and works by stepping in time. For our simulation the granularity is 1 second.

The management of cabs, customers and queues is controlled using three matrices we call **C** (customer), **T** (taxi), **Q** (queue). T and C contains a time entry when they are supposed to make the next decision. For a cab, a decision is made when it arrives to the customer, to see if the customer jumped in another cab or not. The customer will also make decisions, such as call the dispatch if no cab is present at the moment.

The customer matrix **C** is reused when a customer has been taken care of, therefore it is important to place the information about a customer within **T** and not **C**. If a customer i is assigned a cab C_i and instead bails with cab C_j. Additionally, C_i might be so far away that C_j drives customer i to its destination before C_i arrives to the customer. In this case the customer entry i will be erased, or written over with the next customer (modulo size of **C**).

Our first approach was to create a city that was truly random, to be able to simulate specific scenarios using a completely new city for each simulation. Because we did not add direction of edges (to simulate one-way streets) we discarded both ideas due to time constraint. We are not convinced that these details would make any difference.

6 Experimental Results

The results are presented using graphs of the performance. We use the same ideas as in [Krishnan 2008], testing waiting time for customers, and idle driving (without customer). In our tests we also use the quantity of missed customers. The first set of tests was conducted using identical spawning of the customers. This makes the results easier to interpret, because the graphs are more stable.

Additionally, for the GBG (Taxi Göteborg) rule, we also noticed that the average number of missed customers is around 11%, comparable to the real-world data.

In all our tests we add a steady flow of customers. We use one customer per 43 seconds, which seem to work well for our purposes. We have no data on how customers call the dispatch. The customers may contact Taxi Göteborg via telephone or their homepage in bursts, instead of them being uniformly distributed over time, as we assume. Maybe the customer spawning could be successfully modeled using a Poisson process. We simply do not know the way the customers are spawned, but we believe our approximation suffices for this simple simulation.

In the following figures we ran the simulation with a variable number of cabs using the three rules:

1. **GBG rule** (Taxi Göteborg),
2. **Rule 1** (closest cab)
3. **Rule 2** (cab is given priority in queue after miss)

We wish to minimize the customer waiting time, but at the same time we also wish to minimize the number of cabs in the system.

In Figure 6 we can see the idle driving time, when the cabs drives around and looks for customers. The distance is linearly scaled due to the way the simulation calculated the stepping distance. The comparison between the distance will still be valid.

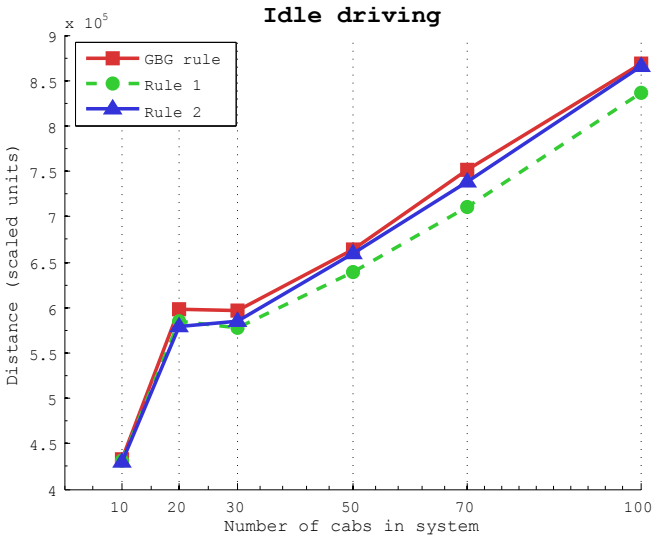

Figure 6: *The idle driving increase as the number of cabs increase in the system.*

By increasing the number of cabs in the system we would expect the total waiting time for all the customers to decrease as can be seen in Figure 7. Note the logarithm in the y axis.

Figure 7: *This simulation show that the priority rule (Rule 2) and the GBG rule outputs the same performance.*

The most important difference between the rules are the total number of missed customers, which we wish to minimize. As we increase the number of cabs, the number of missed customers will increase substantially using GBG rules and Rule 2. It is interesting to note that the idle time for all methods (as can be seen in Figure 6) are nearly the same, but the number of missed customers (hence punishment) differs greatly between GBG rule and rule 1. So, the idle time is reduced greatly for both Rule 1 and GBG rule, but because the cab spawning rate is as low as it is, the cabs will drive and look for a customer during the time it would get a punishment.

Based on the above figures, we choose to use 50 cabs for all subsequent simulations.

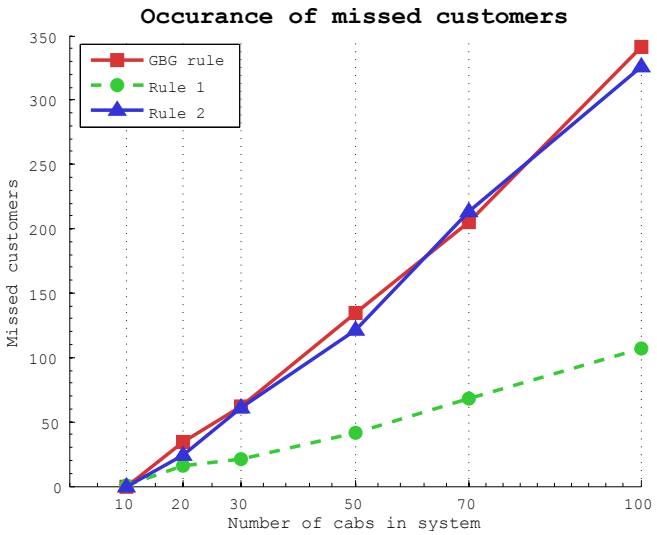

Figure 8: *We clearly see the difference in performance between our proposed Rule 1 and the other rules.*

In Figure 9, we see that the customer waiting time, using the GBG rule, is also gamma distributed. We believe this result is an example of that the simulation and our assumptions are relatively sound and reliable.

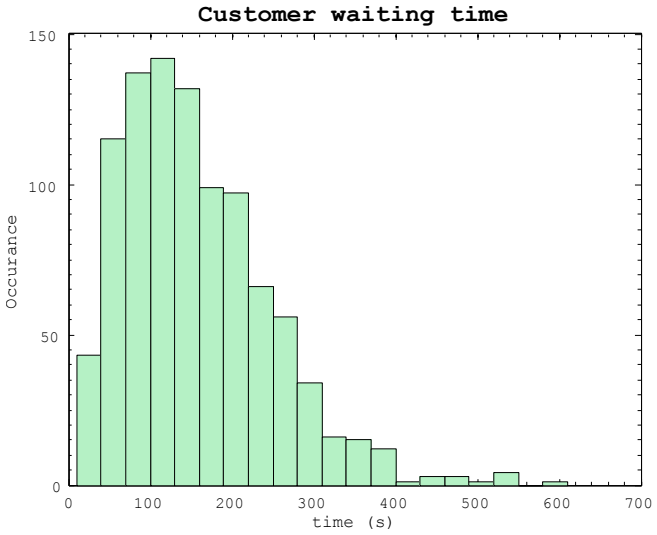

Figure 9: *Histogram of the Waiting time.*

The next simulations are focusing on increasing hotspot probabilities. In Figure 10, we can see how the changing of the hotspot probability affect the outcome of the simulations using the different rules.

Rule 1 performs very well for all cases.At one point source=1 sink=0 all the methods coincide. We can clearly see a pattern as we increase the hotspot sink value. As the source probability is increased, using GBG rule, the number of customers decrease. Rule 1 performs very well for all cases but when source and sink is 1.

For the next two figures, Figure 11 and Figure 12, the performances are bascially equal. As expected, Rule 2 performs poorly, compared to Rule 1, as the figures show.

The conclusion is that the Rule 1 reduces the time customers need to wait and also reduces the number of missed customers on average

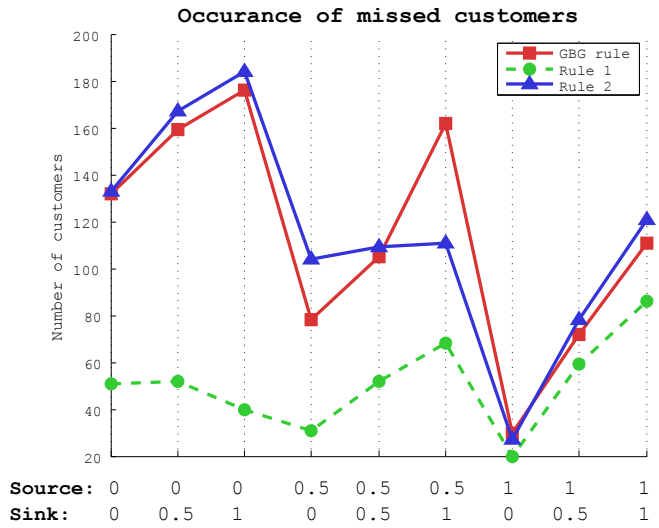

Figure 10: *We can see that the number of missed customers are reduced greatly if we use the shortest distance (Rule 1).*

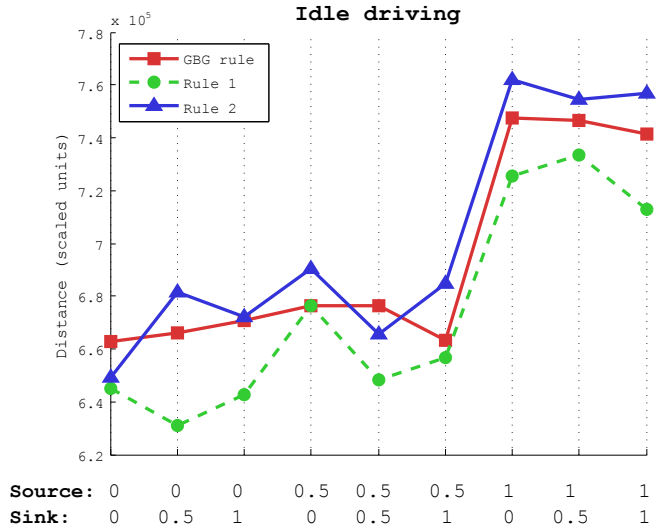

Figure 11: *The distance (scaled) of idle driving is very similar to the figure of waiting times. We see that for source=1 we get an increase of the total amount of idle driving.*

by 40%.

7 Discussion and Conclusion

In this article we proposed a novel simulation algorithm based on a multi-agent system that simulate Taxi Gothenburgs dispatch system. We showed that using our rules will reduce both customer waiting time. By also adding a hotspot, we showed that in a crowded environment the current rules handles the increase of flow in a poor manner, and that the proposed method would decrease both time and missed customers by an average of 40%. The rules we suggested show that in a hotspot situation they are bascially the same. Additionally, we assume that the customers are added into the system at regular intervals, but it is probably more realistic that the customers are generated at random and they are also dependent.

The idle driving, Figure 6, should be the same, because the customers are all the same in all those simulations. The distance is computed using scaled distance units computed using the elapsed

Figure 12: *This graph only confirms that the frequency of missed customers and the customer waiting time is linked.*

time to drive the customer. According to the figure, the distance difference is 5000 (m), for each cab this amounts to a discrepancy of 100 m for 12 hours of simulation. We believe this is due to a numerical rounding bug in the simulation.

8 Future Work

We believe that our simulation would benefit from additional detailed statistical data, such as real customer flow and customer movement while waiting on a cab.

In this report we only compare one cab company, but we believe that statistical data and dispatch rules from competing companies needs to be added to the simulation to increase reliability and accuracy of the results.

Acknowledgements

We would like to thank [anonymous] for the data provided and general help with the way Taxi Göteborg dispatch cabs, details on queueing and the statistical intuition needed.

We would also like to thank our supervisor Claes Andersson for feedback and the “hotspot” idea.

References

AAMENA ALSHAMSI, S. A., AND RAHWAN, I. 2009. Multi-agent self-organization for a taxi dispatch system. Tech. rep., International Foundation for autonomous agents and multiagent systems.

KRISHNAN, S. 2008. Simulation on service vehicle dispatching. Tech. rep., Department of mechanical and Industrial Engineering, University of Toronto.

Taxipriser i goteborg. http://taxipriser.se/bolag-i-goteborg.html. Accessed 21 Dec 2012.