Taxi Contract Net with Historical Data

Evert Etienne & Olivier Kamers

June 7, 2017

Abstract

New York City has more than 13.500 taxis who individually travel more than 110.000km every year to help over 600.000 customers every day! [BY14]
The current shift in Artificial Intelligence and Big Data learns us that most real life problems can be improved by analyzing their generated data. This leads us to investigate improvements to the taxi pickup and delivery problem (PDP) in Manhattan by using historical data analysis among other improvements.

1 Introduction

Contract Net (CNET) Protocols, as proposed by Smith in 1980 [Smi80], have been used to decentralize control in a Multi-Agent system. But this basic CNET does not specify which action(s) agents should undertake when they currently do not have an active contract.

In this research we aim to deduct more efficient idle actions based on the abundance of historical data from New York City (NYC) [NYC] regarding taxi trips from previous years. This information will be used by the taxi agents to predict where new customers will request a pickup in the near future.
Along with the distribution of idle taxis in Manhattan, busy taxis will also be able to communicate with other taxi agents in range to exchange CNET contracts to distribute the work.

More specifically we aim to answer the following questions with the given hypotheses:

1. *Is a discrete field a scalable way to represent historical data?*
   We expect this to be true due to the fact that a discrete field is a compact representation for a large amount of information. The dimensions and discretization of the field can be chosen as a trade-off between precision and required storage and needed computation power.

2. *How does the idle travel distance influence the change in pickup time?*
   We expect to introduce a more efficient idle taxi distribution by using the historical data. The closer the taxis are to the future pickups, the lower the pickup time will be. If this approach is proven efficient, we expect a longer idle travel distance will approach the most efficient distribution possible which will reduce the pickup times. This distance can be a trade-off between usage costs and pickup time reduction.

3. *What is, if it exists, the optimal idle travel distance?*
   We expect that the equilibrium between field attraction and taxi repulsion will be the most efficient distribution our system can achieve to reduce pickup times. This implies no hard limit on the idle travel distance, only a deducted limit will rise from the equilibrium.

4. *Will the average waiting time for the customers decrease?*
   We expect that the average waiting time will decrease because the taxis will move closer to possible pickup locations of future customers.

2 Theory

This section describes the theory that is relevant to the implementation proposed in this paper. Section 2.1 describes the theory regarding the Contract Net Protocol. Section 2.2 describes how fields are used in multi-agent systems.

2.1 Contract Net Protocol

A Contract Net, proposed by Smith in 1980 [Smi80], is a high-level protocol used to specify problem-solving communication and control without centralized coordination. It is designed to facilitate the distributed control of executing a cooperative task, like for example a PDP. A key characteristic of the protocol is the use of negotiation between agents.
Being a high-level protocol, CNET assigns meanings to the information being passed around in the network. More precisely, the messages sent between agents have well-defined semantics. It specifies *what* the agents should say to each other instead of *how* they should communicate. This latter decision is left open for the system designer to decide by choosing or implementing a low-level communication protocol. CNET only assumes the architecture to be a network of loosely coupled, asynchronous nodes. In this context *loosely coupled* means that the agents spend more time executing their own computations than communicating with each other.

Agents and contracts Agents in a CNET can take on one of the two defined roles: *manager* or *contractor*. A contractor is responsible for executing a task. The responsibility of a manager is to make sure a task gets executed by a contractor and to process the results. CNET does not rely on the fact that agents take on exactly one role. This means that agents can switch roles dynamically or even take on both roles at the same time to handle different contracts.
Establishing a contract between agents requires a two-way transfer of information. Managers announce new contract opportunities to the contractors. Contractors evaluate those announcements and decide whether or not they want to place a bid on the announced contract and calculate an appropriate bid. All incoming bids are evaluated by the manager who then offers the contract to the agent with the best bid. Contractors may choose to divide the task contained in the contract into smaller subtasks. Those subtasks can then be assigned to other contractors by repeating the negotiation process where this contractor now acts as a manager.
The decentralized communication model is obtained here by allowing every agent to communicate with every other agent. The communication does not need to pass through one or more central nodes or specialized agents.

Message types As mentioned above, CNET specifies several message types each with their own well-defined semantics.

The first message type is the **task announcement** message. This message is sent by an agent to notify other agents of a new task that is available. The agent that sent the message will then act as the manager of that task. The sending agent can decide whether the message should be broadcast to all other agents or to only a subset of them (called *focused addressing*), reducing the communication overhead in the system. The contents of a task announcement message must consist of the following four elements.
The message must contain the *eligibility specification*, indicating which criteria a contractor must meet in order to submit a bid on the task. Providing this information in the task announcement message greatly reduces the number of bid messages sent in the network as agents that are unable to fulfill the task will not send a bid.
The second element that must be present in the task announcement message is the *task abstraction*. This element briefly describes the task and allows the receiving agent to rank the task amongst other announced tasks it has received.
Thirdly, the message contains a *bid specification*. This is the description of the format

in which the bid is expected.
Finally, an *expiration time* is provided as deadline for receiving bids. Any bids received by the manager after this deadline may be ignored.

The second message type is the **bid** message. After receiving one or more task announcement messages an agent can select one or more of those tasks to bid on. It calculates a bid and sends a bid message to the manager of that task. The bid message contains a *node abstraction* summary of the agent that sent the bid. Bids are queued and ordered by the manager. It is allowed to award the contract to the best bidder even before the deadline has expired. If no satisfactory bids or even no bids at all are received by the manager then it can send another task announcement for the same task.

The third type of message is the **announced award** message. It is sent by the manager to the best bidder and contains a *task specification* which allows the contractor to execute the task.

Finally, some additional message types are defined. The **information** message which is used for general communication between the contractor and manager and the **report** message which is used by the contractor to notify the manager of partial or full success of the task execution. This last one contains a *result description* which is used for reporting the results of the execution. The final defined message is the **termination** message which can be used by the manager to terminate the execution of a task by a contractor.

This protocol is visualized in the sequence diagram shown in Figure 1, created by FIPA, the Foundation for Intelligent Physical Agents [FIP02].

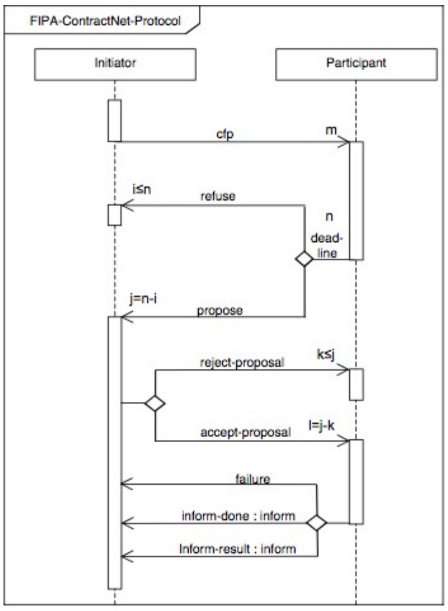

Figure 1: Contract net protocol as specified by FIPA.

Discussion A trade-off that is made in CNET is the fact that an agent can accept multiple contracts and is responsible for queuing them. This can potentially lead to worse overall system performance and a more unbalanced load distribution than when each agent is allowed only 1 task at a time. The advantage of this choice is that an agent does not have to wait for an outstanding bid to be accepted or refused before it makes another bid.

In his paper [Smi80], Smith mentions some additional complications with the proposed negotiation procedure and he proposes some potential solutions to these complications.
The first one is the fact that there should be the option for a busy agent to immediately reply to bids, informing the manager of the fact that they can not accept the task. Currently the agent queues the requests for handling until he has resources available. This allows the manager to make a more suited decision when it receives no bids for its announced task.
The second improvement is the usage of directed contracts. This can be used when the manager knows which agent is suited for executing the task. It directly sends an award message to that specific agent, skipping the announcement and bidding phases. These directed contracts are combined with acknowledgements to ensure that the agent is only given a task that it can execute.
The third proposal is to use request and information messages that can be used by agents to gather information from other agents in the network.

The fourth and final improvement is the usage of node available messages. This can be summarized as the reversal of the normal negotiation procedure. A node that has resources available for a new tasks can send such a message to inform other nodes that it would like to receive a new contract.

Confirmations An issue with the protocol as described by Smith is the following: what should an agent do with its resources if it bids on multiple requests? Should it reserve those resources until it is awarded the task? What should happen if it isn't awarded the task and how long should it wait?
Schillo et al. propose an improvement to the CNET protocol called *The Contract-Net with Confirmation Protocol* (CNCP) [SFK01] to address these issues.
The difference with the normal CNET is that an agent does not commit to executing a task when sending a bid. The commitment is delayed as long as possible in the process. The initiator of the negotiation process collects all bids and sorts them by value. It starts with sending a confirmation request to the first one to ask if that agent can actually execute the task. If the agent can do so it replies to accept the task and only then commits to reserving resources and executing the task. If that agent fails to accept then the initiator moves on to the next bid in the ordered list. This process continues until the task is accepted by an agent or no more agents are left.

The advantage of this approach is that only one agents needs to commit to executing the task and reserve resources for it. A disadvantage is the extra overhead on the initiator to keep track of the ordered list of bids. In practice however this overhead can be kept to a minimum and the advantage outweighs the disadvantage.

2.2 Fields

The use of fields in multi-agent systems is inspired by the laws of physics, as explained by Mamei et al. in [MZL04]. In their paper they make use of distributed computational fields to guide tourists around an unfamiliar museum. In [WBH06], Weyns et al. propose gradient fields as an alternative to and improvement of the contract net approach of task assignment to be able to cope with a more dynamic environment.

Generating a field The value of the field at a given point depends on the objects and agents in the environment. Those objects and agents emit either attractive or repulsive fields. By emitting an attractive field a goal can attract other agents to its location. Repulsive fields are emitted by obstacles and agents, for example to avoid collision or to incorporate competition amongst agents. Agents needs a mechanism to sense the emitted values of the (nearby) environment to aggregate them. It can then act and move based on that aggregated value or vector.

3 Multi-agent system design

3.1 Design

We propose a multi-agent system design specific to a taxi pickup and delivery problem (PDP). The system has two types of agents: **Customers** and **Taxis**. The customer represents a group of people that want to be picked up, therefore having a specific capacity that they require.
Between these agents three types of interactions exist: contracting between customers and taxis, field repulsion between taxis and trading between taxis. The proposed design is applicable to any taxi PDP in general, but some design choices are motivated for our specific use case of Manhattan.

3.1.1 Contract net

A contract negotiation is initiated by a customer, taking on the role of manager, using a broadcast `ContractRequest` to indicate that it wants to be picked up by a taxi. Taxis receiving this request which have enough free capacity reply to the request with a `ContractBid`. If they do not have

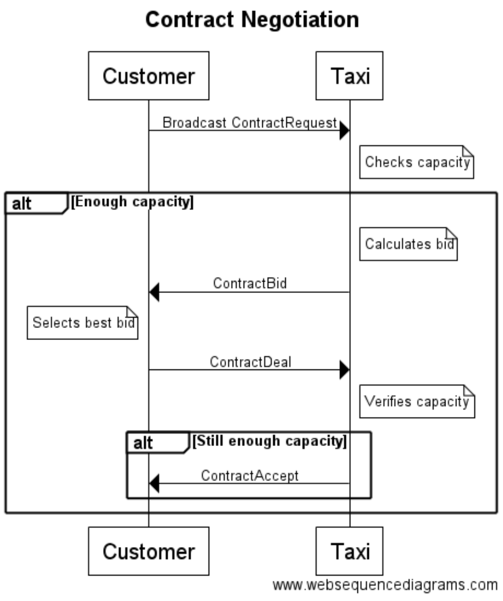

Figure 2: Contract negotiation between Customer and Taxi.

free capacity available for the request, the request is ignored. Taxis are allowed to bid on all incoming requests at the same time. The customer selects the best out of all the bids and sends a `ContractDeal` to the winning taxi. Finally, the taxi should confirm that it can still honor the deal by replying with a `ContractAccept`. This indicates the end of the negotiation procedure and the taxi, taking on the role of contractor, is now responsible for picking up and delivering that customer. This procedure is summarized in Figure 2. It is possible that a customer receives no bids following its request. In that case the customer sends a new request and the negotiation process is repeated.

Taxis have a predefined capacity indicating the maximum number of people that they can transport at once. In our design we propose a system of *shared taxis*. This means that a taxi can take on and execute multiple contracts at the same time, meaning picking up multiple groups of customers at once.
Having multiple customers means that the taxi should calculate an efficient route to pick up and deliver these customers. The shortest route is found by generating all valid permutations of locations the taxi has to visit and calculating their length. A route is valid if for every customer C their pickup location is visited before their delivery location. The total amount of concurrent pickups is limited to keep the route calculation feasible with this simple algorithm given that the amount of permutations for a list with n elements equal is to $n!$ (although this number is reduced by enforcing the validity of routes). We propose limiting the number of concurrent contracts for one taxi to three.

The value of the bid placed by a taxi is calculated using the route algorithm described above. When a taxi receives a request it calculates the length of the route taking into account its current customers and the new customer. The bid is then calculated as the length of that route divided by the speed of the taxi, indicating the time it takes to complete the route.

Both taxis and customers have an unlimited and fully reliable communication range. This design choice is justified because this research is not interested in those factors and is achievable in real life applications as well. Messages are checked and sent every time period of the experiment.

3.1.2 Field

The Contract Net Protocol does not specify what idle agents should do when they currently have no task assigned to them. In our use case of taxi PDP this can be translated to: what should a taxi do when it has no customers? We propose the use of a field generated from historical pickup data to guide the idle taxis to hotspots where it can be expected that new customers will appear. This is expected to reduce the average pickup waiting time for the customers as the idle taxis should be closer by their pickup locations.

The field is a discretization of the historical data in three dimensions: longitude (X), latitude (Y) and time (t). A lower subdivision for the time dimension means grouping more time instances together. The X and Y dimension are discretized as well, creating so called *bins* in the matrix. This discretization simplifies storing and using the field and allows for a large amount of data to be used efficiently. The value of each bin in the field is calculated per time frame by placing the historical pickup data into the bins of this matrix. The higher the value of a bin, the higher the attraction will be for the nearby taxis.

When a taxi is in the idle state it calculates a displacement vector using the current time frame of the field. A default search radius is specified to avoid all taxis from converging to the same global optimum in the field. If no attracting bin is found within a default search radius, the taxi will expand his search until one is found. The displacement vector as defined by the field is updated using information from nearby taxis. This information is passed to other taxis by broadcasting `PositionBroadcast` messages containing the location of a taxi and their free capacity. This free capacity is taken into account to calculate the amount of repulsion for the taxi. Taxis which are full, therefore having free capacity equal to zero, will not repel other taxis as they are not a competitor at that time. The displacement vector is updated by increasing the value in the direction opposite to the other taxi and proportional to its free capacity.

This approach of using historical data for generating the field is very straightforward to implement but requires the assumption that the data from the previous year is a good indicator the the pickups of the current year. An example of the field is shown in Figure 3 for the Manhattan pickup data. More advanced machine learning and data mining techniques could be used to generate a better predictive field. For these techniques we refer to the literature as it is out of the scope for this research project.

3.1.3 Trading

Because of the dynamic nature of the system it is possible that the optimal routes to pick up and drop off customer changes over time. This means that a bid that is chosen by a customer as the best one might not be optimal anymore at a later point in time. Taxis could further optimize their routes by communicating with each other and trading contracts. To achieve this improvement we propose a simple trading mechanism.

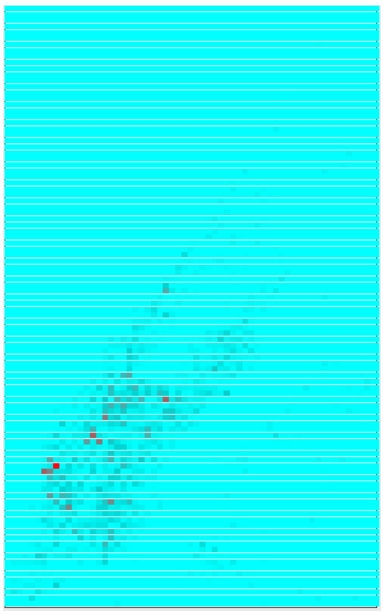
Figure 3: Discretization of the field with values ranging from blue (low) to red (high).

Trading is implemented using a negotiation process similar to the CNET protocol described earlier. A taxi which has contracts for which the customers are not yet picked up, can initiate a trading request by sending a `TradeRequest` message to nearby taxis. The `PositionBroadbast` messages as defined in the previous section can be reused to locate nearby taxis. For this research we limit the trading range to 2.0 - 2.5km.
The contents of this message specify the information needed by the recipient to decide whether or not they want to trade. More specifically: the initiator of the trade request calculates the length of the new route that is obtained if they were to remove the pickup and delivery of the customer from their current route. The message contains this route reduction value and information regarding the customer itself. The recipient of the message then calculates the length of the route they will take if they accept the trade.

The profit of the trade is calculated as the route reduction of the initiator minus the increase in route length of the recipient. If this profit is above a certain threshold to avoid negligible trades, the recipient of the trade request sends a `TradeDeal` message back to the initiator. The initiator collects all received deals and finds the one with the

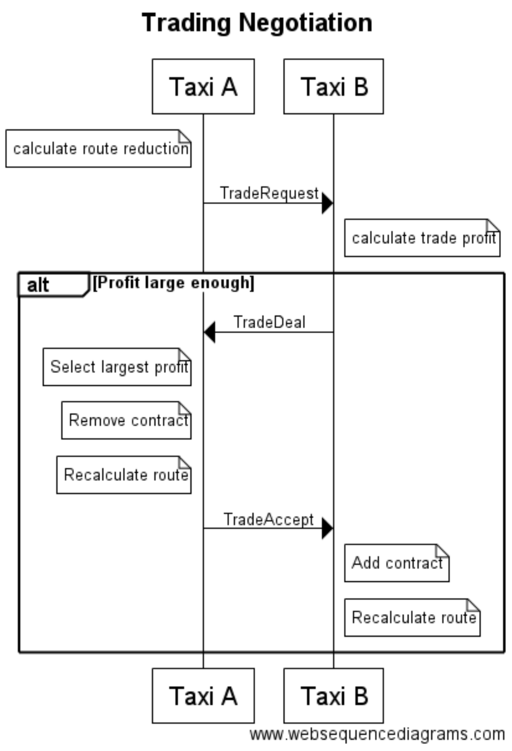

Figure 4: Trading negotiation between Taxi A and Taxi B.

largest profit. It then sends a `TradeAccept` to that taxi. Due to the capacity limitations of a taxi it can only send one trade deal at the same time. It should then wait for the accept or, if the trade deal is not accepted, until the deadline has passed. Only then can it send new trade deals for other requests. This process is summarized in Figure 4 as a sequence diagram.

3.2 Comparison with existing MAS

Contract Net Protocol In the original CNET protocol as described in Section 2.1, agents are only allowed to bid on 1 contract. Our proposal does not limit the taxi agents to 1 bid at a time. The agreement is ratified using a confirmation sent by the taxi agent, similar to the CNCP described in Section 2.1. CNET also states that agents should keep track of the history of sent requests to handle them when they have more resources available. Our implementation of the protocol does not specify a deadline in the messages as its implicitly imposed by immediately handling all messages. This means that if a taxi can not accept a customer it ignores the `ContractRequest`.
This approach has several advantages. The first one is that a customer does not have to wait until the deadline has passed to select the best bid. It can immediately send a contract deal to the highest bidder. The second advantage is that taxis can bid on multiple contracts at once because they can still refuse a deal afterwards.
The disadvantages of this technique are that there is an extra confirmation message that needs to be sent and that the customer needs to resend a contract request if it did not receive any bids in the next tick.

Field based techniques In the field techniques described in Section 2.2, fields are used to guide agents towards goals. In our proposed system a field is used in a somewhat different way. Taxis are only guided by the field in their idle state. The value of the field does not depend on the actual customers (goals) but only on historical pickup data and other nearby taxis in the environment. Taxis are assigned to goals by the contract net protocol.

Trading The trading element of the system as described in 3.1 uses a protocol similar to the contract net protocol without confirmation. The deadline is implicit and the same for every trading message, namely one tick. The agents expect an answer to their trade request to be sent in the next tick, therefore making it available to them two ticks after they sent their request.

4 Experiments

To get a clear view of the influence of the proposed improvements, several experiments are executed using real taxi data. This data consists of all the pickup and deliveries from the yellow cabs in Manhattan. The data sets are available on the website of The City of New York [NYC]. The field is generated using historical data from January 2015 and the data which is sampled to generate customers with their needed capacity and true pickup and delivery locations comes from January 2016. This data is matched up with a 52 week difference so weekdays line up with each other, as opposed to matching the date.

4.1 Setup

For conducting the experiments the design as outlined in Section 3.1 is implemented in Java using the RinSim logistics simulator [vLH12].
The data is preprocessed using an R script to remove the unneeded fields, filter the data for Manhattan and round the coordinates to lower the needed storage space. After this preprocessing step the data is stored in a MySQL database for easy access using a custom `MySQLDataLoader` in RinSim. The data can be efficiently queried using an index on the pickup time column.

The experiments are simulated over the randomly sampled data of a single day and are initiated with an amount of randomly placed taxis that matches the customer sample factor. The taxis have a maximum capacity of 5 individuals. A seed is used for the random number generator to ensure consistency between experiments. The experiments start at midnight and finish when all the customers have been delivered and thus all the taxis are idle.

The simulation is run on a plane road model where the shortest route is always a straight line from pickup to drop off location and no collisions between taxis can occur. This approach is chosen as it will have the least influence on the results that this research is interested in.

4.2 Results

The main statistic to examine the results is a box plot of the customer's waiting time for a pickup and the average of these pickup waiting times.
The initial experiments investigate the basic influences of our improvements, where the first experiment is the basic PDP experiment without any of the proposed improvements to get a comparison base, the second one uses the field, the third one trading and the last one combines both improvements.

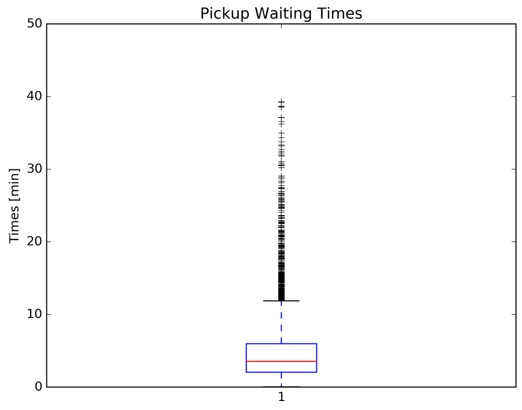

Figure 5: Pickup Waiting time box plot for the default PDP configuration without any improvements.

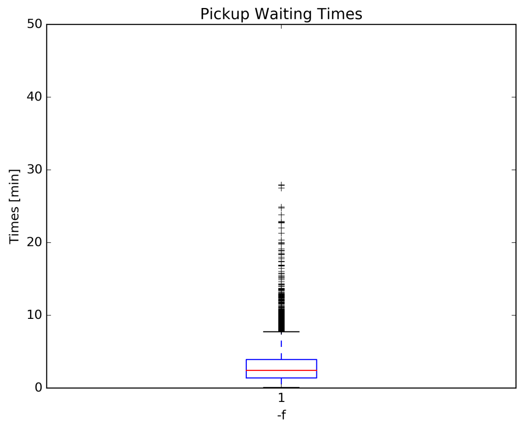

Figure 6: Pickup Waiting time box plot for the experiment using the field improvement.

The usage of the historical data field reduces the average pickup waiting time with 40%: from an average of 5 minutes to 3 minutes. The trading experiment does not yet yield improvements due to poorly chosen default values for other parameters. The distribution of the pickup waiting time in the basic experiment and the experiment with the field enabled are shown in Figure 5 and Figure 6. These results show that using the field reduces the average waiting time for the customers.

Using only the field improvement, the default parameters are varied to find more optimal values. These parameters include the

field discretization, empty taxi influence, a limit on the travel distance for idle taxis and a reduction of the amount of concurrent customers.
The idle travel distance limit leads to slightly worse results which shows that the generated field is a good approximation of the real scenario. This parameter will probably be used in a real life cost trade-off function, taking into account the cost of driving a taxi.
The time discretization affects the results the most as it provides some generalization and prediction. Each frame containing 2 minutes of data results in the lowest average pickup waiting time.

4.3 Analysis

To analyze the results of our best parameter model we run the experiment again over two full days (from January 12 2016 00:00.00 until January 14 2016 23:59.59) using no improvements, using only the field improvement and using the field combined with trading.
Using the improved parameters the customers in the CNET never had to resend any trade request.

Figure 7 shows the *travel time overhead* caused by accepting multiple customers at once in a taxi for the experiment using both the field and trading improvement. This overhead ratio is calculated for each customer as follows: it is the total distance it has traveled in the taxi divided by the shortest distance from its pickup to delivery location. This denominator represents the straight line distance in our plane road model, which is the route the taxi would take if it only picks up that customer. These results show that the overhead stays low enough to allow accepting multiple passengers for an overall efficiency increase.
Figure 8 shows the trade profits calculated as route length reduction in individual taxi routes, also for the same experiment where both the field and trading is used. The minimum trade profit is set to 5 for all experiments to avoid small trades with negligible profits. The histogram clearly shows that

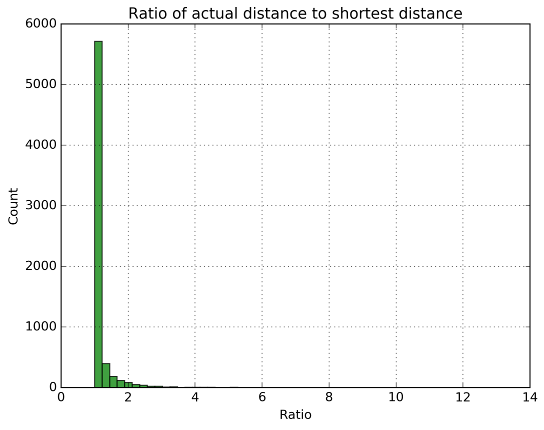

Figure 7: Customer travel time overhead for the experiment with field and trading improvement.

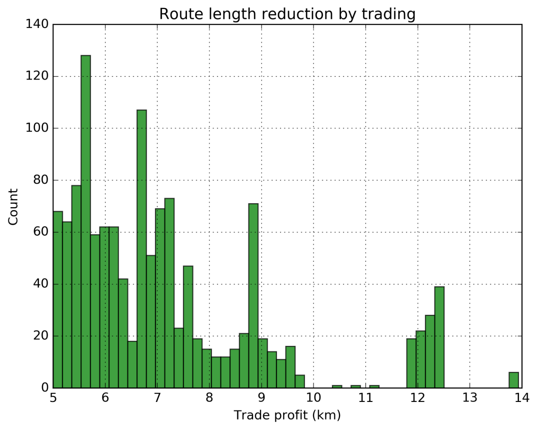

Figure 8: Histogram of trade profits (in km) for all trades.

the trading system is useful for reducing the total distance traveled by the taxis as there are many trades with a high profit.

Figure 9 shows that the trading mechanism decreases the average pickup time but results in a few more outliers. This can be explained by the fact that there are some contracts which are often traded between queues in some taxis and ending up at the end of the route whilst other customers are picked up faster. Section 4.4 proposes an enhancement to the system to avoid this starvation.

Finally, Figure 10, Figure 11 and Figure 12 show the evolution of the number of idle taxis and waiting customers during the simulation for respectively the experiments

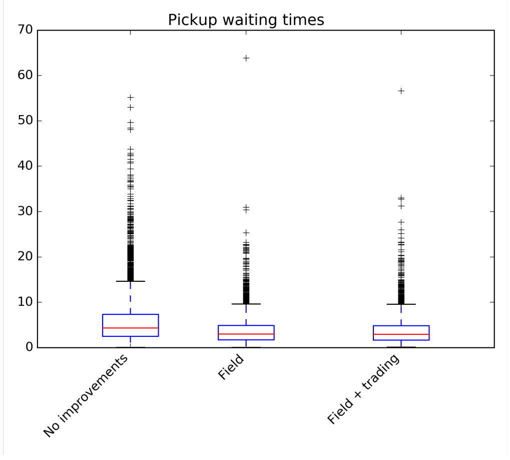

Figure 9: Pickup waiting time box plots for the final experiments.

using no field and no trading, using only the field and using both the field and the trading improvements. These results show that our improvements have a positive influence on these measures by lowering the number of waiting customers at all times and most noticeable at peak moments like 18:00.

4.4 Enhancements

We also propose some possible further improvements for future research.

The first improvement is to avoid or reduce the risk of starvation of customers in taxis. Currently the system does not guarantee that a customer will be delivered before a given deadline. This can be taken into account by modifying the route calculation algorithm to not only use the length of the route as a measure but also the deadlines for delivering customers and giving higher weight to customers which have been waiting for a longer time. The trading system should also take into account customers that have been waiting for a long time.

The second improvement that we propose is to make the field smarter. This can be achieved in two ways.
The first one is to use a machine learning algorithm to make use of the large amount of historical data that is available for taxi pickups in several cities. There are many al-

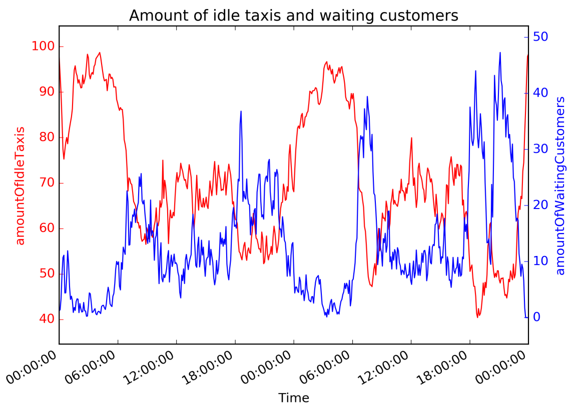

Figure 10: Evolution of idle taxis and waiting customers without experiment improvements.

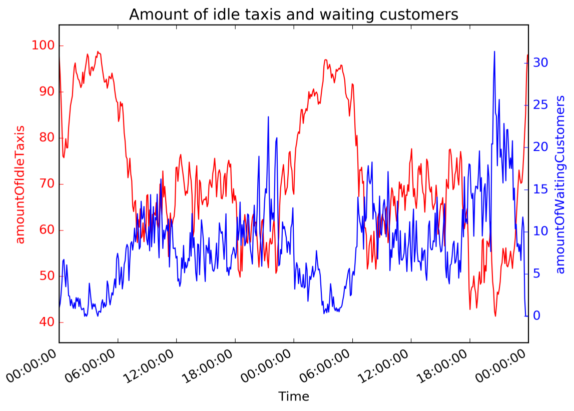

Figure 11: Evolution of idle taxis and waiting customers with field improvement.

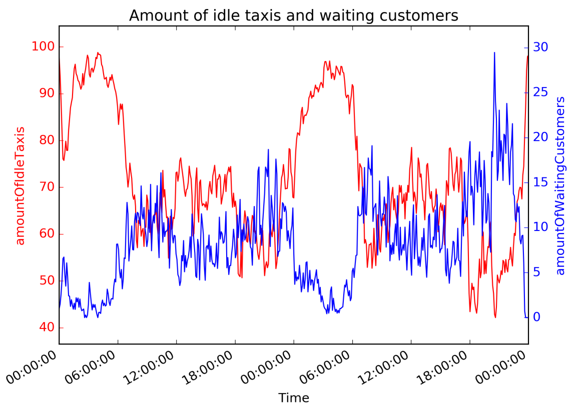

Figure 12: Evolution of idle taxis and waiting customers with field and trading improvements.

gorithms available for forecasting using time series data. We refer to the book by Hamilton [Ham94] for an overview.
The second way to achieve an improved field is to integrate knowledge regarding the current customers and a wider range of nearby taxis. This can be implemented using messages sent between taxis to propagate their local knowledge.

5 Conclusion

After the analysis of our results, we can now answer our questions and check our hypotheses.

1. *Is a discrete field a scalable way to represent historical data?* The discretization of the field allows memory and calculation constraints to be handled. It does not take up much space and is shareable with every taxi so only has to be calculated once in advance. We conclude that the field representation is indeed a scalable way to represent the historical knowledge.

2. *How does the idle travel distance influence the change in pickup time?* The pickup time increases slightly when posing a limit on the idle travel distance as expected. This distance can be limited as conclusion of a real life cost analysis, taking into account factors like fuel expenses for taxis. For this research project, no limit on the idle travel distance resulted in the fastest pickups but that would be unfeasible in real life situations.

3. *What is, if it exists, the optimal idle travel distance?* The limit is as expected the equilibrium achieved by not constraining the idle travel distance.

4. *Will the average waiting time for the customers decrease?* The average pickup waiting time did decrease immensely for all customers as shown in the previous section.

References

[BY14] Michael R. Bloomberg and David Yassky. 2014 Taxicab Factbook. *Taxicab Factbook*, pages 1–13, 2014.

[FIP02] FIPA. *FIPA Contract Net Interaction Protocol Specification*. FIPA, 2002.

[Ham94] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[MZL04] Marco Mamei, Franco Zambonelli, and Letizia Leonardi. Co-fields: A physically inspired approach to motion coordination. *IEEE Pervasive Computing*, 3(2):52–61, 2004.

[NYC] Nyc taxi & limousine commission - trip record data. `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`. Accessed: 2017-04-26.

[SFK01] Michael Schillo, Klaus Fischer, and Tore Knabe. Technical Memo The Contract-Net with Confirmation Protocol : An Improved Mechanism for Task Assignment DFKI GmbH German Research Centre for Artificial Intelligence. 49(631), 2001.

[Smi80] Reid G Smith. The Contract Net Protocol : High-Level Communication and Control in a Distributed Problem Solver. C(12):1104–1113, 1980.

[vLH12] Rinde RS van Lon and Tom Holvoet. Rinsim: a simulator for collective adaptive systems in transportation and logistics. In *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems*, pages 231–232. IEEE, September 2012.

[WBH06] Danny Weyns, Nelis Boucké, and Tom Holvoet. Gradient field-based task assignment in an AGV transportation system. *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems - AAMAS '06*, page 842, 2006.