Multiagent Self-organization for a Taxi Dispatch System

Aamena Alshamsi
Faculty of Informatics
The British University in Dubai
P.O.Box 502216, Dubai,
United Arab Emirates
aamena_alshamsi@yahoo.com

Sherief Abdallah
Faculty of Informatics, The
British University in Dubai
P.O.Box 502216, Dubai, UAE
(Fellow) School of Informatics,
University of Edinburgh
Edinburgh, EH8 9LE, UK
sherief.abdallah@buid.ac.ae

Iyad Rahwan
Faculty of Informatics, The
British University in Dubai
P.O.Box 502216, Dubai, UAE
(Fellow) School of Informatics,
University of Edinburgh
Edinburgh, EH8 9LE, UK
irahwan@acm.org

ABSTRACT

The taxi dispatch problem involves assigning taxis to callers waiting at different locations. A dispatch system currently in use by a major taxi company divides the city (in which the system operates) into regional dispatch areas. Each area has fixed designated adjacent areas hand-coded by human experts. When a local area does not have vacant cabs, the system chooses an adjacent area to search. However, such fixed, hand-coded adjacency of areas is not always a good indicator because it does not take into consideration frequent changes in traffic patterns and road structure. This causes dispatch officials to override the system by manually enforcing movement on taxis. In this dissertation, I apply a multiagent self-organization technique to this problem, dynamically modifying the adjacency of dispatch areas. We compare performance with actual data from, and a simulation of, an operational dispatch system. The proposed technique decreases the total waiting time by up to 25% in comparison with the real system and increases taxi utilization by 20% in comparison with results of the simulation without self-organization. Interestingly, we also discover that human intervention (by either the taxi-dispatch officials or the taxi drivers) to manually overcome the limitations of the existing dispatch system can be counterproductive when used with a self-organizing system.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Experimentation

Keywords

Multiagent Systems, Reoganization, Network

1. INTRODUCTION

The taxi dispatch problem is the problem of assigning taxis to callers waiting at different locations. A good taxi dispatch system satisfies all dispatch parties: the potential passenger and the taxi driver. From the passenger's perspective, the waiting time should be minimal. From the taxi driver's perspective, the travel time of the taxi to pick up the customer (empty cruise time) and the idle time should be minimal as well. There are other goals that need to be taken into consideration such as maximizing the number of dispatched calls and maximizing the total number of served calls per taxi. One of the approaches to solve this problem is to use fixed regional dispatch areas. These regional areas are identified by their Global Positioning System (GPS) locations and obtained by taking into account traffic, street structures and population density. The location of each incoming call or taxi is identified by its corresponding dispatch area. This approach can be mapped to a multiagent system as follows. Each area is an agent; each call is a task; and each taxi is a resource. The network of the areas interacts to run the dispatch process.

The majority of the previous studies in the taxi dispatch problem did not utilize multiagent systems. They focused on the assignment method aiming at making an assigned taxi reach the customer in the shortest time possible using different parameters such as road networks and real time traffic [2] [5]. Other studies incorporated the satisfaction of drivers in the assignment method [11].

Our work uses a self-organization technique to optimize the performance in a taxi dispatch system. We develop a computer simulation of a real taxi dispatch system (initialized using actual data obtained from a major taxi operator) to demonstrate that our proposed approach outperforms the conventional existing system in terms of customer waiting time (respecting customer satisfaction) and outperforms the simulation of the existing system in terms of the number of dispatched calls per taxi (directly proportional to fleet utilization).

This work advances the state of the art in taxi dispatch systems by making three main contributions. First, we develop a realistic simulator of the taxi dispatch problem that is based on real data acquired from a taxi dispatch company in the middle east. The simulator does not only follow that same dispatch process, but also simulates some human behaviors of the dispatch parties such as taxi drivers and dispatch officials. Second, to our knowledge this is the first work to apply a self-organization technique on a taxi dispatch problem and benchmark it against the performance of an existing operational system showing superior performance. Third, we list a number of lessons about modeling

Cite as: Multiagent Self-organization for a Taxi Dispatch System, Aamena Alshamsi, Sherief Abdallah and Iyad Rahwan, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

the taxi dispatch domain in order to provide key foundation for further researches. In particular, by comparing the performance of the simulation with real-world data, we highlight some key human factors that need to be taken into account when modelling this domain.

The rest of the paper is organized as follows. Section 2 describes the existing taxi dispatch system that we simulated. Section 3 describes our simulator and how it mimics the existing system. Section 4 describes our self-organization technique. Section 5 presents and discusses our experimental results. Section 6 reviews the previous work. At the end, Section 7 concludes and suggests possible future extensions.

2. THE EXISTING TAXI DISPATCH SYSTEM

This work relies on the existing dispatch system of a company that is operational in a city with high population density in the Middle East.[1] The existing taxi dispatch system uses fixed ragional areas to dispatch calls. The main city is partitioned into 131 areas besides 10 more areas of surrounding cities. The classification of areas depends on population density, traffic, customer demand and other factors related to the dispatch process. Boundaries of areas are identified by GPS coordinates and each area has its own *cab queue* and *call queue*. The *cab queue* contains a list of vacant cabs that are located within an area in a descending order according to the vacancy period and it keeps track of availability of those cabs. It is updated whenever a cab enters the area or leaves it or the availability status changes from “vacant”to “occupied”or “on call”or vice versa. The *call queue* of an area contains a list of waiting calls that originate in that area in a descending order according to the call’s waiting time.

2.1 Dispatch Policy

The system uses the following dispatch policy (based on interviews with officials from the taxi company). Whenever a customer calls the dispatch center, his/her credentials, location, time, the number of passengers and preferences are entered into the system. Based on the address provided, the call operator tries to identify the customer’s suburb location. Then, the call is placed in the area that contains the defined suburb given that each area contains a number of suburbs. Based on that, the system automatically starts looking for a vacant cab within the selected area. If there are vacant cabs within the selected area, the system assigns the call to the cab that has been vacant the longest. If the call is placed in an area with no vacant cabs available, the system starts looking, after one minute, for the cab that has been vacant the longest within a predefined list of adjacent areas. If there are no vacant cabs in the main area and the adjacent areas, the system waits for a cab to become available in either the main area or the adjacent areas. If the waiting time for any call has exceeded 1 hour, the call is cancelled.

2.2 Adjacency Schedule

The adjacency of areas is defined in a schedule based on the experience of the dispatch center officials and some statistics of high-demand areas. The schedule is updated regularly (e.g. every few months) manually to take into consideration changes in traffic patterns, so high-traffic areas need adjacent areas. The adjacency of areas neither necessarily reflects physical adjacency nor implies symmetric adjacency of areas in the schedule.

[1]For confidentiality reasons, we are unable to disclose the name of the company at this stage.

2.3 General Statistics

Table 1 displays important statistics during the period 6/8/2007 - 11/9/2007, from 9:00 am to 3:00 pm (in order to avoid rush hours). [2]

These statistics are used to benchmark our simulator against the existing system and later to evaluate our proposed approach against the existing system.

Parameter	**Average in minutes**	**Standard Deviation**
Dispatch Time	5.45	9.8
Pickup Time	14.5	17.2
Total Waiting Time	20	20.3

Table 1: General Statitisics

2.4 Limitations of the existing dispatch system

The existing system has a number of limitations that are described in this section. We later show how our approach addresses these limitations.

2.4.1 *Customer Satisfaction and Total Income*

The system uses a dispatch policy that is biased in favor of the driver’s interest rather than the customer’s interest. More importantly, the current system does not take into account the profit of the taxi company as a whole: priority is set to cabs that have been vacant the longest rather than minimizing the total waiting time of the customer.

2.4.2 *Adjacency of Dispatch Areas*

The existing system defines the adjacency of areas manually every few months. This manual adjustment can be suboptimal because it relies on human judgment and only occurs at rather long intervals.

3. SIMULATING EXISTING SYSTEM

It is impractical to deploy a new approach immediately in the real world without carefully studying the new approach in a simulated environment. The purpose of this study is to make a case for incorporating the multi-agent approach by showing its potential benefit in a simulated environment based on real data.

This section describes our simulator. Table 2 lists important terms that are used in this section and the rest of the paper.

Our simulator uses the same dispatch policy used by the current system and also uses real data in generating calls and computing travel delay. The simulator takes into consideration uncontrollable factors such as the number of incoming calls, traffic, and availability of cabs in an abstract manner that is reasonably accurate yet tractable (more details later). The simulator starts with a number of cabs distributed among areas. Then, at each time unit (minute), it checks for incoming calls, attempts to dispatch them along

[2]Fridays and Saturdays which are weekend days in the city are excluded. Also, Thursdays are excluded.

Term	Description
Cab or Car	Taxi
Dispatch Time	How long the system takes to dispatch an incoming call (in minutes)
Pick up Time	how long an assigned cab takes to pickup a customer since the call is dispatched (in minutes)
Total Waiting Time	Pick up time + Dispatch Time
Job or task	Call
Search Areas or Dispatch Areas	Areas predefined by the existing system
Customer or Caller	Potential Passenger
Main Area	Caller's Area
Utilization of a neighbor area	The usefulness of an adjacent area measured by the number of times an area assigns a call to a vacant cab in this neighbor area

Table 2: Terms used in the document

with the waiting calls according to the dispatch policy and availability of cabs and moves cabs to their destinations. At the end of a simulation run, various statistics are collected, including taxi utilization and average total waiting time.

3.1 Real Data

Average and standard deviation are calculated on some key data obtained from the real system. The period of time whereby data is obtained is identified to ensure accuracy since traffic differs at peak hours versus other remaining hours and differs also in working days versus weekends. The period is defined to be from 6/8/2007 till 11/9/2007, where Thursdays, Fridays and Saturdays are excluded and time is limited to be from 9:00 am till 3:00 pm.

Fleet Distribution.
The taxi dispatch database contains the number of cabs in each area at each hour of the day. Thus, different distributions are obtained from the database and fed into the simulator. To run the simulator, a number of cabs are initially distributed among areas by identifying the number of cabs at each area. At time 0 of the simulation, all cabs are vacant.

Call Distribution.
A call log was taken from the dispatch database. We have only used data about successful calls in our simulator, because unsuccessful calls lacked some details. The log contains the following attributes for each call: call time, location (area) and destination (area). This is used to generate tasks in various simulation runs.

Trip Duration.
It is difficult to calculate the duration of trips in the simulator because many parameters are needed to be taken into account such as traffic, road structures and GPS locations. Instead, our simulator assumes the duration between any pair of areas follows a normal distribution. The average and standard deviation of the duration between each pair of areas is calculated from the real-data. This captures (approximately) many factors that are very difficult to simulate such as traffic and street structures.

Area Adjacency Schedule.
The latest schedule of area adjacency is used which lists all areas and their corresponding adjacent areas if available.

Dispatch Policy.
The policy is described earlier in Section 2.1. To run the simulator, one should select a predefined time/day period. Based on the obtained data on the selected period, a number of cabs are distributed among the areas. Then each minute, the simulator checks for incoming calls based on the obtained call log at the predefined period. Meanwhile, the simulator dispatches calls according to the defined dispatch policy, moves cabs toward their destination, keeps track of cabs and calls details and moves some vacant cabs to congested areas if needed. At periodic intervals of the simulation (e.g. 300 time units) and at the end a the simulator run, statistics are displayed showing the following:

1. The total number of completed calls, dispatched calls, cancelled calls and waiting calls.
2. The maximum, minimum, average and standard deviation of number of calls served by a taxi.
3. The maximum, minimum, average and standard deviation of calls' dispatch time.
4. The maximum, minimum, average and standard deviation of calls' pick up time.
5. The maximum, minimum, average and standard deviation of calls' total waiting time.

3.2 System Behavior

The existing dispatch process is summarized in algorithm .1.

Algorithm .1: Dispatch Algorithm

```
Input: Calls and Cabs
Output: Dispatching calls to cabs
foreach incoming call or waiting call $i$ with waiting time $\leq$ 60 minutes do
    if there is a vacant cab $j$ in the call's area then
        Call $i$ is dispatched to cab $j$;
    else if waiting time >1 minute then
        if there are vacant cabs $l$ in the adjacent areas of call's $i$ area then
            Call $i$ is dispatched to the cab $s$ that has been vacant the longest among cabs $l$;
        else
            Call $i$ is added to the end of the waiting list
        end
end
```

The dispatch algorithm above does not account for all actual behavior in practice. This is because drivers sometimes work autonomously, driving around and picking up passengers from the street, etc. Also, drivers get ad hoc instructions from the dispatch center or the operation section. We simulate the behavior of drivers in moving toward congested areas by using four strategies to move cabs that stay vacant to areas that have shortage in vacant taxis. This applies only to movements not controlled by the dispatch center. The strategies have been set according to interviews presented with dispatch officials since the dispatch process is affected by human interference. We tried to simulate the following main dispatch parties' behaviors: the movement of drivers to congested areas regardless of their current location, instructions provided by some operation officials to send taxis to areas that are *expected* to have high demand, and instructions provided by some dispatch officials to send nearest taxis to areas with high demand. Moreover, different sets of strategies have been used to test the simulator as shown below.

Strategy1 Vacant cabs in other cities moves to predefined destinations;

Strategy2 The cab that is vacant the longest moves to the area with maximum number of waiting calls;

Strategy3 Vacant cabs in other cities moves to areas with maximum number of waiting calls;

Strategy4 Nearest vacant cab moves to area with calls that have being waiting the longest;

The strategies are grouped as follows for testing purpose:

The strict strategy (Combination 1) consists of strategies 1 and 3. Thus, each vacant cab outside the main city moves to the area with maximum number of waiting calls. Otherwise, the cab moves to the predefined destination according to the cab's location.

The flexible strategy (Combination 2) consists of strategies 1, 2, 3, and 4

4. SELF-ORGANIZATION

The existing dispatch system uses a hard-coded schedule of adjacency for the dispatch purpose. Each area uses its neighbor(s), if it has shortage in taxis while receiving a call, and assigns the call to the cab that has been vacant longest among the adjacent areas (For more details, please refer to Section 2). Some areas have no adjacent areas, and others have more than one adjacent area.

Since the adjacency schedule is hand coded and has been set by dispatch experts, it doesn't take into consideration frequent changes in street structures or traffic. We propose in this section a self-organizing approach that automatically adapts the areas' adjacency (starting from the hand-coded adjacency schedule).

Algorithm .2 shows our proposed self-organizing mechanism, which works as follows. Each area j that has waiting calls more than a threshold T_w (we use $T_w = 10$ in our experiments) checks its neighborhood. If area j has a neighbor i whose utilization (number of times that the main area j assign its calls to area i) is greater than threshold T_u (our experiments use $T_u = 10$), it checks area i's neighborhood. The algorithm uses parameter p_o to control the frequency of self-organization: with probability p_o self-organization occurs. If area i has neighbor z that has the maximum utilization (among neighbors of area i) that is greater than threshold T_i (our experiments use $T_i = 10$), then area z is added as a neighbor of area j (if it is not already a neighbor of area j). Furthermore, the area with the maximum number of vacant cabs (the maximum number of vacant cabs should be at least 5 in our experiments) offers to be added to the neighborhood of area j with a certain probability. This strategy is necessary to connect areas that are fully isolated (the hand-coded adjacency schedule contains areas with no neighbors). To avoid excess in number of neighbors of an area, the neighbor with minimum utilization is removed until the number of neighbors reaches threshold N_{max}(we use $N_{max} = 5$ in our experiments).

Algorithm .2: Strategy 1 for self-organization

```
Input: Current Adjacency Schedule
Output: Adjacency Schedule is self-organized
foreach area i at time n with pending calls >= $T_w$ do
    with probability $p_o$= n, if area i has a neighbor j
    with utilization >= $T_u$ and area j has at least one
    neighbor z that has the max utilization among
    neighbors of area j that is >= $T_i$ ;
    then
        add area z as a neighbor of area i;
    end
    with probability $p_o$, let area y be the area with the
    max number of vacant cabs among all the areas.;
    add area y as a neighbor of area i;
    while the number of i's neighbors > $N_{max}$ do
        remove the neighbor with minimum utilization;
    end
end
```

5. EXPERIMENTAL RESULTS

We have conducted experiments for the simulation of the existing system and the simulation with self-organization technique. We have used different settings such as distribution of cabs among areas at the first run of the simulator and different movements of vacant cabs. The purpose of the experiments is to compare the performance of the simulation to the existing system and to compare the simulation with self-organization to the existing system and to the simulation of the existing system. The evaluation includes the following measurements desribed in Table 3.

Measurement	**Abbrev.**
Averaged dispatch time	*ADT*
Averaged pick up time	*APT*
Averaged total waiting time	*ATWT*
Percentage of cancelled calls (Cancellation Rate)	*CT*
Percentage of waiting calls at a given run (Waiting Rate)	*WT*
Percentage of served calls (Success Rate)	*ST*
Average total number of calls served by each cab (Fleet Utilization)	*FU*
Averaged degree distribution	*ADD*

Table 3: Terms used in the experiments

The total number of cabs during any time of day or night is approximately 2400 cabs. We selected a fleet distribution with a total of 1622 cabs to all of the conducted experiments. This distribution is captured on 12/08/07 at 9:00 am. We tried also distributions with lower total number of taxis such as 793 (the total number of vacant cabs on the same day and hour) which shows roughly the same performance. However, for brevity we display only the distribution with total of 1622 cabs. Note that not all the trips in the real system are obtained via calls. Some cabs pick up passengers from the street which is not covered by the simulation but implicitly it is taken in to consideration. The simulation is not utilizing the whole fleet, so that some of the cabs do not receive any call during the total runs of the simulation. This, mimics the real scenario when some cabs get busy picking up passengers from the street all the time and receive few or no calls.

5.1 The simulation of the existing system

To ensure that the developed dispatch simulation uses the exact dispatch policy used by the existing system, we have conducted two experiments using the two different combinations of strategies of moving vacant cabs as described in Section 2.1.

Table 4 illustrates the obtained accumulated results after the last run along with the statistics of the real taxi dispatch system.

The two expirements use the following parameters:

- Total number of cabs: **1622**
- Total number of calls: **43348 (Actual Data)**
- Total number of runs: **4504**
- Periodic statistics at each: **300 runs**

The results of the simulation using the strict strategy (Combination 1) are unsatisfactory when compared to the real data. This is expected since in reality vacant taxis attempt to move looking for potential passengers unlike the movement in the simulation using the strict strategy (Combination 1), which allows the return of vacant cabs to the main city only. Thus, the averaged dispatch time and the total waiting time increases significantly. As a result, the cancellation rate increases and the success rate drops. Note that the log that we used contains only successful calls. Therefore, the real system has success rate of 100% for the data we have used.

When the flexible strategy (Combination 2) is used to move vacant cabs in the simulation, the results improve significantly. The number of dispatched calls increases to reach 82.2%. Consequently, the cancellation rate declines sharply to reach 17% of the total calls. Also, the number of waiting calls in the queue at the last run has been reduced. In addition, the fleet utilization is enhanced as a result of applying the flexible strategy (Combination 2) in the simulation.

The average dispatch time and the average total waiting time drop by around 6 minutes, but they are still higher than the real system (by 15 minutes approximately for the dispatch time and 9 minutes for the total waiting time). While surprising at first, it is actually justified because the flexible strategy (Combination 2) of movement of vacant cabs toward congested areas does not take place frequently and is triggered only when the number of waiting calls is above a threshold so the travel time of the vacant cabs affects the dispatch time of waiting calls. Although we tried to simulate the behavior of the involved parties in the dispatch process, there are still other human factors that are difficult to simulate and predict. In general, the pick up time in the simulation is lower than the existing one. This might be because the simulation is restricted to assign calls to vacant cabs within the main area or the adjacent areas of the main area of the call. While in the real system, officials may force-assign calls to vacant cabs in other areas which increases the travel time of the assigned cab to reach the customer. We will use the simulation with the flexible strategy (Combination 2) to simulate the existing system in the subsequent sections because it captures (to an extent) the human behavior. The following section compares our proposed self-organizing approach to two benchmarks: the simulated existing system, which provides a fair comparison in terms of modeling human behavior, and the existing system itself, which provides a rough estimate of how our proposed architecture would perform in practice. It should be noted that at the end our results here are based on simulation. While the results as we show are quite promising, a fair comparison to the existing system is only possible through incorporating our approach into the real live system.

Rule	ADT	APT	$ATWT$	CR	SR	WR	FU
Real	5.45 ±9.8	14.5 ±17.2	20 ±20.3	0	100	N/A	N/A
Comb. 1	26.45 ±25.8	9 ±6.58	35.4 ±26.3	47.9	50.8	1.26	13.55 ±13.2
Comb. 2	20.57 ±21	8.5 ±5.79	29 ±22	17	82.2	0.7	21.9 ±8.47

Table 4: Simulation of existing system with different strategies of vacant cab movements

5.2 The simulation with self-organization

We have tried different values of of self-organization probability p_o = (0.001, 0.1, 0.4, 0.8). The obtained results are compared to the existing system and the simulating of the existing system. We calculate the averaged degree distribution ADD of the network of areas to verify the stability (convergence) of our approach. The degree of a node in a network is the total number of connections it has to other nodes and the degree distribution is the total distribution of degrees in the whole network [7].

The following experiments use the flexible strategy (Combination 2) to move vacant cabs. Table 5 illustrates the obtained results. The table also contains the statistics of the existing system for reference given the first two rows here are basically row 1 and 3 in Table 4.

P_o	ADT	APT	$ATWT$	CR	SR	WR	FU	ADD
Real	5.45 ±9.8	14.5 ±17.2	20 ±20.3	0	100	N/A	N/A	1.33 ±1.06
0	20.57 ±21	8.5 ±5.79	29 ±22	17	82.2	0.7	21.9 ±8.4	1.33 ±1.06
0.001	20.22 ±20.7	8.5 ±5.7	28.73 ±21.5	14.9	84.3	0.6	22.49 ±8.4	1.44 ±1.12
0.1	18.56 ±20.7	8.76 ±6.1	27.33 ±21.7	17.9	81.2	0.7	21.68 ±8	1.8 ±1.33
0.4	18.47 ±20.4	8.86 ±6.4	27.33 ±21.3	14.8	84.3	0.7	22.51 ±8.6	1.91 ±1.37
0.8	19.57 ±20.7	8.72 ±6.1	28.29 ±21.8	15.5	83.7	0.7	22.32 ±8.4	1.87 ±1.28

Table 5: Accumulated statistics with different probabilities of self-organization using the flexible strategy (Combination2)

Overall, the self-organization is not effective while using the flexible strategy (Combination 2) for moving vacant cabs. There is no enhancement in the results when self-organization is incorporated in the simulation. The average degree distribution has still increased which means the self-organization is taking place but there is no noticeable improvement. This was surprising at first, because one would expect self-organization to improve performance over a fixed organization even by a small margin. Upon careful inspection, we have found that the flexible strategy (Combination 2) causes excessive movement of vacant cabs, which in turn reduces the frequency of assigning calls to vacant cabs in adjacent areas. Although self-organization process is taking place and adding new neighbors, the process has no effect because areas do not utilize their neighbors. This observation has lead us to step back and try the strict strategy (Combination 1) of moving vacant cabs in the following experiments.

5.2.1 *The strict strategy (Combination 1) for Movement of vacant cabs*

Table 6 compares our self-organizing approach using the strict movement strategy (and under different values of p_o) to of the existing system.

P$_o$	*ADT*	*APT*	*ATWT*	*CR*	*SR*	*WR*	*FU*	*ADD*
Real	5.45 ±9.8	14.5 ±17.2	20 ±20.3	0	100	N/A	N/A	1.33 ±1.06
0	20.5 ±21	8.5 ±5.7	29 ±22	17	82.2	0.7	21.9 ±8.4	1.33 ±1.06
0.001	11.5 ±17.9	11.25 ±9.1	22.82 ±19.5	8.2	91.5	0.16	24.38 ±16.9	1.54 ±1.19
0.1	2.8 ±7.8	12.53 ±10.9	15.39 ±13.5	0.6	99.1	0.16	26.41 ±15	1.69 ±1.24
0.4	2.9 ±8.1	12.23 ±10.5	15.16 ±13.2	0.6	99.1	0.13	26.42 ±14.4	1.7 ±1.31
0.8	2.7 ±8.1	12.3 ±11.1	15.14 ±13.8	0.6	99.2	0.07	26.45 ±15.1	1.7 ±1.31

Table 6: Accumulated statistics with different probabilities of self-organization using the strict strategy (Combination1)

In comparison with the simulation without self-organization, when $p_o = 0.001$ the average dispatch time is reduced by around 9 minutes which causes a decline in the total waiting time by 8 minutes on average. However, the pick up time rises slightly because the self-organization technique doesn't take into consideration the duration of among areas when adding new neighbors which yields in higher travel time of taxis to pick up customers. With $p_o = 0.001$, the results are still not better than the real system because using a relatively low probability of self-organization, delays the results to converge. Moreover, the averaged degree distribution increases slightly by 0.2. On the other hand, when $p_o = 0.1$ or greater, the system's performance significantly improves across all measures in comparison with both the real system data and the simulation of the real system without organization. Increasing p_o beyond 0.1 does not improve performance tangibly, suggesting the convergence of our self-organizing mechanism. Figure 1 plots the averaged dispatch time that is computed every 300 time steps (=300 minutes) for each simulation run to measure performance improvement of self-organization. Initially, the averaged dispatch time starts low then it starts to raise. The reason is that all cabs are vacant at the beginning then, they get busy serving calls which causes the gradual increase of the dispatch time. Without self organization, the averaged dispatch time increases over time. When p_o of 0.001 is used, the averaged dispatch time reduces gradually after a peak at time 900 to reach the minimum at the last run but results take longer time to converge since self-organization doesn't take place frequently. While for p_o of 0.1 or greater, the averaged dispatch time starts to decline after a slight increase at the beginning. Then, it increases again a bit with some fluctuations due to changes in the availability of cabs and the need to self-organize the network to tackle for new needs.

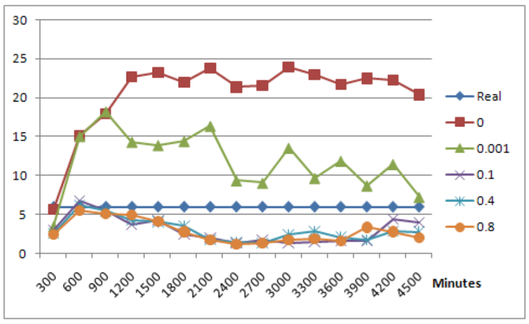

Figure 1: Average Dispatch Time for different values of p_o plus the existing system

Figure 2 plots the average pick up time computed for each 300 time steps of each simulation run. When self-organization has been incorporated in the simulation, the pick up time increases slightly. When $p_o = 0.001$, the increase is slower in comparison with higher values of p_o since the self-organization does not take place frequently. Note that $p_o = 0$ has the minimum pickup time because it only assigns cabs to local available cabs or immediately adjacent areas.

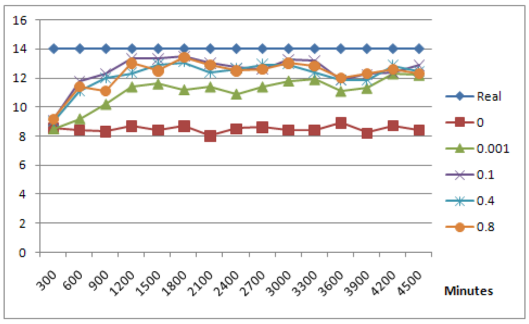

Figure 2: Average Pick up Time for different values of p_o plus the existing system

The average total waiting time computed for each 300 time step (Figure 3) is very similar to Figure 1 since it is dependent on the dispatch time.

One might wonder if the system achieves better performance by simply overconnecting the areas. To investigate this further, we plotted the average *degree distribution* and standard deviation for each 300 time step for the simulation using 0.1 p_o of self-organization in Figure 4 to show the changes in the degree distribution. We can see that the degree distribution remains relatively stable, which suggests that the system is not overconnecting areas, but rather improving the quality of the adjacency schedule.

Finally, Figure 5 plots the number of agents which have a number of neighbors: 0, 1, 2, 3, 4, 5 respectively at both the first time step and the last one for the simulation using p_o of 0.1. The overall degree distribution has not changed

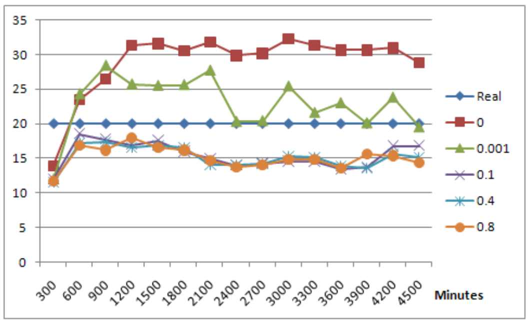

Figure 3: Average Total Waiting Time for different values of p_o plus the existing system's one

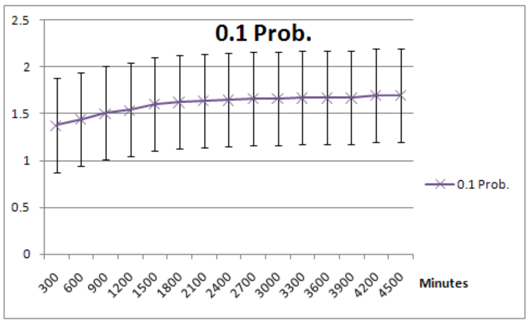

Figure 4: Average Degree Distribution for p_o 0.1

significantly. There is a small drop in isolated areas (with 0 neighbors) at the expense of slight increase in areas with 3,4, and 5 neighbors.

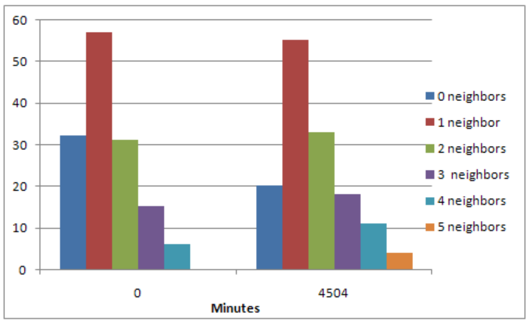

Figure 5: Number of agents having 0,1,2,3,4,5 neigbors for p_o 0.1

6. RELATED WORK

The related works to this work focus on one of the following methods to solve the taxi dispatch problem: (1) assignment method (2) Multiagent Systems. This section briefly describes the used methods and eventually addresses the difference of our work in comparison with the related work and how it overcomes the limitations. Also, the section presents some works that addressed the freight transportation problem in general.

6.1 Assignment Optimisation Approaches

Most approaches that have been presented on the problem of establishing a flexible taxi dispatch system focused on the assignment method as the assigned taxi should reach the customer in the shortest time possible. For example, Chung [2] proposed to use the A* search algorithm to calculate the shortest path by adjusting it to accommodate the road network's conditions. Also, Der-Horng et al [5] proposed a dispatch system whereby the dispatch of taxis is determined by real-time traffic instead of just using the shortest line path to the customer. Another approach proposed by Shrivastra et al [11] uses a fuzzy rule combination approach to solve taxi dispatch problems. Two contradicting rules are satisfied by the proposed approach simultaneously which are "nearest vehicle first"and "the least utilized vehicle first". More importantly, these approaches require the availability of supplementary data such as road structures, or real time traffic information which might be unavailable or up-to-date. Another aspect that is not covered by the mentioned solutions is how to enhance the structure of the taxi network and manage it automatically.

6.2 Multi-Agent Approaches

Some other researches use Multiagent communication to dispatch taxis to customers. Seow et al [10] proposed a multiagent architecture, whereby taxis are represented by collaborative agents that in turn negotiate on behalf of taxi drivers for incoming calls. The proposed solution aims at increasing customer satisfaction as well as driver satisfaction. In comparison with an existing systems in Singapore, the proposed system reduced passenger pick-up time and empty taxi cruise time. The proposed architecture shares the limitation of the solutions mentioned in Section 6.1 in terms of its data requirements (e.g. real time GPS locations of taxis). Moreover, unlike our work, it focuses on the assignment method only rather than the structure of the network and it is based on de-centralized networks.

Another interesting approach to the taxi dispatch problem is highlighted by de Weerdt et al [4]. A resource-based planning framework is presented where agents can merge plans by exchanging their resources. They proposed a plan-merging algorithm that is evaluated on a taxi domain. Empirical results showed a decrease of the total distance driven by all taxis if passengers are allowed to share rides. Of course, this approach requires customers to be willing to share rides, and hence has a different focus to our work.

Some researchers have realized that enhancing assignment methods alone does not necessarily make dispatch systems efficient. This is because, the way the network of taxis and tasks is structured affects the assignment method as well. The better the network is organized and structured, the less time is needed for call assignment, passenger pick up time and average travel distance. Sander et al [9] presented a novel algorithm for efficient task allocation that uses computational geometry techniques to determine adjacency information for the agents. The adjacency information helps agents to determine which neighboring nodes should be considered in the decision making process as they look for a nearby task to accomplish. Neverthless, it is stated in the work that the algorithm doesn't consider some costs associated with the agents such as distance traveled which is directly relevant to the customer satistfaction. The main goal of the solution is to maximize the number of dispatched jobs.

Freight transportation is a slightly different problem from taxi dispatch. This is mainly because freight transportation can allow multiple shipments to be bundled together to optimize the utilization of container space. This is equivalent to taxi sharing among passengers, which is not yet practical. Davidsson et al [3] surveyed some agent-based approaches to freight transportation and traffic management and prsented a framework for assesing such approaches. Other studies addressed the problem of global transportation scheduling such as the work of Perugini et al [8] which requires many transport organization to co-operate and coordinate to transport partial requests along partial routes to accomplish transport requests. Another related area is the optimization of flexible multi-vehicle pick-up and delivery problems within defined time frame, such as the work of Dorer and Calisti [6].

7. CONCLUSION AND FUTURE WORK

The paper proposes a multiagent self-organization technique to a taxi dispatch system. The effeciency of the technique has been demonstrated by simulating an existing taxi dispatch system and incorporating the self-organization technique. The self-organization technique allows agent to add or remove other agents from their neighborhood where agents are dispatch areas, tasks are calls and resources that fulfill tasks are taxis. Expiremental results show that the proposed self-organization technique decreased the total waiting time by up to 25% in comparison with the real system and increased taxi utilization by 20% in comparison with results of the simulation without self-organization.

We are currently working on (gradually) implementing our proposed approach in the real live taxi dispatch system. We are also investigating the use of multiagent reinforcement learning in combination with our self-organizing technique, following the work of Abdallah and Lesser [1] to decrease the total waiting time further. Another interesting point is investigating driver fairness, so that drivers roughly receive the same number of calls.

8. REFERENCES

[1] S. Abdallah and V. Lesser. Multiagent reinforcement learning and self-organization in a network of agents. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 172–179, 2007.

[2] L. C. Chung. GPS taxi dispatch system based on A* shortest path algorithm. Master's thesis, Submitted to the Department of Transportation and Logistics at Malausia University of Science and Technology (MUST) in partial fulfillment of the requirements for the degree of Master of Science in Transportation and Logistics, 2005.

[3] P. Davidsson, L. Henesey, L. Ramstedt, J. Törnquist, and F. Wenstedt. Agent-based approaches to transport logistics. In F. Klügl, A. Bazzan, and S. Ossowski, editors, *Applications of Agent Technology in Traffic and Transportation Book*, Whitestein Series in Software Agent Technologies. Birkhäuser Basel, Basel, Switzerland, 2005.

[4] M. M. de Weerdt, R. P. van der Krogt, and C. Witteveen. Resource based multi agent plan merging: framework and application. In *Proceedings of the 22nd Annual Workshop of the UK Planning and Scheduling Special Interest Group (PlanSig)*, pages 218–233, 2003.

[5] L. Der-Horng, H. Wang, R. L. Cheu, and S. H. Teo. A taxi dispatch system based on current demands and real-time traffic conditions. In *Transportation Research Record*, pages 193–200, 2004.

[6] K. Dorer and M. Calisti. An adaptive approach to dynamic transport optimization. In F. Klügl, A. Bazzan, and S. Ossowski, editors, *Applications of Agent Technology in Traffic and Transportation Book*, Whitestein Series in Software Agent Technologies, pages 33–49. Birkhäuser Basel, Basel, Switzerland, 2005.

[7] S. N. Dorogovtsev and J. F. F. Mendes. Evolution of networks. In *Advances in Physics*, pages 1079–1187, 2002.

[8] D. Perugini, D. Lambert, L. Sterling, and A. Pearce. Provisional agreement protocol for global transportation scheduling. In *Workshop on agents in traffic and transportation held in conjunction with the International Conference on Autonomous Agents and Multi Agent Systems (AAMAS04)*. 2004.

[9] P. V. Sander, D. Peleshchuk, and B. J. Grosz. A scalable, distributed algorithm for efficient task allocation. In *AAMAS 02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1191–1198, 2002.

[10] K. T. Seow, N. H. Dang, and D.-H. Lee. Towards an automated multiagent taxi-dispatch system. In *Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on*, pages 1045–1050, 2007.

[11] M. Shrivastra, P. K. Chande, A. S. Monga, and H. Kashiwagi. Taxi dispatch: A fuzzy rule approach. In *IEEE Conference on Intelligent Transportation Systems*, pages 978–982, 1997.