Minimizing waiting time in last mile delivery for smart city

Lindawati
SAP Innovation Center Singapore, Singapore
lindawati@sap.com
Aldy Gunawan
School of Information Systems, Singapore Management University, Singapore
aldygunawan@smu.edu.sg

Abstract
Customers consider a reliable and on-time last mile delivery in Omni-channel business as important as price and good's quality. We focus on minimizing the waiting time for customers and carriers in last mile delivery. We model the problem as an optimization problem and develop Genetic Algorithm to solve it.

Keywords: last mile delivery, waiting time, genetic algorithm

INTRODUCTION

The Omni-channel business model, such as e-commerce and offline-to-online commerce (Du and Tang, 2014), creates significant last mile deliveries. For courier companies engaged by the Omni-channel companies, it is not only increasing their revenues but also their challenges (The Logistics Institute Asia Pacific, 2013). These last mile deliveries become higher in volumes but smaller in sizes of packages. It needs to be picked, packed, placed and delivered all the way to the customers' home (DHL Trend Research, 2014; Mitrovic-Minic et al. 2004).

The challenges become more complicated when customers request for a narrow delivery time that can be as narrow as 30 minutes (Himelstein, 1999; Punakivi et al., 2001). The customers consider this as important as price and quality of products (Laudon and Traver, 2007; Li and Lee, 1994). These deliveries can be very critical steps in Omni-channel business model especially in smart city environment like Singapore with its Smart Nation vision (Infocomm Development Authority of Singapore, 2014) where most of the information including delivery information can be accessed from many channels through a Smart Nation Platform. The ability to fulfil these deliveries on time could determine the success of the Omni-channel companies (Lee and Hwang, 2001).

To fulfill customers' narrow delivery time-windows, some courier companies increase their fleet. Each fleet would cover less number of deliveries. It can be done by physically buying new vehicles or by leveraging on crowd-sourcing driver using *uber-like* model (Uber, 2015; Russell, 2015; Lawler, 2015). Having less number of deliveries, the fleet drivers may require to spend substantial time to wait from one delivery to another as illustrated in Figure 1. They wait for the time windows in the previous or current customer locations or even in the road sides. These would ultimately increases the last-mile delivery cost that already very expensive and estimated to between 13% and 75% of the total logistics costs in some industry (Gevaers et al., 2011). It is true

for courier companies that own their own fleet or using *uber-like* model. In addition to the cost, these waiting may also give negative impact to the courier companies. For example, if the fleet drivers need to wait in the loading docks/parking lots, the loading dock managements may fail complaints to the courier companies for staying too long and prevent other vehicles to come in.

Figure 1 - illustration of waiting time

In this paper, we propose a mathematical model that attempts to minimize the total waiting time for the last mile delivery problem with a minimum number of vehicles. The narrow time-windows and same-day delivery requests are considered as the hard constraints. We solve the problem in two steps: (1) calculating the minimum number of vehicles using greedy algorithm and (2) finding a route that minimizes the fleet total waiting time using Genetic Algorithm.

Experiments are run on generated delivery data extracted from real demand data in Singapore. Since the demand data do not have time-windows, we generate very narrow time-windows between 10 to 30 minutes to test our proposed approach. It is shown that our proposed approach is able to get optimum solutions for small instances. For large instances, because the exact algorithm cannot produce optimum solutions, we compare our result with manual assignment based on delivery location area. It is shown that the average decrement of waiting time up to 74% in average.

The remaining of the paper is organized as follows: Section 2 describes the problem formulation including assumptions and notations used in the mathematical model. Section 3 presents the proposed approach and section 4 describes the computational experiments and finally, Section 6 summarizes the conclusions and future research direction.

PROBLEM FORMULATION

In this paper, we assume that the number of fleet is sufficient enough and it consists of homogeneous vehicles with the same volume and weight capacity. Let N be the set of delivery requests. Each delivery $i \in N$ is associated with delivery load size q_i and destination v_i.

Let $V = P \cup \{v_0\}$ be the set of nodes where $P = \{ v_i \in V \mid i = 1,2, \dots, |N|\}$ represents the customer destination locations and node v_0 denotes the depot location where a fleet of vehicles are housed and deliveries are consolidated. M denotes the set of vehicles. Each vehicle has a volume and weight capacity Q and time capacity T. Vehicle trip needs to start from the depot v_0 with a set of deliveries $V_i \subset P$ and end in the depot v_0 with no cargo/delivery to ensure the same-day delivery policy. For all $i, j \in V$, d_{ij} and t_{ij} represent a non-negative travel distance and a non-negative travel time between i and j, respectively.

$[e_i, l_i]$ denotes the time window from customer where the delivery service at location i must take place and $0 < l_i - e_i \leq \varepsilon$. ε is a small number (i.e. 10 to 30). If a vehicle k reaches location i before e_i, it needs to wait w_i time units until e_i to deliver the goods. If a vehicle k reaches location i after l_i, a delivery cannot be completed. A delivery route R_k for vehicle k is a directed route for a set of deliveries $V_i \subset P$ such that:

1. It starts and ends in v_0
2. Vehicle k visits each location i exactly once
3. The vehicle load at any one time never exceeds Q
4. The arrival time A_i and departure time D_i of any location i satisfy $D_i \in [e_i, l_i]$ where $D_i = \max\{A_i, e_i\}$ and $w_i = \begin{cases} e_i - A_i, & A_i < e_i \\ 0, & A_i \geq e_i \end{cases}$

Using these notations, we define the narrow time-windows delivery problem as minimizing w_i such that:

$$\forall\, i \in N, \sum_{k=1}^{M} Z_{ik} = 1 \tag{1}$$

$$\forall\, i \in V, \sum_{k=1}^{M} \sum_{j=1}^{V} x_{ijk} = 1 \tag{2}$$

$$\forall\, k \in M, \sum_{i=1}^{V} x_{i0k} = 1 \tag{3}$$

$$\forall\, k \in M, \sum_{j=1}^{V} x_{0jk} = 1 \tag{4}$$

$$(\forall\, h \in V)(\forall\, k \in M), \sum_{i=1}^{V} x_{ihk} - \sum_{j=1}^{V} x_{hjk} = 0 \tag{5}$$

$$(\forall\, j \in V)(\forall\, k \in M), \sum_{i=1}^{V} x_{ijk}\, Q \geq y_j \tag{6}$$

$$(\forall\, i, j \in V)(\forall\, k \in M), x_{ijk} = 1 \Rightarrow y_i + q_i = y_j \tag{7}$$

$$y_0 = 0 \tag{8}$$

$$(\forall\, i \in V), y_i \geq 0 \tag{9}$$

$$(\forall\, h \in V)(\forall\, k \in M), x_{ijk} = 1 \Rightarrow D_i \leq l_i \tag{10}$$

Constraint (1) ensures that each delivery request is assigned to exactly one vehicle. Constraint (2) ensures that each job is visited exactly one. Constraints (3) and (4) ensure that each vehicle departs from and return to the depot. Constraints (5) ensures that if a vehicle arrives at a node then it must also depart from that node. Constraints (6) to (9) refer to the capacity constraints. Constraint (10) ensures a delivery has to be completed within its time window.

PROPOSED APPROACH

From our discussion with two courier companies in Singapore, it is revealed that for they need to deliver more than 3,000 parcels each day to different customer's locations. With these large problem sizes, it would be difficult to find and prove the existence of an optimal solution using exact algorithm, especially within a short computation time. It would be necessary to develop a heuristic approach to achieve higher efficiency and better service levels performance for home delivery service.

We propose heuristics algorithms, greedy algorithm and Genetic Algorithm (GA). Greedy algorithm is a heuristic algorithm that always make locally optimal choice in hope to arrive at a global optimal (Cormen et al., 2001). We propose greedy algorithm to find minimum number of vehicles to fulfil the last mile deliveries without violating the customers' narrow time windows. While GA is a meta-heuristics algorithms that moves from one population of chromosomes to another population by genetics-inspired operators of selection, crossover and mutation (Mitchell, 1999). GA is used to search optimal delivery routes for vehicles to minimize their waiting time by considering customer's narrow time windows as the hard constraints. Other constraints such soft fleet capacities and operation hours are also build-into the developed models.

Greedy Algorithm to determine the minimum number of vehicles

In order to find the number of vehicles, we first order the N delivery requests based on their time windows $[e_i, l_i]$ as N_{ord}. We start with M set of vehicle where number of vehicle $|M|$=1. For each delivery $i \in N_{ord}$, we assign it to an available vehicle with minimum waiting time. The vehicle should be able to fulfil the delivery request without violating the time windows $[e_i, l_i]$. If there is no available vehicle to fulfil the delivery, we increase the number of vehicle by one unit (e.g. $|M| + 1$). It is repeated until each delivery is assigned. After each delivery $i \in N_{ord}$ is assigned, we return $|M|$ as the minimum number of vehicles needed.

Genetic Algorithm to find the optimum route

In our GA, each chromosome in the population pool is transformed into fleet routes. The evolutionary part (such as selection) is conducted as in conventional GAs using a problem-specific crossover and mutation to guarantee feasibility for the generated solutions. The best found solution across all population is then reported.

As in (Ombuki et al., 2006), a chromosome represents the fleet route is given by an integer string of length $|N|$, where N is the set of delivery requests as illustrated in Figure 2. A gene in a given chromosome is the node number assigned to a customer and the sequence of genes in the chromosome is the order of visitation of customers for all the fleet vehicles. No delimiter is used to indicate the beginning or end of a route for individual vehicle but each gene would record the vehicle number to serve it.

Figure 2 - GA Chromosome Representation

The fitness function for each chromosome is evaluated in term of its objective function value for the feasible solution and objective function value plus penalize value for the infeasible solution. Penalize value is set as fixed large number. Hence, the optimal solution is a feasible with minimum fitness function.

An outline of the GA procedure is summarized in Figure 3. It starts by generating an initial population of chromosomes and evolving over an iteration process to find a better solution involves four important elements: initial population generation, selection, crossover and mutation.

Procedure Genetic Algorithm:
Inputs: *i*: problem instance

Method:
1. Generate initial population for population *t*
2. Evaluate feasibility and fitness function for each chromosome in population *t*
3. Repeat until a stopping criteria is satisfied
   3.1. Select parents from population *t* using a designated selection criteria
   3.2. Perform cross-over to generate new offspring for population *t+1*
   3.3. Perform mutation on new offspring in population *t+1*
   3.4. Evaluate feasibility and fitness function for each chromosome in population *t+1*
   3.5. Replace population *t* with population *t+1*

Output: *s*: best solution

Figure 3 - Genetic Algorithm procedure

To generate the initial population, we craft two methods: greedy and random. The methods are summarized in Figure 4 and 5, respectively. The greedy method is similar to the greedy algorithm to find the number of vehicles. The differences is in the assignment of the delivery order. We use the greedy method to generate as many feasible solutions as possible. While the random method is used to maintain the randomness in the solutions. We implement each method to generate half of the initial populations.

In selection, chromosomes in a population are selected as parents to reproduce new offspring. The chromosomes with good fitness function are often being selected than poor ones. For the selection criteria, we use the tournament 2 selection where it chooses two parents randomly and compares their fitness value, and the one with the higher fitness value gets to mate. It is done twice to get two parents.

In reproduction stage, the new offspring is created by selected parents using a problem specific crossover mechanism. We design a specific crossover strategy that can exchange subparts of the chromosome from two selected parents by mimicking biological recombination between two single-chromosome organisms, while in the same time preserving the feasibility of the offspring.

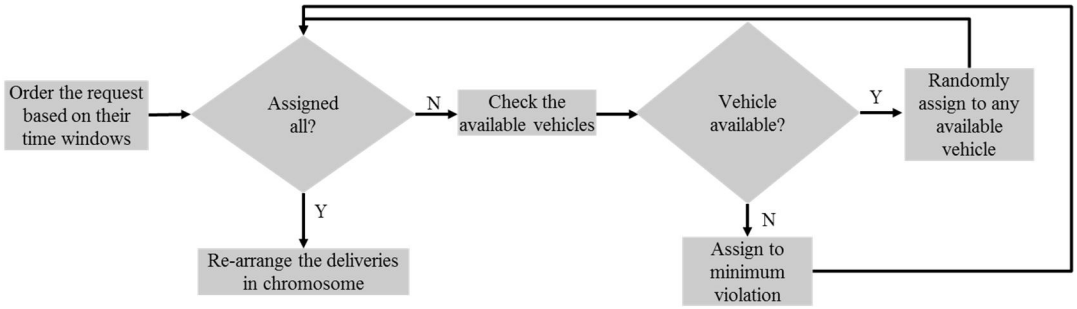

Figure 4 - Initial Population Greedy Generation Method

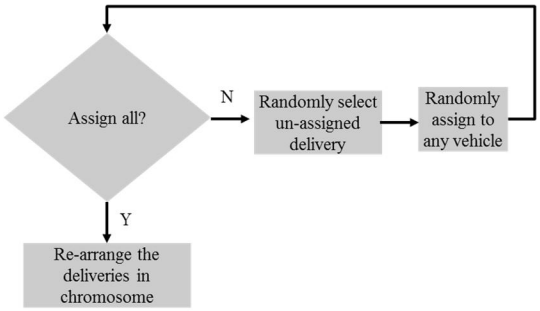

Figure 5 - Initial Population Random Generation Method

The crossover strategy works by first selecting a random crossover point which in our GA is based on vehicle number. We assign all customers before the crossover point in Parent 1 to the new offspring and then the customers after the crossover point in Parent 2 and have not been assigned are inserted to the new offspring. Since we need to fulfill all the delivery demands, the next step is to find the best possible location for the unassigned customers and randomly insert it in the best feasible location. We may need to re-arrange the customer order to ensure the feasibility. The crossover strategy is summarized in Figure 6.

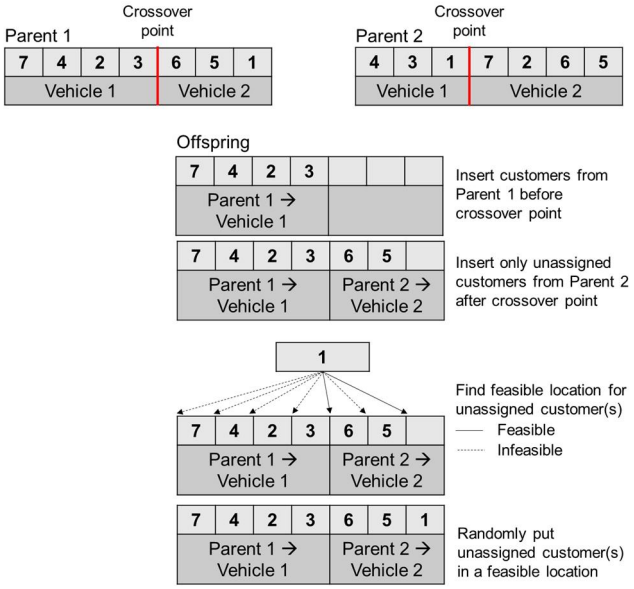

Figure 6 - Crossover Strategy

To diverse the offspring from their parent population and maintain feasibility of the offspring, we apply a mutation mechanism with a given probability. It swap two customers that have similar time windows as illustrated in Figure 7. The mutation mechanism works by first randomly select one customer from the chromosome. We then search another customer with similar time windows. If there are more than one customer with similar time windows, we randomly choose one customer to be swapped. After swapping, we re-arrange the chromosome accordingly.

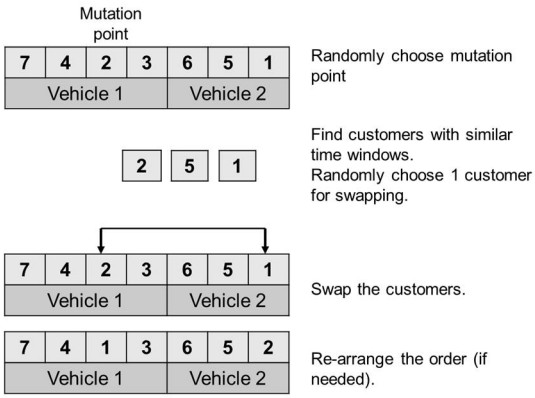

Figure 7 - Mutation Strategy

COMPUTATIONAL EXPERIMENTS

This section describes experiments carried out to investigate the performance of our proposed approach. We randomly generate 10 problem instances with number of customers range from 10 to 100 extracted from real delivery demand data of a courier company in Singapore. We convert the distance between two customers' location to minutes using point-to-point routing solution available online (Google Maps, 2015). We assume that the distance between two customers is the same in each opposite direction and each customers can be visited from all other customers, forming an undirected complete graph. Since most of the real delivery demand data do not have time-windows, we purposely generate very narrow time-windows between 10 to 30 minutes to test the performance of our approach.

In step one, we calculate the number of vehicles to fulfill the time windows for all delivery demands using greedy algorithm. We assume there is unlimited number of vehicles. The greedy algorithm would return minimum number of vehicles required to fulfill all the delivery requests.

In step two, we use the GA algorithm that have three important parameters to determine its behavior, namely: populationSize, crossOverPoint and mutationRate. We conduct a small experiment to evaluate 10 random parameter values and choose the best one as the value for each parameter as described in Table 1. We set the number of generation to 500 and run the GA for 100 iteration. We record the best found solution for each iteration.

Table 1 – GA Parameters

Parameter	Description	Value
populationSize	Population size	500
crossOverPoint	Probability of cross over point	0.7
mutationRate	Probability to run mutation for new offspring	0.03

To measure the performance of our proposed approach, we compare the result with the optimum solutions generated from the exact algorithm. The exact algorithm is used only to generate the optimum route from a given delivery order set N and number of vehicles $|M|$. The number of vehicles $|M|$ is taken from the step 1 of our proposed approach. We limit the run time of the exact algorithm to 2-hours. We observe that the exact algorithm can only produce the optimum solution for two instances which considered as small instances. In our experiment, the small instances are problem instances with 10 and 20 customers.

We report the average best found solution from each iteration and the comparison with the optimum solutions in Table 2. The result shows that our proposed GA is able to produce the optimum solutions within a short time period. It is less than 10% and 5% from the time need for exact algorithm for instance with 10 and 20 customers respectively.

Table 2 – GA Comparison Result with Exact Algorithm

Number of Customers	Step One - Number of Vehicles	Exact Algorithm		Genetic Algorithm	
		Objective Value Found	Time (second)	Objective Value Found	Time (second)
10	2	0	300	0	25
20	4	32	1,250	32	52
30	7	-	-	239	72
40	7	-	-	34	94
50	10	-	-	152	125
60	11	-	-	151	144
70	11	-	-	189	178
80	15	-	-	851	205
90	14	-	-	388	225
100	17	-	-	440	261

Table 3 – GA Comparison Result with Manual Assignment

Number of Customers	Step One - Number of Vehicles	Manual Assignment	Genetic Algorithm	
		Waiting Time	Objective Value Found	Improvement (%)
10	2	85*	0	100.00
20	4	238*	32	86.55
30	7	690*	239	65.36
40	7	475*	34	92.84
50	10	649*	152	76.58
60	11	789*	151	80.86
70	11	684*	189	72.37
80	15	1,393*	851	38.91
90	14	1,033*	388	62.44
100	17	1,236*	440	64.40

* = There are one or more delivery that violates the time windows.

To measure the performance of our proposed approach in large instances, we compare our result with manual assignment. In manual assignment, the delivery requests are grouped by its delivery location. The groups are statically generated into several areas. Each area is calculated by

dividing the overall area coverage of the deliveries by number of vehicles $|M|$. It is not based on number of delivery requests in each area. We report the average best found solution for each iteration and its improvement from manual assignment in Table 3. The result shows that our proposed approach is able to provide a better solution compare to the manual assignment in two aspects: time windows violation and average waiting time. In term of time windows violation, our approach eliminates the time windows violation by providing on-time delivery for each delivery request. While in average waiting time, our approach decreases the average wait time by more than 74% on average.

CONCLUSIONS

In this paper, we focus our study on the urban last mile delivery for Omni-channel business (such as e-commerce and offline-to-online commerce). Customers may have their own narrow preferred time windows. In order to fulfill these preferences, courier companies need to spend a substantial idle time to wait for the correct delivery time windows which increase its transportation cost.

We develop a mathematical model with the objective of minimizing the total waiting time for last mile delivery with a minimum number of vehicles. It is implemented for home deliveries with narrow time windows. Due to the difficulty to solve large instances, we propose an approach using two heuristic algorithms, greedy algorithm and Genetic Algorithm (GA), to tackle the problem. Greedy algorithm is used to calculate the minimum number of vehicles needed to fulfil the delivery requests while GA is used to generate route that minimize the fleet total waiting time. Our proposed approach is able to improve the quality of the solution by decreasing the total waiting time up to 74% on average compared to the manual assignment.

Nonetheless, we see two limitations that we would like to study in near future. We would like to test the scalability of our approach with the real delivery demand data. The second limitation is related to the proposed GA elements. Currently, we craft problem specific elements to ensure the feasibility of the solutions. We may want to explore and try other well-known mechanism for GA such as uniform crossover (Whitley, 1994) to further improve the performance of the GA algorithm.

ACKNOWLEDGEMENT

This research is partially funded by the Economic Development Board and the National Research Foundation of Singapore.

Bibliography

Cormen, T.H., C.E. Leiserson, R.L. Rivest, C. Stein. 2009. Introduction to Algorithms. MIT Press, USA.

DHL Trend Research. 2014. Delivering insight today. Creating value tomorrow! *Logistics Trend Radar.* Available at http://www.dhl.com/content/dam/downloads/g0/about_us/logistics_insights/DHL_Logistics-TrendRadar_2014.pdf (accessed date December 29, 2015).

Du, Y., Y. Tang. 2014. Study on the Development of O2O E-commerce Platform of China from the Perspective of Offline Service Quality. *International Journal of Business and Social Science* **5**(4): 308-312.

Edwards, J., A. McKinnon, T. Cherrett, F. McLeod, L. Song. 2009. The impact of failed home deliveries on carbon emissions: Are collection/delivery points environmentally-friendly alternatives? *14th Annual Logistics Research Network Conference, 9th - 11th September 2009, Cardiff.*

Eshelman, L.J., R. Caruana, J.D. Schaffer. 1989. Biases in the crossover landscape. *Proceedings of the 3rd International Conference on Genetic Algorithms,* June 1989, Virginia, USA.

Gevaers, R., E. Van de Voorde, T. Vanelslander. 2011. Characteristics and typology of last-mile logistics from an innovation perspective in an urban context. In Macharis, C., S. Melo, eds. *City distribution and urban freight transport: multiple perspectives*. Edward Elgar Publishing, 56-71.

Google Maps. 2015. [Singapore] [Street map]. Available at https://www.google.com.sg/maps/place/Singapore/@1.3150701,103.7069324,11z/data=!3m1!4b1!4m2!3m1!1s0x31da11238a8b9375:0x887869cf52abf5c4 (accessed date December 29, 2015).

Himelstein, L. 1999. Can you sell groceries like books? *Business Week*, E-Biz, July 26, No. 3639: 26-9.

Infocomm Development Authority of Singapore. 2014. Factsheet Smart Nation Platform. Available at https://www.ida.gov.sg/~/media/Files/About%20Us/Newsroom/Media%20Releases/2014/0617_smartnation/AnnexA_sn.pdf (accessed date January 4, 2016).

Laudon, K. C., C.G. Traver. 2007. *E-commerce*. Pearson/Addison Wesley, New York.

Lawler, R. 2015. Cargomatic Gets $8 Million To Build An Uber For Truckers. Available at http://techcrunch.com/2015/01/29/cargomatic/ (accessed date December 29, 2015).

Lee, H. L., S. Whang. 2001. Winning the last mile of e-commerce. *MIT Sloan Management Review* **42**(4): 54-62.

Li, L., Y.S. Lee. 1994. Pricing and Delivery-Time Performance in a Competitive Environment. *Management Science* **40**(5):633-646.

Mitchell, M. 1999. An Introduction to Genetic Algorithms. MIT Press, USA.

Mitrovic-Minic, S., R. Krishnamurti, G. Laporte. 2004, Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B* **38**:669-685.

Ombuki, B., B.J. Ross, F. Hanshar. 2006. Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence* **24**(1): 17-30.

Punakivi, M., H. Yrjola, J. Holmstrom. 2001. Solving the last mile issue: reception box or delivery box? *International Journal of Physical Distribution & Logistics* **31**(6):427-439.

Russell, J. 2015. 'Uber For Logistics' Startup Lalamove Lands $10M To Expand In China And Southeast Asia. Available at http://techcrunch.com/2015/01/05/lalamove-10-million/ (accessed date December 29, 2015).

The Logistics Institute Asia Pacific. 2013. Synchronized Last-Mile Logistics for Sustainable, Efficient Urban Delivery. *TLI - Asia Pacific White Papers Series*. Available at http://www.tliap.nus.edu.sg/research/Research_resources.aspx (accessed date August 14, 2013).

Uber. 2015. UberCARGO: A Reliable Ride For Your Items. Available at https://newsroom.uber.com/hong-kong/2015/01/a-ride-for-your-goods-introducing-ubercargo/ (accessed date December 29, 2015).

Whitley, D. 1994. A genetic algorithm tutorial. *Statistics and computing* **4**(2): 65-85.