

## DIPLOMA THESIS ASSIGNMENT

**Student:** Bc. Jiří Starý  
**Study programme:** Open Informatics  
**Specialisation:** Artificial Intelligence  
**Title of Diploma Thesis:** Agent-based Model of Parcel Logistics

### Guidelines:

1. Familiarize yourself with parcel logistics and agent-based modeling.
2. Analyze the parcel logistics domain in terms of its constituent agents, their behavior and interactions.
3. Formalize the parcel delivery problem as a constraint optimization problem.
4. Develop agent-based parcel delivery job allocation and vehicle routing algorithms.
5. Design and implement warehouse and vehicle agents capable of allocating and delivering parcels according to 4.
6. Integrate 4) and 5) into an agent-based model of parcel logistics built on the AgentPolis platform.
7. Evaluate the characteristics of the model on a range of test scenarios.

**Bibliography/Sources:** Will be provided by the supervisor.

**Diploma Thesis Supervisor:** Ing. Michal Jakob, Ph.D.

**Valid until:** the end of the summer semester of academic year 2012/2013

  
prof. Ing. Vladimír Mařík, DrSc.  
Head of Department



  
prof. Ing. Pavel Ripka, CSc.  
Dean



## ZADÁNÍ DIPLOMOVÉ PRÁCE

**Student:** Bc. Jiří Starý  
**Studijní program:** Otevřená informatika (magisterský)  
**Obor:** Umělá inteligence  
**Název tématu:** Agentní model logistiky balíkové přepravy

### Pokyny pro vypracování:

1. Seznamte se s logistikou přepravy balíků a agentovým modelováním.
2. Analyzujte logistiku balíkové přepravy a identifikujte zúčastněné agenty, jejich chování a interakce.
3. Formalizujte problém doručování balíků jako problém optimalizace s omezujícími podmínkami.
4. Navrhněte agentní algoritmy pro alokaci zásilek a trasování vozidel.
5. Navrhněte a implementujte skladové a vozidlové agenty schopné alokovat a doručovat zásilky dle algoritmů navržených v 4).
6. Integrujte 4) a 5) do agentního modelu logistiky balíkové přepravy postavené na platformě AgentPolis.
7. Vyhodnoťte charakteristiky modelu na množině testovacích scénářů.

**Seznam odborné literatury:** Dodá vedoucí práce.

**Vedoucí diplomové práce:** Ing. Michal Jakob, Ph.D.

**Platnost zadání:** do konce letního semestru 2012/2013

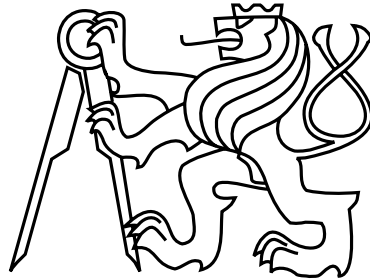
  
prof. Ing. Vladimír Mařík, DrSc.  
vedoucí katedry



prof. Ing. Pavel Ripka, CSc.  
děkan



Czech Technical University in Prague  
Faculty of Electrical Engineering  
Department of Cybernetics



Master's Thesis

## **Agent-based Model of Parcel Logistics**

*Bc. Jiří Starý*

Supervisor: Ing. Michal Jakob, Ph.D.

Study Programme: Open Informatics

Field of Study: Artificial Intelligence

May 10, 2012



## Aknowledgements

I would like to thank my mentor for a great motivation and support. Mr. Jakob, who was relentlessly pushing me forward for the entire time, my family, friends and all the people that supported me, while I was writing this thesis. Without you, most of this would never be possible.






## Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 9. 5. 2012

  
.....  
Podpis autora práce



# Abstract

The purpose of this work is to leverage the agent-based modeling onto the process of parcel delivery. First, some related works are shown, while also describing the benefits of agent-based modeling. A groundwork for the the task is laid, when it is defined formally as a constraint optimization problem. An allocation algorithm based on the definition is formed. The model of the problem is built and described. The implementation is constructed over an event-based agent simulation platform for the urban environment, AgentPolis. Inside, the vehicles move on a graph representation of the streets. Using the A\* planner and with the estimated distance as a cost, they compete for the deliveries. Later, on the city of Prague, several allocation strategies are attempted, while an insight into the interactions between agents is gained and the results discussed. Two variants of the allocation algorithm were measured, each performing better under different conditions and metrics.

# Abstrakt

Tato práce se zabývá využitím agentního modelování při doručování balíků. Nejprve jsou popsány související práce a výhody tohoto přístupu. Základem ke specifikaci této úlohy bylo zvoleno formální definování problému, jako problému s omezujícími podmínkami. Pomocí této definice je vytvořen alokační algoritmus. Vzniká model systému, který je popsán a implementován. Tato implementaci je postavena na událostmi řízené simulační platformě pro modelování městského prostředí, AgentPolis. V ní vozidla využívají grafové reprezentace ulic. V této reprezentaci soutěží o zakázky s využitím plánovače A\* a pomocí jejich ocenění přibližnou vzdáleností. Na mapě Prahy je poté vyzkoušeno několik alokačních strategií, dávajících nahlédnout do fungování systému, a diskutovány výsledky. Dvě varianty alokačního algoritmu byly změřeny, každá vhodnější pro jiné podmínky a metriky.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Outline	2
<b>2</b>	<b>Background and related work</b>	<b>3</b>
2.1	Overview of existing approaches for cargo delivery	3
2.1.1	Area division	3
2.1.2	Assignment and scheduling	4
2.1.3	Meta-heuristics	4
2.2	Agent-based modeling	5
<b>3</b>	<b>Problem formalization</b>	<b>7</b>
3.1	Objective function	8
3.2	Solution representation	9
3.3	Notation	11
3.4	Metrics	11
<b>4</b>	<b>Allocation algorithm</b>	<b>13</b>
4.1	Agent interaction	13
4.2	Evaluation of deliveries by insertion heuristic	14
4.3	Allocation	15
<b>5</b>	<b>Simulation model</b>	<b>17</b>
5.1	Agents	17
5.2	Agent interaction	18
5.3	Agent actions	18
5.4	Pseudo algorithms	18
5.5	Environment model	20
<b>6</b>	<b>Implementation</b>	<b>23</b>
6.1	Platform	23
6.1.1	Visualization	23
6.2	Environment	24
6.2.1	Communication protocol	24
6.2.2	Sensors	24
6.2.3	Dispatcher implementation	25
6.2.4	Vehicle implementation	26

6.2.5	Map	27
6.3	Producing results	27
6.3.1	Launcher	28
<b>7</b>	<b>Experiments</b>	<b>29</b>
7.1	Experiment setting	29
7.1.1	Scenario parameters	29
7.2	Experiment results	30
7.2.1	System throughput measurement	31
7.2.2	Bidding with real distance	32
7.3	Simulation run details	34
7.4	Summary	38
<b>8</b>	<b>Conclusion</b>	<b>41</b>
8.1	Future work	42
<b>A</b>	<b>List of abbreviations</b>	<b>45</b>
<b>B</b>	<b>User manual</b>	<b>47</b>
B.1	Simulation	47
B.1.1	Google Earth	47
B.1.2	Output	48
B.2	Launcher	48
B.2.1	Launcher.config	48
B.2.2	Config.groovy	49
B.2.3	Output	49
<b>C</b>	<b>Contents of the enclosed CD</b>	<b>51</b>

# List of Figures

4.1	Interaction between vehicle and dispatcher . . . . .	14
5.1	Vehicle lifecycle . . . . .	19
5.2	Deliveries diagram . . . . .	20
5.3	Class diagram of interactions . . . . .	21
5.4	Environment diagram . . . . .	21
6.1	Visualization, on the left using Swing, behind in Google Earth . . . . .	24
6.2	Planning delivery path in real time . . . . .	26
7.1	Increasing amount of vehicles in experiment with an insufficient throughput. Simulation: 1h, warehouses :2, capacity: 20, delivery frequency:10s, fully loaded	31
7.2	Increasing amount of vehicles in experiment with an insufficient throughput. Simulation: 1h, warehouses :2, capacity: 20, delivery frequency:10s, fully loaded	32
7.3	Finding optimal amount of vehicles by delivery time for several delivery fre- quencies. Simulation:1 h, warehouses :2, capacity: 20, not fully loaded . . . . .	34
7.4	Correlation between amount of warehouses and distance . . . . .	34
7.5	Correlation between amount of warehouses and time . . . . .	35
7.6	No influence of capacity in a system with low amount of deliveries and unfilled vehicles . . . . .	35
7.7	Significant difference in delivery time with increase of capacity in a system with low amount of deliveries and fully loaded vehicles . . . . .	36
7.8	Mileage saved by the use of vehicles with higher capacity . . . . .	36
7.9	No influence of capacity in a system with high amount of deliveries and un- filled vehicles . . . . .	37
7.10	Influence of capacity in a system with high amount of deliveries and fully loaded vehicles . . . . .	38
7.11	Agent interaction by the number of messages sent . . . . .	39
B.1	Google Earth screenshot . . . . .	48





# List of Tables

7.1	Table comparing results for two different simulation durations . . . . .	33
7.2	Performance with the real distance bidding. . . . .	33
7.3	Performance for different simulation runs. . . . .	37



# Chapter 1

## Introduction

The delivery of goods is inextricably linked to the image of a living, bustling city as a blood running through its veins. From bringing goods to the shops, same day e-shop client delivery to small point-to-point couriers deliveries, it is an overall important aspect of each city which generates a significant portion of the city traffic.

Each delivery agency might utilize a variety of means of transportation, differing by speed, freight capacity and fuel consumption. Ranging from large vans up to single man-powered bicycles, all these means are employed to strike a complex balance among delivery time, service cost and environmental concerns. The desired goals are higher efficiency and better fleet management [13]. In a traditional sense, orders are placed by clients to a centralized office, which then sorts information by hand using teams of skilled operators and passes it on. Courier deliveries aren't often assigned to a single vehicle, more common approach is to notify all available vehicles in the near vicinity including freelance drivers. They then independently decide if accepting the delivery is feasible for them. The key competitive point is the efficiency of the runs. Moreover, scheduling in this field has some additional difficulties. Some amount of fairness has to be established to decrease the number of drivers with low order placement, otherwise their satisfaction with the job would plummet down in short order. Apart from the driver's psychology, other dynamically changing factors include traffic congestions, creation or disappearance of standing orders, vehicle malfunction etc. [8], which makes the whole area notoriously difficult to manage efficiently. In case of express deliveries or supply chain cargo, consignments are heavily bundled and gathered at local departments from where they are distributed to the local area.

In this paper a stress will be put onto deliveries in a large or medium sized cities shipped from a small set of warehouses to different locations, which is common to e-shops for example, or other logistics when goods should be delivered on short notice and planning of deliveries cannot be efficiently set beforehand. It was a great technological boom including GPS, RFID devices and cloud computing that tracks demands, vehicles and parcels in real time, that introduced this type of agile delivery. Dispatching is THE crucial task for the whole courier company and its proper application can result in great improvements in its efficiency and customer comfort, as the same day delivery begins to emerge as the new standard and companies will be hard pressed to decrease the delivery time.

When seeking new ways to tackle urban challenges, an agent-based modeling seems to be the ideal tool for the task at hand. With it, we should be able to build a model of a delivery

system to help us grasp the issue and gain a clearer insight into complex relationships in this field.

## 1.1 Outline

This work on parcel logistics is divided into 8 chapters. You are now reading an Introduction in the first chapter. In the second chapter, existing approaches for a parcel logistics are discussed and benefits of agent-based modeling are laid out. In the third chapter, a formal definition of the problem is set, including a definition of the result with several metrics to measure its quality. The next chapter is dedicated to the structure of an allocation algorithm built on the formal definition. With the existence of an algorithm, it is possible to analyze the model of a delivery environment including agent behaviour in chapter 5. The sixth chapter describes the implementation details of the model, while also mentioning the AgentPolis platform onto which it was built. Experiments with the deliveries on the map of Prague are the contents of the ensuing chapter, comparing several settings of the allocation algorithm, speculating about the ideal vehicle count and other important parameters. In the last chapter, the conclusions are drawn and the possibilities of a future work are drafted.

## Chapter 2

# Background and related work

In this chapter a selection from the numerous approaches commonly taken when solving parcel logistics or similar problems is presented. The existing approaches are divided into 3 sections. First encompasses the possibilities of dividing the concerned area in which deliveries take place, second is more focused on tackling the dynamical part of the problem and the last talks about the use of meta-heuristics. In the end, some benefits of the agent-based modeling are highlighted.

### 2.1 Overview of existing approaches for cargo delivery

#### 2.1.1 Area division

When covering a large area (city) it is not an uncommon approach to divide it into several regions or clusters. The reasonings behind this cut is thus: Couriers move only in designated area and packages can be bundled when going in the same direction. Moreover idle runs while they move empty or from distant location are minimized as the length of trips is shortened. Each area typically has assigned its own fleet of couriers. In the real-life scenarios, the system should react in some way if high transportation demand drains the whole area empty of couriers, or on the other hand all couriers aggregate in too small location and create too hard competition for each other, so some way of predicting future orders can be useful. The key point is how and where the packets should be transferred from one courier to another. Several division proposed here: [\[14\]](#)

#### Hub logistics

Resembling a way the more traditional logistic deliveries operate (express service), the packages are transferred to a central Hub and then moved according to destination coordinates. There may be only one central hub in a high order volumes area surrounded by a several less desired destinations, or a solitary hub for each region. It is argued that the introduction of hubs should increase bundling of packages. However deliveries of cargo might be delayed when waiting for more packages going in same direction and rerouting at each hub.

### Fixed exchange points

A city is divided into a several clusters, depending on the amount of cargo and each one is assigned its own fleet of couriers. The cluster borders contains a fixed points, where the cargo is exchanged between couriers.

#### 2.1.2 Assignment and scheduling

##### Time slicing approach

The time during the day is sliced into a several periods and new orders are not considered, until the new time period is reached. Thus the dynamic element of the problem is removed. However, the price for this decision is a delay in dispatching orders and possible loss of better fleet schedules. The main benefit in the application being the savings in the terms of computational time.

##### Adaptive scheduling approach

Can be used for this type of scheduling in the real time. Considering that a schedule already exist and it is not necessary to reschedule the whole roster, merely to incorporate the changing events as they occur. The assignment of orders to the couriers is delayed and can be re-assigned before the pick-up point is reached, if overall performance is increased.

#### 2.1.3 Meta-heuristics

Because of the complex nature of the problem posed, it is in practice often useful to use some of smart set of meta-heuristics. The dynamic nature of the problem not only makes the finding of the (closely) optimal solutions costly, but their net value measured at the given time changes as new events occur. So, it is a prudent standpoint to expect changes in vehicle routes and allow flexible planning by use of heuristics, allowing sub-optimal paths that are more amenable to change, or by keeping a list of possibilities. When generalized to non-specific routing and delivery problem, the selection of a suitable heuristic depends on the dynamicity of the problem. That being defined [17] as tuple of rate of change and its urgency. The latter being the time between announcement of a change and its manifestation.

- tabu search [12]

is a kind of local search that evades the repetition and local minima by banning the same or similar solutions for a short period of time. The usual application is by branching possible future paths. As adding new orders change only some of them, a significant portion of time is saved.

- variable neighbourhood search [17]

for the problem a set of arbitrary neighbourhood structures is generated, and each is pondered successively - usually by some black-box search procedure - to determine if the solution has improved. The neighbourhoods are traversed successively in order of their closeness until some stop criterion is met.

- double horizon heuristic [16]

manages the problem with respect to the two goals. The short term, which usually means minimizing the distance traveled and corresponds to the solution of the static variant of the problem. The long term goal, which is to maintain routes in a state when change can be easily incorporated in the schedule. This goal usually works with a notion of slack time, that allows insertion of new tasks.

- ant colony

Inspired by the ant pheromone trail, a positive feed-back loop that favours past successful solutions is made. Usually a heuristic is computed for the chosen move and added with a bonus for each time the move was used in the past.

- genetic algorithms [17]

collection of the plans valid at an event horizon is kept and improved solutions are sought using traditional genetic operators.

- cheapest insertion [16]

starting from a valid plan, new delivery locations are inserted to a position that causes minimal cost increase.

## 2.2 Agent-based modeling

The ability to model a complex system is a key to gaining deeper understanding of the problem and its possible solutions.

Agent-based modeling is a process of looking into a system from its constituent parts [10]. By describing the elements on a microscopical scale, higher and more complex patterns emerge. This kind of modeling usually exploits the fact, that the whole is greater than just the sum of its parts and the interactions do matter.

In ABM these smallest particles are called agents. They each act as an independent individual, defined by its own behaviour rules. By means of interaction between each other and the environment, the model of the system surfaces. Using this decentralized model one may change its parameters and observe changes in global system behaviour without a prior knowledge about the global system [7]. This approach is especially useful when dealing with problems that cannot be easily or efficiently described by top-down approach with simplified or averaged number and relations.

Ubiquitous mindset more than technology as one can look on handful of problems from afore-said perspective.





## Chapter 3

# Problem formalization

For the purpose of modeling an ensuing version of the dynamic pickup and delivery problem is considered. Given a set of warehouses in a city plan an efficient route for each delivery that optimizes a given criterion is sought. This efficiency might be defined using one of these metrics: Delivery time, distance traveled, fairness...etc. On the other hand, a delivery destination can be any location inside the city. Every item, that has to reach its destination, has a homogeneous size and will be transported by a member of homogeneous fleet, which has a limited load capacity. Access to the cargo or loading order is disregarded. Vehicles initially start their journey in a single depot, while the new orders are ceaselessly being created in all warehouses and vehicles might be reassigned during deliveries. All orders must be served.

In a single time instant  $t$  the problem can be formulated as a triple  $P^t$ :

$$P^t = (V, D_a, D_n) \quad (3.1)$$

$V$  - vehicles,  $D_a$  - assigned deliveries,  $D_n$  - new (unassigned) deliveries

where  $D_a = \{d_1, \dots, d_a\}$  are all deliveries already assigned to a vehicle and  $D_n = \{d'_1, \dots, d'_{n^*}\}$  represents new deliveries to be allocated to a vehicle.

Having individual vehicle positions defined in an euclidean space by a function  $\lambda$ ,

$$V = (v_1, \dots, v_k) \quad (3.2)$$

$$\lambda : V \rightarrow R^2 \quad (3.3)$$

each can be assigned a set of deliveries, using assignment function  $A$  matching a delivery  $d$  with a vehicle  $v$ ,

$$A : D_a \rightarrow V \quad (3.4)$$

while the last set  $D_n$  consist of new deliveries, that waits to be processed. When the delivery is delivered, it is removed from list of assigned and no longer of any concern.

Let us consider each pickup and delivery location for a single delivery as a node in directed graph, with regards to fact, that each pickup must precede its delivery (precedence constraint 3.13). Each order carries information when it was placed. Moreover consider only

orders that are already present in the source location at the given time. Consider delivery as a

$$D = \{(l_s, l_d, d, t^*) | l_s \neq l_d, (l_s, l_d \in R^2), d \in N, t^* \in R\} \quad (3.5)$$

Where  $l_s$  is a source location and  $l_d$  is a destination location,  $d$  is demand - number representing type of goods to be transferred and  $t^*$  is the time when the order was made. Each node in the graph has defined coordinates in the euclidean space and each arc contains a description of its cost (distance).

$$x_{ij} = \{(l_s, l_d, c) | l_s, l_d \in R^2, c \in R\}, \quad (3.6)$$

### 3.1 Objective function

At a time  $t$ , it is possible to formulate multiple traveling salesman problem (mTSP) over available delivery nodes  $\{d_a | (v_a, d_a) \in D_a, v_a \in \{1, \dots, k\}\} \cup D_n$  for  $m$  vehicles currently at a warehouse, because for the pickup and delivery locations that exists in same time, the problem is reduced to afore-said mTSP [16]. Hence visiting multiple nodes representing loading multiple items in a warehouse should have zero-cost link when the vehicle is not required to wait.

Problem formulation based on mTSP [9].

$$\text{minimize } \sum_{i=1}^n \sum_{j=1}^n c_{ij} \sum_{k=1}^m x_{ijk} \quad (3.7)$$

Minimize cost visited locations where:  $k$  - vehicle index,  $i$  - source index,  $j$  - destination index,  $x_{ijk}$  - if vehicle  $k$  visits location  $i$  before  $j$ ,  $c_{ij}$  - cost of traveling between  $i$  and  $j$ ,  $n$  number of deliveries,  $m$  number of vehicles

$$s.t \sum_{i=1}^n \sum_{k=1}^m x_{ijk} = 1, j = 1..n \quad (3.8)$$

Each location is visited by just one vehicle:

$$\sum_{i=1}^n x_{ipk} - \sum_{j=1}^n x_{pjk} = 0, k = 1..m, p = 1..n \quad (3.9)$$

When vehicle enters in a location, then it must leave from the same location:  $p$  intermediate location index

$$\sum_{j=1}^n x_{1jk} = 1, k = 1..m \quad (3.10)$$

Each vehicle is used exactly once.

$$u_i - u_j + n \sum_{k=1}^m x_{ijk} \leq n - 1, i \neq j = 2..n \quad (3.11)$$

$u_i$  - order of location  $i$  in tour

$$x_{ijk} \in \{0, 1\}, i, j = 1..n, k = 1..m \quad (3.12)$$

$k$  - vehicle index,  $i$  - source index,  $j$  - destination index,  $x_{ijk}$  - if vehicle  $k$  visits location  $i$  before  $j$ ,  $n$  number of deliveries,  $m$  number of vehicles

In other words, the problem can be viewed as a set of multiple traveling salesman problems, where each is valid for a short period of time. The calculated path can change between different instances, but it is the actual path that is traveled, which is important and should be optimized. In reality, actual solution to the problem as defined below, is an open path and a schedule for leaving each location( vehicles are allowed to wait), which can efficiently solve the mTSP, that will be only fully known in the future, when part of the distance has been already traveled. This basic formulation ensures each location is visited just once and by a single vehicle only. Schedule is represented as a node labeling  $B$ , where arrival time to the nodes (locations) by each vehicle is stored.

$$B_{ik} \leq B_{jk}, \forall i, j, k \quad (3.13)$$

$$B_{jk} - B_{ik} - t_{ij} \geq M * (x_{ijk} - 1) \quad (3.14)$$

$k$  - vehicle index,  $M$  - large positive integer,

$B_{ik}$  - arrival time of vehicle  $k$  at location  $i$

Precedence constraint 3.13 forces vehicle visit time of source location  $B_i$  to precede  $B_j$  delivery time for each item. Note that the inequity 3.14 allows vehicle to wait at given location as well as specifying the minimal travel time  $t_{ij}$  between locations.

One other constraint remains to be added. Adding a vehicle capacity (which is sometimes ignored in literature ) to the equations changes the nature of mTSP to very closely related capacity VRP (vehicle routing problem) also based on mTSP.

$$Q_{ik} - Q_{jk} - 1 \geq M * (x_{ijk} - 1) \quad (3.15)$$

$$0 \leq Q_{ik} \leq C, \forall i, j, k \quad (3.16)$$

$k$  - vehicle index,  $Q_{ik}$  - capacity of vehicle  $k$  at location  $i$ ,  $C$  - vehicle maximal capacity

Adding variable  $Q$  representing current goods in a vehicle at each location and restricting its maximal capacity. The equation 3.15 describes homogeneous size of the deliveries.

## 3.2 Solution representation

A solution is an assignment of locations and time schedule to each vehicle that should be visited in the future. It is considered valid, if it respects vehicle capacity, precedence constraint and minimal travel times between locations. Including time  $t_0$  is useful to indicate ordering and validity of the solutions.

$$((x_{111}, \dots, x_{nnk}), (B_{11}, \dots, B_{nk}), (t_0)) \quad (3.17)$$

$x_{ijk}$  - if vehicle  $k$  visits node  $i$  before  $j$ ,  $B_{ik}$  - arrival time of vehicle  $k$  at  $i$

$t_0$  - time when solution was generated,  $n$  - number of deliveries

### 3.3 Notation

Description of variables used in the formal definition.

$a$	assignment of delivery to vehicle
$B_{ik}$	time when location $i$ was visited by vehicle $k$
$C$	maximum capacity of each vehicle
$c_{ij}$	traveling distance from $i$ to $j$
$D_a$	set of assigned deliveries
$D_n$	set of unassigned deliveries
$i$	location index
$j$	location index
$k$	vehicle index
$m$	number of vehicles
$M$	large positive integer
$n$	number of assigned deliveries (number of locations/2)
$Q_{ik}$	payload of vehicle $k$ at location $i$
$t$	current time, e.g. time when problem is computed
$t^*$	time of order creation
$u_i$	node(location) potential, indicates order in tour
$x_{ijk}$	whether vehicle $k$ visits location $i$ before $j$

### 3.4 Metrics

Obtained solutions, while satisfying all defined constraints, will be also evaluated using these additional metrics. Minimization function as specified in equation 3.7 define us, what we want to optimize, but for a measurement of slightly different aspects, without the need of changing the problem formulation, following metrics are introduced:

#### Mileage

Metric minimizing mileage is important, because it conveys the information about one of primary cost factor of transportation - fuel consumption. Average distance traveled per delivery is used as it can better measure cost-efficiency in a transportation company.

$$\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n c_{ij} \sum_{k=1}^m x_{ijk} \quad (3.18)$$

$n$  - number of deliveries,  $m$  - number of vehicles,

$x_{ijk}$  - if vehicle  $k$  visits node  $i$  before  $j$ ,  $c_{ij}$  - cost of traveling between  $i$  and  $j$

**Average delivery time**

On the other hand, delivery time is the principal indicator of customer satisfaction with the service. Averaging time between creation of each delivery and the moment of reaching its destination is used.

$$\frac{1}{n} \sum_{i=1}^n (B_{jk} - t_i^*) \quad (3.19)$$

$n$  - number of deliveries,  $B_{jk}$  - delivery time at location  $j$  by vehicle  $k$ ,

$t_i^*$  - order  $i$  creation time,

## Chapter 4

# Allocation algorithm

In this chapter a detailed description of an algorithm, that is capable of solving the problem emerges from the formal description gained in the previous chapter. Deliveries, representing the input in such an algorithm, have to be transformed into a sequence of places to be visited and times, when they should arrive. Because of the multi-agent aspect of the approach taken, the algorithm, as well as the knowledge of the outcome will be distributed. Its formation will be based on agent interaction.

### 4.1 Agent interaction

Manners of the interaction are usually called protocols. The protocol will be closely linked with what was previously defined as an assignment function. Using a communication protocol, deliveries will be economically allocated to the best suited vehicles for their delivery. The one chosen for its versatile characteristic is a variation on the Contract net protocol [19]. That can be described in terms of 3 actions:

- The initiator sends out a call for proposals
- Each participant reviews it and bids on the feasible ones accordingly.
- The initiator chooses the best bid, which marks the participant, that is awarded the contract, while all other bids are rejected.

In principle any agent could act as initiator, be it a dispatcher or a vehicle, so agents can re-negotiate deliveries between themselves later on, but for simplicity, only the dispatcher is granted this role. Each participant contains a value function prescribing its interest in delivery. It could be set to minimize a range of different settings, including the best average time or shortest mileage, even having dissimilar preferences in each agent depending on the cost parameter used [15].

However in this case, the distance traveled, as gained by the cheapest insertion heuristics, will be used to estimate the cost of adding a new delivery. While the other metrics will be used ex-post to compare the quality of the solution, this will be the primary variable

minimized in accordance to the formal definition. When the cost is computed by the participant, it is announced to the initiator as an agent's bid. After a prescribed period of time contract ownership is awarded to the winner.

In practice, any accepted contract may change the optimal paths of several vehicles. There exists some strategies [18] for promoting change between agents consisting of improvement or re-negotiation of older contracts. For example: After a new order is accepted, agent may re-negotiate its deliveries, that are delayed due to this new order and pass them on to the other agents. Using some of these strategies could be a good idea for future work.

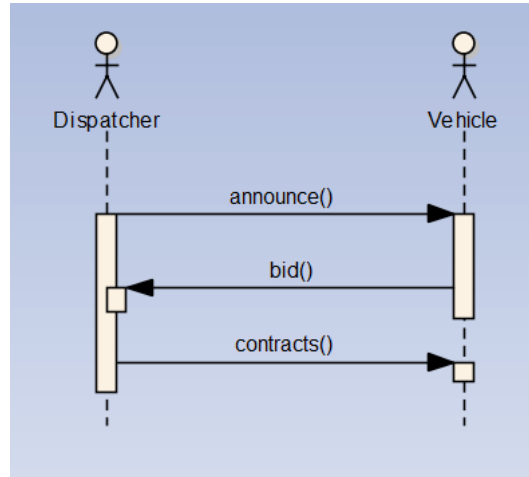


Figure 4.1: Interaction between vehicle and dispatcher

## 4.2 Evaluation of deliveries by insertion heuristic

Future path of the vehicle is not computed exactly, but estimated using insertion heuristic. Deliveries are progressively inserted into existing path in any possible position. Of these, only the best place in projected path is sought, so that the additional cost (accounted for by the new delivery) is minimized.

The algorithm makes use of another of the agent's features, his memory. Here, it stores two important information. First, the location of the closest warehouse, which is updated beforehand as the vehicle moves. Second, a sequence of valid bids, representing agent's interest in each delivery. Here the order is not arbitrary, but marks a deliberated preference in visiting each of the delivery locations. Naturally, it relies on the warehouse selection. For this reason, the evaluation is performed only when the vehicle does not move. Agent retains a set of known deliveries. When new request for evaluation is issued, this set is consulted first. If the delivery is not previously known, a bid is created by seeking an ideal position among memorized older bids.

- Bid by euclidean distance

Distance between any two points is approximated as an Euclidean distance by the latitude and longitude of the appropriate graph nodes.



- Bid by real distance

Is a more precise, but resource intensive approach, where the distance between any possible points is sought as a sum of distances on the shortest path in underlying map graph.

Of the two described approaches above, which were both implemented, the first one was used, because it is much faster. If a delivery is known, a bid for it has already been computed and it is skipped. However that does not mean it's value is fixed. When a new bid is inserted in the middle of a sequence, the ensuing bids must have their values updated to reflect the delay in delivery.

### 4.3 Allocation

The crux of the allocation is a dispatcher's table, which stores the valid bids. According to the assignment type, the deliveries are distributed between vehicle bidders.

#### By one delivery

For each delivery in allocation table, the bids are assessed independently and vehicle with the cheapest bid is chosen.

#### By full vehicle

When the allocation table contains enough entries, a single vehicle is chosen. To ensure that the cost is minimal, bids that represents last delivery of every vehicle are compared and the cheapest vehicle is chosen.

#### Ensure delivery

However these approaches do not guarantee that every delivery can reach its destination, as would be the case, if a single distant destination was never served, in favour of newly created deliveries close-by. To rectify this situation, a possibility to halt announcement of new deliveries to vehicles, until prior cargo was served, is added into algorithm.



# Chapter 5

## Simulation model

This chapter describes the simulation model, agents appearing in allocation algorithm and the environment, in which they act. The agent behaviour is described in terms of actions and interactions needed to fulfill their role.

### 5.1 Agents

A first step is to determine what is an agent. Then it is necessary to describe it by the means of its actions and interactions. From an object-oriented perspective we have 4 entities. There is a vehicle, a delivery dispatcher, a warehouse and a delivery. I have selected the vehicle and the dispatcher to act as agents for the following reasons. Single dispatcher is a single point that announces all unassigned and newly created deliveries from every warehouse to vehicle agents, that actively choose which delivery to accept. Why not choose warehouse as an agent and let it assign deliveries to passive vehicles, that just execute the assigned deliveries? Having vehicles as the decisive agents brings out better flexibility and possibility for future extension, when vehicles ponder additional constraints. When receiving a new order, the vehicle can actively choose the closest warehouse on its track and add it to the path.

#### Dispatcher

Dispatcher will handle deliveries from all warehouses and his job will be solely to inform mobile agents. When a new order is received, all vehicles are notified. Each vehicle will then respond with its estimated delivery cost. If no one is interested in delivery, it is re-announced by the dispatcher, in future also with increased priority [14].

#### Vehicle

Vehicle may load any cargo, up to its maximal capacity. When accepting new order, appropriate goods that fulfill delivery requirements is chosen from closest warehouse in relation to the vehicle. Euclidean distance is used. It is up to the vehicle to decide, when to visit a warehouse again, but serving all loaded deliveries first is obligatory, before it is allowed to bid again.

## 5.2 Agent interaction

Represent the means agents communicate with each other. In the model, it is restricted to the same interaction, as prescribed by the Contract net protocol in previous chapter, as there are no other agents present.

- announce  
announce new deliveries to all vehicles
- bid  
vehicle bids on known unassigned deliveries.
- contracts  
announce to all vehicles which deliveries they have won

## 5.3 Agent actions

Action is a method, that have an impact on the environment, a way agent can interact with its surroundings. Actions and activities usually form the lifecycle of the agent, a repeated behaviour, describing how the agent reacts in different situations.

### Dispatcher actions

Does not have any actions, as he does not directly interact with the environment, communication with vehicles is not considered an action.

### Vehicle actions

During its lifecycle a vehicle performs several sets of actions. From moving to warehouse, loading goods, reaching destination and unloading. By accepting a contract, a vehicle makes a promise of performing these actions.

- load  
load deliveries
- move  
move to a (delivery/warehouse) location
- unload  
unload delivery

## 5.4 Pseudo algorithms

A behaviour of the agents can be described by the following pseudo-code.

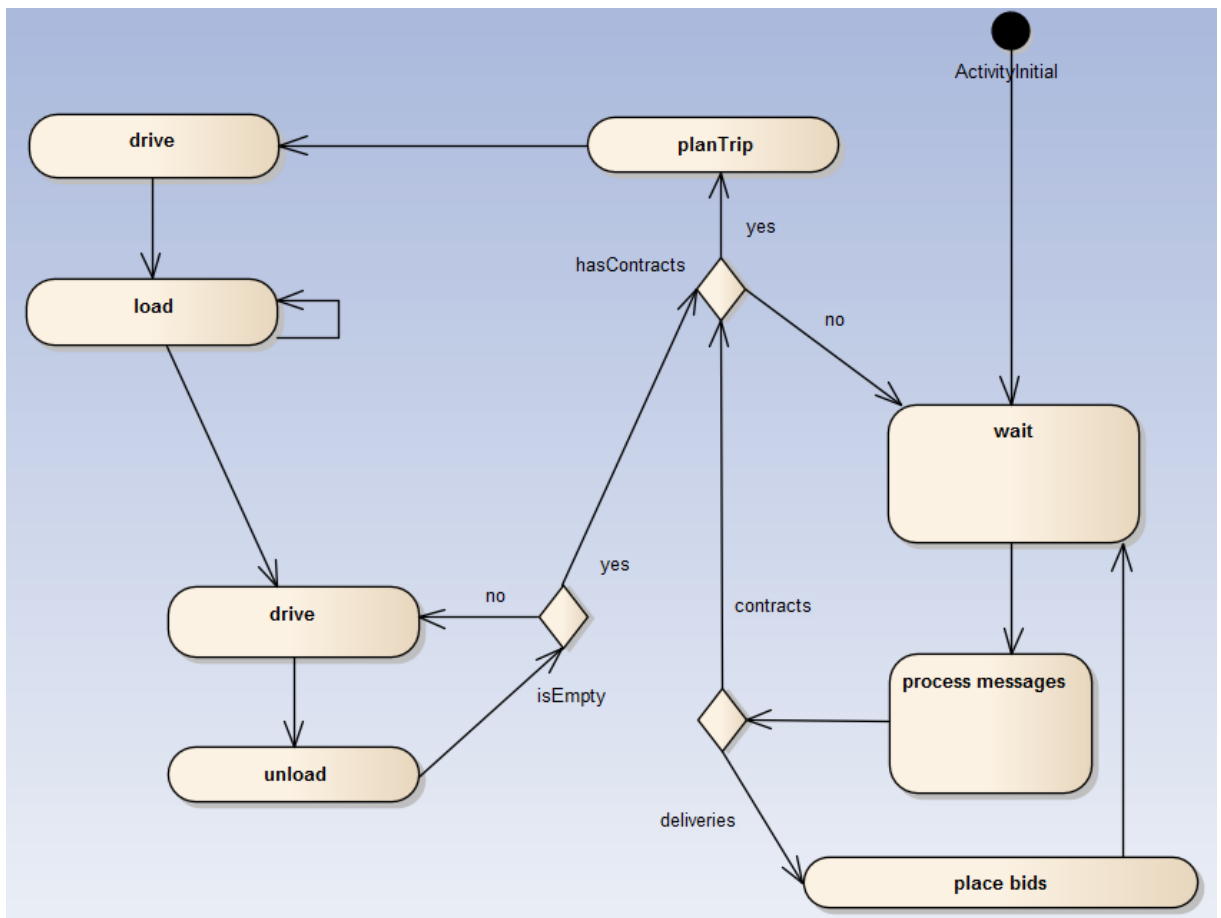


Figure 5.1: Vehicle lifecycle

## Vehicle

```

while(true)
  D:= announce()
  for (d: D)
    bid(d)
  end
  C := contracts()
  move(findWarehouse())
  load(C)
  for(d: C)
    move(d)
    unload(d)
  end
end
end

```

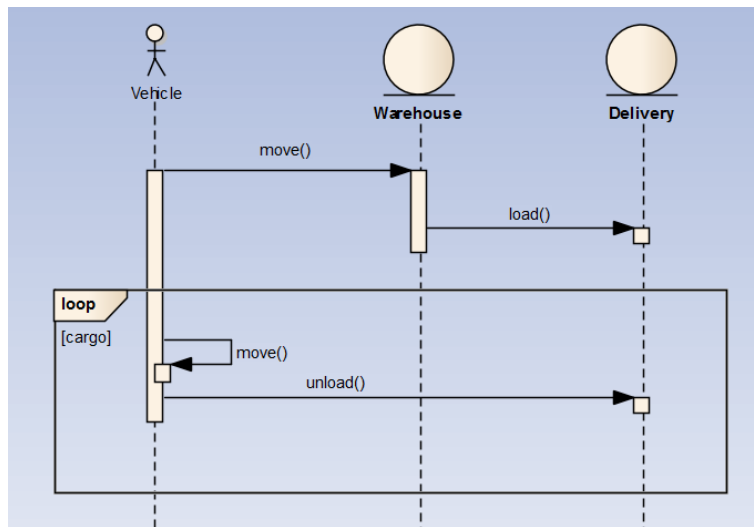


Figure 5.2: Deliveries diagram

## Dispatcher

```

while(true)
  D := D + getNewDeliveries()
  for (d: D)
    announce(d)
  end
  D_a := contracts()
  D := D \ D_a
end
  
```

## 5.5 Environment model

A model of the environment consist of a street graph and agents. During analysis, it became apparent that vehicle should be modeled not as a single entity, but as a driver agent, that stores bids and plans in his memory, and physical representation of the vehicle that moves on the graph. The graph is made of street nodes, while some of them may represent a warehouse or a delivery point. In our case all warehouses have unlimited supplies of goods and a vehicle always chooses the closest one.

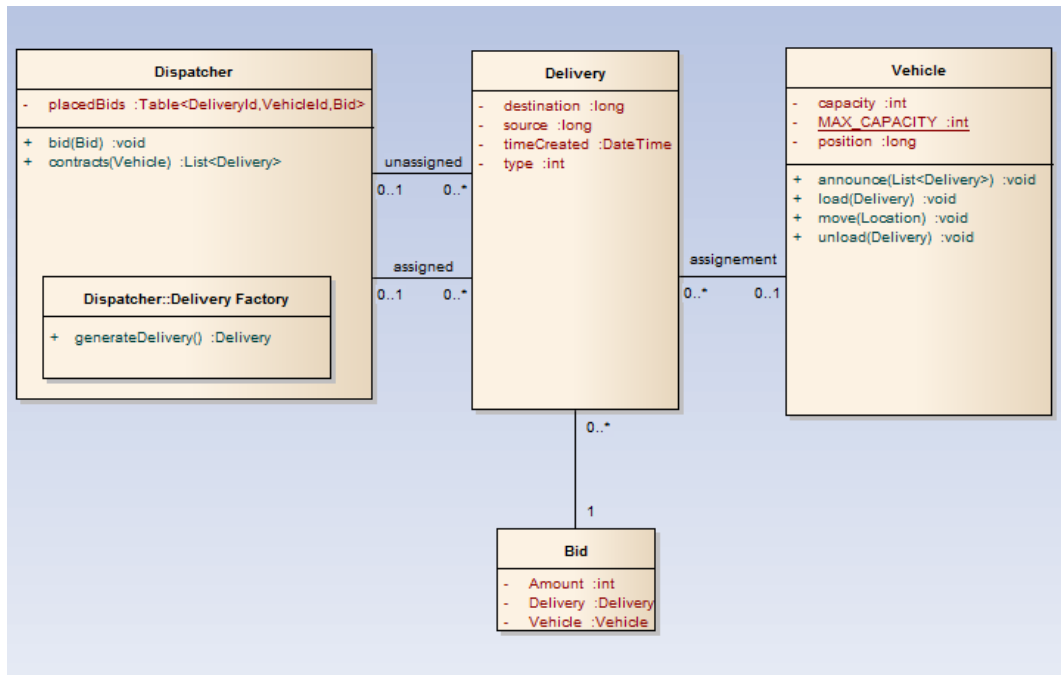


Figure 5.3: Class diagram of interactions

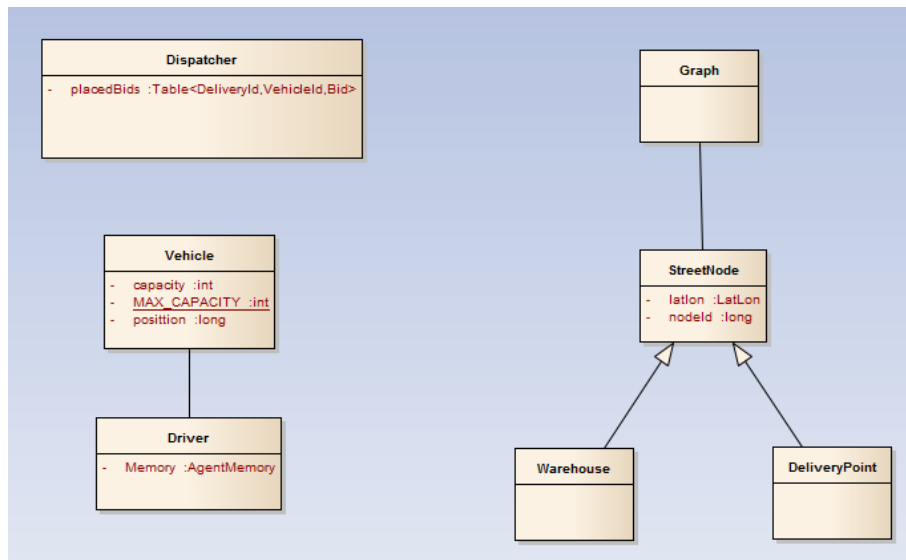


Figure 5.4: Environment diagram





## Chapter 6

# Implementation

The reviewed problem will be modeled on an agent-based platform named AgentPolis[5]. A suitable-sized city will be selected in the next chapter to show some tangible results. The platform enables agent movement and communication by processing discrete events. Implemented or extended parts can be categorized in 3 sections: a specialized simulation environment, the visualization and an application launcher.

### 6.1 Platform

The programming platform used for the simulation contains several stacked-up projects running Java SE. At the bottom lies a general event-based platform Alite [6]. Its core concept is a priority event queue, that sort events in order of execution time. From there, events are taken one by one and processed by a single thread. Upon it lies AgentPolis, extending the general framework in the terms of urban simulation. Key concepts like agent's lifecycle, actions and sensors are defined here. While implementing the delivery environment, one other project based on AgentPolis was used. From PublicTransportation, several activities were reused, mainly concerning vehicle movement and basic definition of a vehicle.

#### 6.1.1 Visualization

Platform provides two distinctly different visualization options. Internal, written using java Swing and external in collaboration with Google Earth[1]. First one is implicit and required only registering the agent type, when simulation scenario is created. It was used for debugging and only the moving entities - agents - are shown. Second option is preferred and was used for visualization of vehicles, as well as warehouses and delivery points. It required an additional work in wrapping data in entities, that can be shown in GoogleEarth, as well as linking them with appropriate updaters, that periodically refresh position of said entities shown in the application. Wrapper class was called ABuildingUpdateGEFactory as it was based on code used for buildings visualization, which fitted well the purpose.

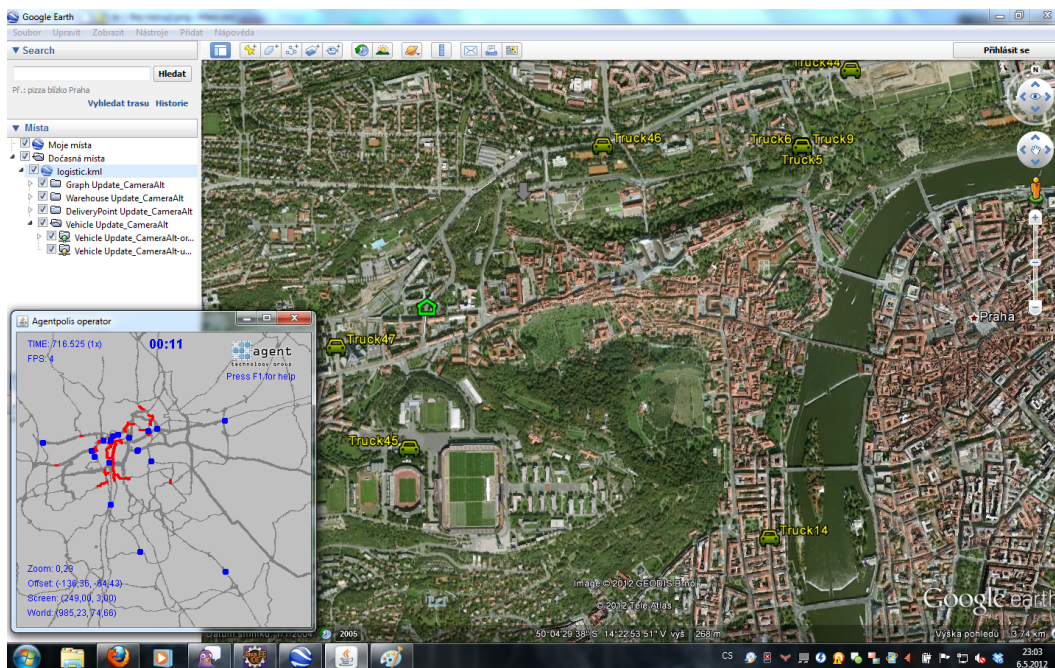


Figure 6.1: Visualization, on the left using Swing, behind in Google Earth

## 6.2 Environment

### 6.2.1 Communication protocol

Modified version of the communication protocol used in publictransport demo example[4] is used. The original protocol allowed only communication between two agents based on agent's id, which I found limiting. Now, messages can be sent either by agent's id or the agent registers for a message classname and listens for any message of that type (broadcast). A class was created for every agent type (vehicle and dispatcher). An enumeration describes each state of agent. Dispatcher (announcing new deliveries, awarding contracts), Vehicle (bidding).

### 6.2.2 Sensors

The information from the environment are not accessible directly, but any kind of knowledge is gained only through sensors, given to agents at the moment of their creation.

#### Position sensor

The position sensors provided a vital information about location of vehicles in the environment. They were reused from the AgentPolis platform in two different ways. Passively, as in case of the AgentPositionSensor, when the vehicle driver agent queried its position in

the specific sections of the vehicle's lifecycle to determine his relation to the closest warehouse. Actively, as in case of `VehiclePositionAction`, when the vehicle was issued to unload the goods as soon the delivery location is reached.

### Mileage sensor

A new sensor, which was built around the `VehiclePositionAction`. It is triggered whenever a vehicle changes its position. It compares current position with the previous one and adds the distance between these neighbouring points (taken from the street graph). The total distance traveled since the sensor was created - in meters (the unit of underlying graph) - is then stored in its inner variable. The sensor can be queried for this value on demand.

### 6.2.3 Dispatcher implementation

Dispatcher was implemented as an agent with a simple lifecycle of periodical reannouncement of deliveries and sending contracts. Creating new deliveries using the `ADeliveryFactory` class was in the end moved out of the agent class, instead a callback, that is called for each new delivery is present.

### Announcing deliveries

As soon as a new delivery is created, it is announced to all vehicles using this callback, unless ensure delivery is true and number of unassigned deliveries is above capacity. Afterwards, if the delivery is not assigned to a vehicle, it is reannounced again, after the given timeout using a callback from event queue. The unannounced deliveries are backed by a map, that keeps the order of insertion.

### Awarding contracts

The bids are stored in a two-dimensional table, indexed by delivery id and vehicle id. The value of a vehicle's bid is replaced, if a new one bid is received. Depending on the dispatcher settings, a delivery is awarded to the lowest bid in the table or the cheapest row. Cost of the row is determined by the vehicle's most expensive (last bid in the row), if it should be loaded full. If any assignment is successful, the table of bids is cleared and will be filled again, when the remaining deliveries are reannounced. That concludes the bidding cycle.

This was a primary reason, why ensure delivery configuration option was implemented very simply by limiting announcement of new deliveries. More gradual process would require a complete redesign of bidding process. The dispatcher's allocation table allows for updating a bid before contracts are assigned repeatedly, but the table in vehicles does not. It would not have been easily compatible with insertion heuristic, as this depends on previously computed path fragments. Joint benefit is also a much faster simulation run with high amount of deliveries.

### 6.2.4 Vehicle implementation

Vehicle was implemented by its physical representation in a class called `Truck`, extending basic vehicle representation by the list of deliveries. Then a driver agent was added, to govern over the planning of its movement and evaluation of the deliveries. These two were bound into one lifecycle activity 5.1, which means that the driver cannot leave his vehicle.

To keep a clear design concept, the lifecycle is only concerned with the vehicle and delivery movement and a new interface `BidStrategy` was introduced to contain the logic for the evaluation of deliveries in the bidding process.

#### Driver memory

Memory is used to store parts of the future plans and issued bids. Primary purpose is to share the information and serve as a kind of interface between vehicle lifecycle and `BidStrategy` class. The agent keeps 3 lists of deliveries in his memory. The list of current bids, the list of contracted deliveries and the list of deliveries on board. After a bid was won, the delivery moves from first list to the second, when it is loaded to the last, and when its delivered it is removed completely.

#### Bidding

Vehicle can bid, if it is waiting in a warehouse, or after the last delivery is served, to ensure bid validity. In any case, it has stopped and it is not moving. The bids are stored in a map in agent's memory. Reannouncing same deliveries has no effect, their values are cached until a contract is received and all bids are cleared.



Figure 6.2: Planning delivery path in real time

#### Planning delivery path

When contracted deliveries are received, the deliveries are sorted according to the order computed at the bidding time. Even if some deliveries were bidden, but not contracted, their removal should not matter, according to the Triangle postulate. Then a trip on the graph for the first delivery is computed and vehicle starts moving. After each stop, a new trip is computed, until all deliveries are served.

For a planning of the delivery path, an A\* planner implementation with an Euclidean heuristic is used. A\* algorithm with an admissible heuristic (such as Euclidean) is guaranteed to find the shortest path in a graph. A heuristic in this case serves only to speed up the finding of the solution.

### 6.2.5 Map

The interaction with the map is fully handled by the AgentPolis platform. It enables loading of any OpenStreetMap[2] of reasonable size. Because usually a selection from \*.osm file or files is cut out to form the map, implying that only a part of the real map will be used, the simulation must handle gracefully any problems arising from its imperfect foundations. Commonly, on the edge of a simulation map, a small amount of roads, that are not connected with the rest, may appear. If a vehicle is sent to an unreachable destination, this incident is logged and vehicle continues with the next delivery. In case of an unreachable warehouse, the vehicle in question blacklists this warehouse and continues using others. If any vehicle has no reachable warehouse, the simulation is halted with an error. For details on the map selection see [Experiments](#).

#### Warehouse nodes



Warehouse nodes are created as a random selection of nodes. An initialization task inside the platform is used to select them, after underlying graph has been loaded. It is assumed that they do not change during the run of the simulation.

#### Delivery points



Delivery points are randomly chosen nodes of the graph, serving as a destination for the delivery. Their selection is done on purely non-deterministic basis from all available graph nodes. The frequency of their creating is governed by a configuration parameter, that determines maximum time instant before a new one must be created. Possible future extension would be to allow cyclically change this option on-the-fly for more dynamic delivery environment.

These special nodes are thin wrappers around graph nodes, for the time being kept in a separate data storage, because they contain additional information, that is not directly relevant for the path planning process. Furthermore, it does not contain relevant information for every node, only a small portion, which has this addition thin wrapper. They are used for locating the closest warehouse and may contain some visualization information.

## 6.3 Producing results

The simulation output is a log file of events. Not all events are interesting and only those, that were chosen for their informative value are stored in a csv file by an event processor. In the case of parcel delivery, two types of events were specified for logging. Those pertaining to the state of a delivery - new, assigned, loaded, delivered - and those regarding the position of the vehicles. For more clarity, the vehicle events are used only on places of interest,

which are basically 2 kinds of waypoints - warehouses and delivery points. For debugging, an AgentPolis vehicle event could be used, showing movement between any 2 points, but for a practical use is too verbose.

To understand how metrics are built from the logged events, firstly the parameters of afore-said events are of concern.

The purpose of vehicle events is to show the information about vehicles, in this case focus is placed on the mileage traveled by each vehicle. Delivery events on the other hand, track alteration of the delivery states in time. Some thought was given to minimize the amount of rows generated, and finally, due to assignment and load delivery action, in events, the deliveries are referred by the list of ids. In other event states this convention is upheld too, to keep the consistency.

### 6.3.1 Launcher

In a single simulation run metrics (3.4) are not produced, but for convenience are included into my simulation launcher. This small standalone java application periodically executes simulation for a given number of iterations and configurations. The configurations are created as a Cartesian product of all possible values in launcher.config. Resulting metrics are computed from the simulation output and represents a new line in the overall results. If an error occurs in the simulation, no data are written for this run.

For the average mileage per delivery metric, a map is used, where the total distance traveled by each vehicle is stored. This value is updated as the log file is read row by row. Gradually, the number of delivery items created and delivered is counted. Finally, the total mileage of all vehicles is divided by the number of finished deliveries. This metric does not account for the number of vehicles used.

For the average delivery time metric, 2 maps are used for the creation and delivery times, with the delivery id as a key. After that, a sum of the time differences is determined, then divided by the number of delivered items. If no deliveries were served, the result is set to 0. This metric omits deliveries, that has not yet reached its destinations.

In the resulting file, a scenario configuration is written, as well as the mentioned metrics and number of items generated/delivered.

# Chapter 7

## Experiments

After the model was implemented and its behaviour tested on a few small random scenarios, a more rationalized approach in conducting the experiments had to be taken. Here, the observations are outlined, as well as the consequent behavior of the model based on different parameters provided.

### 7.1 Experiment setting

All experiments were conducted on an OpenStreetMap of Prague [11], bounded by a rectangle of latlon coordinates 49.936, 14.215, 50.191, 14.725. Moreover, the map was further filtered to contain only the pure street information by the means of Osmosis[3] program.

```
osmosis --rx file=input.osm --bounding-box top=50.191 left=14.215 bottom=49.936
right=14.725 completeWays=true --used-node --wx input_all.osm
```

```
osmosis --rx file=input_all.osm --tf accept-ways highway=motorway,motorway_link,
trunk,trunk_link,primary,primary_link,secondary,secondary_link --used-node
--wx prague_highway.osm
```

Each experiment was executed 10 times to cope with a randomness factor, given by the random location of warehouses and deliveries. An error caused by a short duration of the simulation and this randomness was estimated. Following issues were looked upon: How the behaviour of the allocation algorithm changes based upon throughput of the system. The correlation between number of vehicles and warehouses on one hand and average delivery time and mileage on the other. The comparison of allocation algorithms and the influence of a simulation duration. The metrics (average delivery time and mileage per delivery) used are defined in the section 3.4.

#### 7.1.1 Scenario parameters

Apart from the two simulation parameters

- agentMoveSpeedInkmph
- simulationDurationInMilisec

the model of parcel logistics contains numerous parameters, a subset, that was deemed important, was used in the ensuing experiments.

- deliveryFrequency
- ensureDelivery
- numberOfVehicles
- vehicleCapacity
- vehicleFullLoad
- warehouseCount

Parameters that were kept constant:

- reannounceFrequency
- vehicleLocationHome

## 7.2 Experiment results

### Warehouses

The importance of having more warehouses increases with the introduction of new vehicles. It is clearly shown that the major change is between 1 and 2 warehouses, while after that, the change is not so apparent at the first glance.

### Vehicles

The key question regarding vehicles in this delivery model, is how to determine a closely optimal amount of vehicles, while regarding the actual simulation configuration. If we look from the average delivery time perspective, we search for a point where adding new vehicles has no significant benefit. Demonstrated on a single example (7.3): 10 vehicles is sufficient for the delivery frequency of 100s, 50 vehicles are enough for the frequency of 20s. Notice the linear reduction of the lines in the middle. They are over the throughput of the system and deliveries are generated faster than delivered. Increasing the generation frequency causes new deliveries to be passed on later, until a sufficient amount of vehicles is available, then the delivery time is nearly constant.



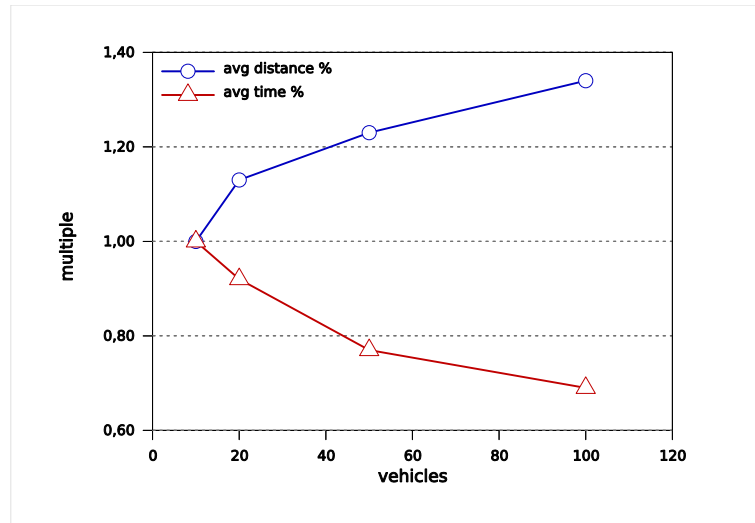


Figure 7.1: Increasing amount of vehicles in experiment with an insufficient throughput. Simulation: 1h, warehouses :2, capacity: 20, delivery frequency:10s, fully loaded

### 7.2.1 System throughput measurement

System throughput is the ability of the allocation algorithm to assign deliveries in certain quantity and transport them to a given location.

I have looked upon the system from two different angles, first when amount of deliveries in the system is low, and vehicles are not fully utilized. Then with the over-abundance of deliveries.

#### Low amount of deliveries

Increasing the vehicle capacity or count has no effect on the performance of the allocation algorithm, if vehicles are permitted to leave only partially loaded (7.6). However, if vehicles are forced to wait for a full load, then there can be seen a drastic increase of the delivery time (about 50% in this case 7.7), while savings on the mileage are only about 20% (7.8). Unsurprisingly, making the changes on the vehicle count does not matter, as a small amount of vehicles is needed to cover all deliveries.

#### High amount of deliveries

Vehicle capacity on the partially loaded vehicles has no effect. However, if the vehicles must be filled completely, the influence of capacity can shift both ways, even if this change may not be very significant. This is, because increasing the load capacity of an insufficient amount of vehicle helps faster delivery, while having a lot of vehicles with lower capacity causes more concurrent deliveries by more vehicles, than if their capacity was augmented.

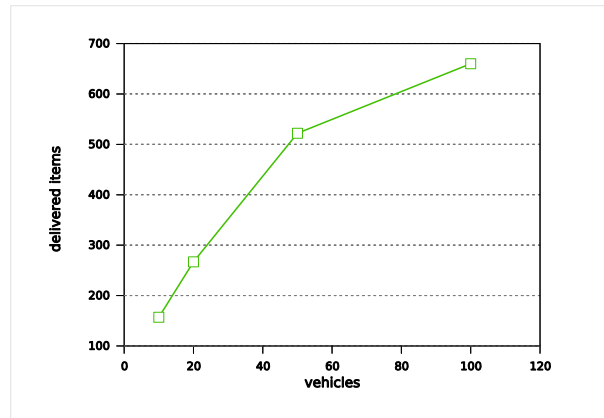


Figure 7.2: Increasing amount of vehicles in experiment with an insufficient throughput. Simulation: 1h, warehouses :2, capacity: 20, delivery frequency:10s, fully loaded

Another interesting observation is, that in a system with enough deliveries, the average delivery time and the average mileage are linearly dependent on each other(7.1). This holds true, because vehicles are not required to wait and move without great delays.

Other potential problem with the metrics is, that they are simulation-time dependent, in the case of delivery generation over-extension beyond the throughput of the system. This may be an obvious fact, chiefly when put by a simpler example, but may cause difficulties with comparing performance with different problem configurations. The table below shows an example how generation of deliveries distorts the average delivery time based on simulation duration. Because more deliveries are created than served, an increasing backlog of deliveries is being queued and the delivery time is prolonged more and more.

Similar may be observed, when looking on the numbers of new deliveries and those, that reach their destinations. Here, no small number of undelivered goods are those, that are on board of vehicles when the simulation has ended. On the first glance, the ratio of delivered to newly created goods may be increasing with longer simulation time, but if an absolute number of undelivered goods is growing, the resulting metrics are still unstable for the same reason.

On a more brighter note, numbers in the table for 100 vehicles (finally representing a sufficient throughput - as seen on 7.3 ) - are very close, for both simulation times, showing that the simulation time of only 1 hour can be quite accurate.(under 0.1 % difference for the average distance  $\doteq$  10meters, 9 % for the average delivery time  $\doteq$  1min )

### 7.2.2 Bidding with real distance

In all our other experiments, the vehicles bidded on deliveries with an Euclidean distance cost estimation of the path. Here, we look at what difference does it make, in terms of additional distance traveled in the first case and computation effort in the second.

From the results in the table 7.2, it can be seen that with using a planner to compute the real distance, the simulation can quickly became too obstinent to model in the real-time or

duration	vehicles	10	20	50	100
1h	avg delivery time	1487563ms	1235975ms	743768ms	598227ms
	avg distance	12441m	14826m	18107m	18503m
	avg no. of new	726	716	719	719
	avg no. of delivered	165	267	492	593
2h	avg delivery time	2893122ms	2222790ms	1014312ms	547957ms
	avg distance	11878m	12903m	17298m	184938m
	avg no. of new	1428	1442	1443	1435
	avg no. of delivered	351	637	1132	1320

Table 7.1: Table comparing results for two different simulation durations

duration (ms)	vehicle count	capacity	full load	del. freq.	distance (m)	time (ms)	new	delivered	real time taken
<b>Euclidean</b>									
3600000	10	20	true	10	11910	1559334	721	147	<b>00:00:09</b>
7200000	100	20	true	10	17100	1832411	1446	798	<b>00:01:09</b>
<b>Real distance</b>									
3600000	10	20	true	10	11766	1558308	718	146	<b>00:21:59</b>
7200000	100	20	true	10	N/A	N/A	N/A	N/A	<b>47:42:00*</b>
Remaining parameters are Ensure delivery: true, warehouse count: 10									
* value estimated from a 1/9 duration of a single iteration, all other values avg. of 10 iterations									

Table 7.2: Performance with the real distance bidding.

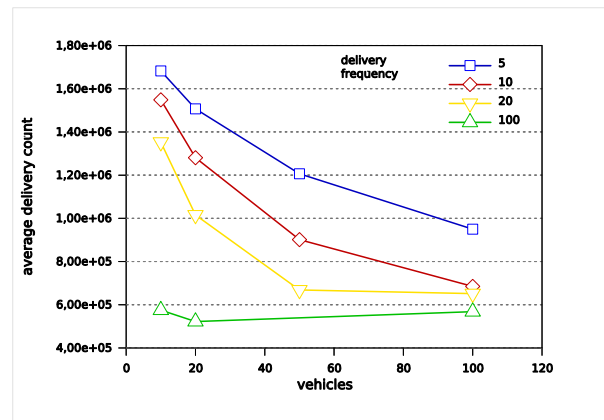


Figure 7.3: Finding optimal amount of vehicles by delivery time for several delivery frequencies. Simulation:1 h, warehouses :2, capacity: 20, not fully loaded

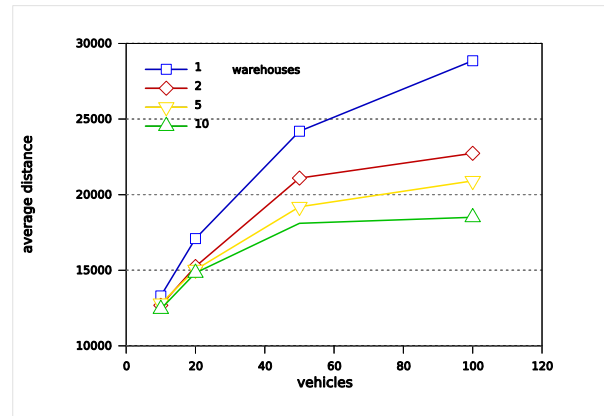


Figure 7.4: Correlation between amount of warehouses and distance

even in a reasonable time-period. The expected benefit of reduced mileage is an insignificant 1% difference, which in the mentioned experiment is below a margin of the random noise factors. Due to the cited computation complexity, larger experiments were not attempted and henceforth, the exact road cost was deemed to be too demanding without significant benefits, while the preference of the Euclidean distance was justified.

### 7.3 Simulation run details

All experiments were conducted on a personal laptop with a Intel Core2Duo T8300 processor 2,4 GHz with a 4GB memory and a 32-bit Windows 7 operating system.

All simulations were run with maximal simulation speed and without any visualization by a simple launcher app, that spawns a new Java virtual machine for each run, to filter of

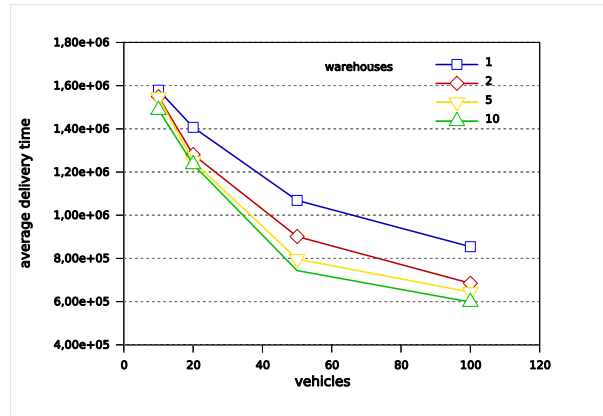


Figure 7.5: Correlation between amount of warehouses and time

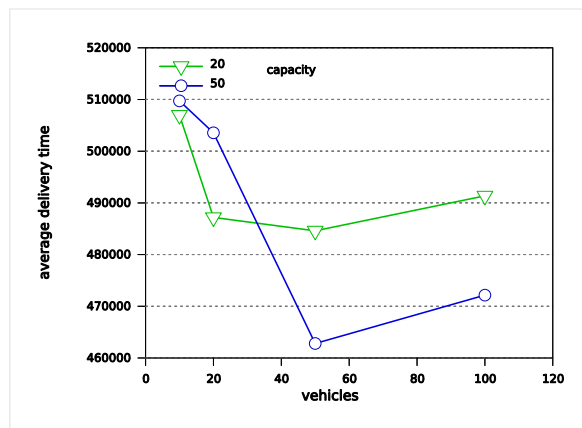


Figure 7.6: No influence of capacity in a system with low amount of deliveries and unfilled vehicles

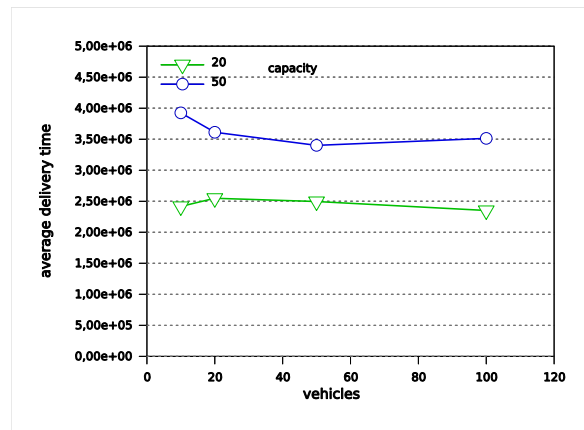


Figure 7.7: Significant difference in delivery time with increase of capacity in a system with low amount of deliveries and fully loaded vehicles

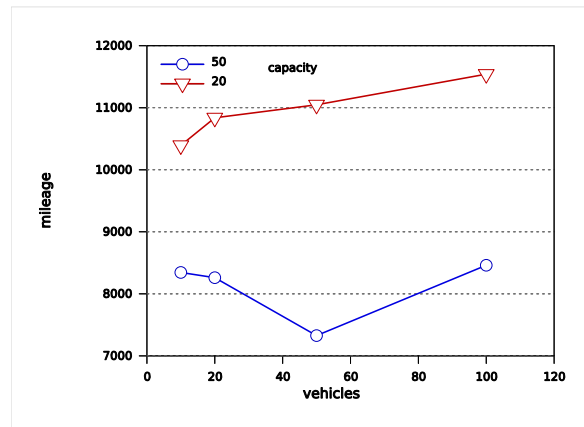


Figure 7.8: Mileage saved by the use of vehicles with higher capacity

possible influence of previous experiments, with the following vm arguments:

```
-Xms512m -Xmx512m
```

## Memory

Memory is not an issue in this simulation, the largest portion is probably taken by the loaded .osm map of Prague. It drops if a smaller map is used. The measured consumption (without visualization) was roughly between 250-270 MB, depending on the exact configuration. When running with a visualization, usage may peak to some higher values, but assigned 512 MB was always sufficient.

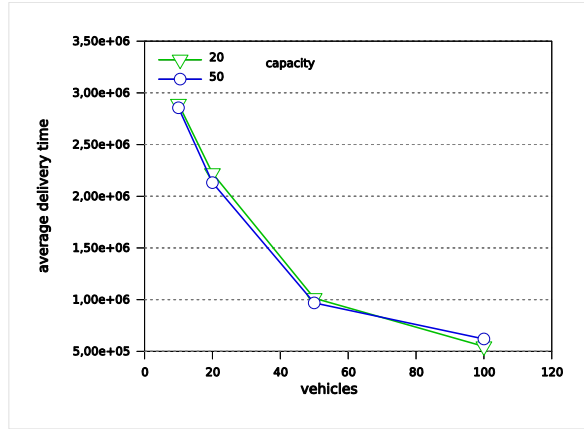


Figure 7.9: No influence of capacity in a system with high amount of deliveries and unfilled vehicles

duration (ms)	vehicle count	capacity	full load	del. freq.	ware-houses	ensure delivery	new	delivered	real time taken
7200000	10	20	true	10	10	true	1428,2	340,8	<b>00:00:11</b>
7200000	100	20	true	10	10	true	1446,1	797,8	<b>00:01:09</b>
7200000	10	50	true	10	10	true	1449,8	358,3	<b>00:00:27</b>
7200000	100	50	true	10	10	true	1432,9	867,1	<b>00:05:44</b>
3600000	100	20	true	5	2	false	1431,3	666,7	<b>01:51:41</b>
3600000	10	20	true	5	2	false	1434,1	162,8	<b>00:00:44</b>
3600000	100	20	true	20	2	false	362,4	166,2	<b>00:10:08</b>
3600000	100	20	false	5	1	false	1430,7	820,7	<b>00:08:08</b>
86400000	20	20	false	20	10	false	8622,1	8100,6	<b>00:10:51</b>

Table 7.3: Performance for different simulation runs.

## CPU

Usage vary and is dependent on many factors, most noticeably the speed of the delivery generation, the simulation duration, the capacity of vehicles and their count. The measurement might be inaccurate, in the range of seconds by a small margin, as the numbers were not taken from simulation logs, but additionally from the timestamp of the simulation's result file.

Values in the table were chosen to represent a wide range of scenarios very simply. Usually the maximal and minimal values of the key attributes are present. First expected behaviour, that was confirmed, is that allocating deliveries for a partially empty vehicle is always faster, than in the case when fully loaded vehicles are required. This comes from the fact, that vehicles in the latter allocation strategy have to asses more deliveries in otherwise very similar conditions. This, combined with the expectation of more stable times, is the reason, why majority of results in table 7.3 are fully loaded.

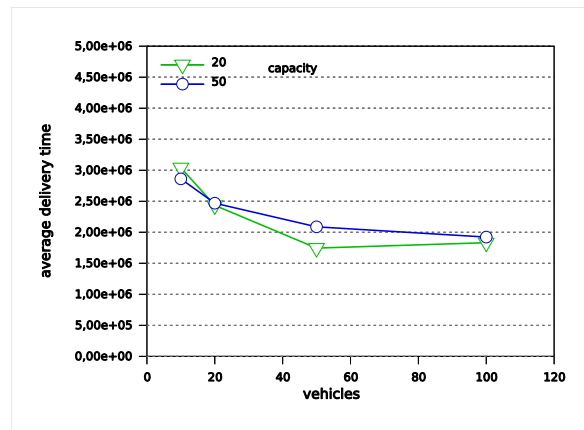


Figure 7.10: Influence of capacity in a system with high amount of deliveries and fully loaded vehicles

From the table is also apparent, when comparing first four lines, that increasing the capacity is more computationally intensive than adding new vehicles. Increasing a vehicle count number is linear, when ensure delivery is true, but grows very quickly within the scenario with high amount of deliveries (low delivery frequency). This brings us to what has the most adverse effect on allocation performance. It is the number of unassigned deliveries, or more precisely deliveries, that are offered to each vehicle.

### Exchanged messages

From performance point of view, another insightful perspective to observe, may be the number of messages agents exchange between themselves. It could show us increase and decline in the activity of the agents.

The number of messages sent together seems to grow linearly towards the vehicle count, while the total amount of messages sent by all vehicles to dispatcher grows far more quickly, than in the opposite direction. This corresponds to growing number of vehicles, that are free to answer and place bids, as the full and moving vehicles do not respond. This example intentionally show a case with low vehicle capacity. The dispatcher messages are less and less frequent per a single delivery, because more and more of them are taken on the first announcement, causing the number of messages sent by the dispatcher to grow more slowly.

## 7.4 Summary

It was verified that performing simulation in duration of only 1 hour was precise enough to grant credible results. Most significant difference in warehouse count was seen between 1 and 2, while adding more had only a little impact. From performance-wise point of view, the greatest influence was by the number of unassigned deliveries each vehicle had to evaluate, after that was the vehicle capacity and the vehicle count as such. Limiting the number of unassigned deliveries had serious impact on the of computation, bringing



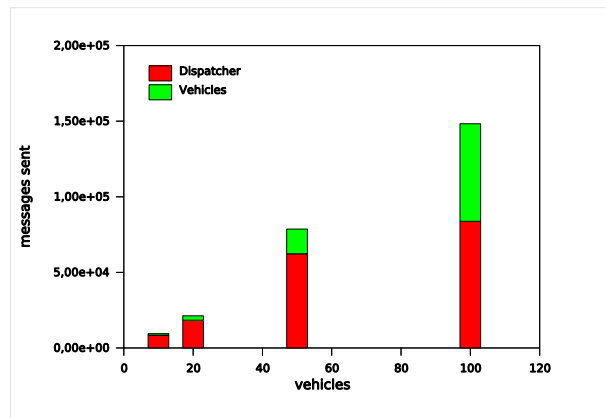


Figure 7.11: Agent interaction by the number of messages sent

a value previously nearly 2 hours to a 5-minute range. A great performance penalty with almost no gain was seen, when using the real distance instead of the Euclidean in the bidding process. Looking at the messages that agents sent each other, we saw a shifting trend in the direction of the messages, while bulk of the messages, caused by the continuous reannounced by the dispatcher, changed into the vehicle bidding messages, if their number was augmented.



## Chapter 8

# Conclusion

After becoming familiar with the concepts of agent-based modeling and some previous works in parcel logistics domain, the formal definition was assembled and metrics for its evaluation set. Agents in the domain were identified as a dispatcher and a vehicle.

Using a working model of parcel delivery in an urban area, the formalized problem of parcel logistics was analyzed in the terms of agent-based modeling and several interesting observations were gained. Two variants of an allocation algorithms were implemented on the AgentPolis platform, with no clear winner, each more suited for a different situation. Assigning deliveries by one and issuing vehicles without full load resulted in short delivery times and excelled in environment with low delivery generation. Waiting for a full vehicle load resulted in a significant savings in mileage of all vehicles, but generally at a greater loss of delivery time, which is fine, if we focus on the cost of transportation only. The model also allowed us to estimate an ideal count of vehicles and warehouses and to optimize the transportation service.

Two major bottle-necks were dealt with, that at first prevented sufficient scaling of the model. The speed of planning, that was proven crucial in delivery evaluation and resulted in the replacement of A\* planner with Euclidean distance approximation where possible and the number of deliveries assessed by each vehicle. If the generation of deliveries exceeded capacity of the transportation system by a significant margin, the computation was slowed down significantly. Limiting the dispatching to the oldest portion of undelivered items reverted computation time to the same values with the added benefit of having a guarantee that the items are delivered.

Agent-based modeling was shown as a good and viable approach on modeling parcel logistics, with the attainable application of its far-reaching possibilities. With having a model, it can now grow and improve and be used for optimization of its parameters. A distance, for example, with a realistic set of data could be used to determine the best position of the warehouses in the city, given the exactly same time and distance metrics. The optimal count of the vehicles with regards to their actual load in the real time can be considered. With a slight deviation of purpose, such a model can be used even for a range of different things than just a parcel logistics: for the fire brigades to dispatch the closest truck with regards to a water source location, to assign volunteers to clear the snow in the city and salt the roads from the nearest store.

## 8.1 Future work

Future work and enhancements of the presented model may continue in many directions. Several possibilities are drafted below:

- Movement

How the delivery model is impacted by gradual slowing down and stopping at destinations, driver work shifts, dispensing of fuel at the petrol stations. Using a path planning with auxiliary criteria - including congestion, other traffic, handling undelivered items, or time windows.

- Ensure delivery

A number of possibilities to improved the taken simple approach exists. To ensure that a delivery reaches its destination requires a delivery time window, a dispatcher capacity limiting announcing of new deliveries, or an order of deliveries modified by the creation time - an aging process. The dispatcher could set a reward, increasing with time, until the delivery is contracted. Another possibility could be a delivery aging process, that would mean contracting done in several rounds with different costs, while at least some of the oldest deliveries must be contracted.

- Agent interaction

Allow bidding when the vehicle moves and the closest warehouse changes. The vehicle agents could negotiate between themselves, swapping some deliveries to improve their paths. While in a depot, if more vehicles are waiting, only one would respond, while the others will be “sleeping”.

- Delivery frequency

Dynamically changing the speed of the delivery generation to include peak hours to form a cyclic period and generation of a small portion of deliveries known beforehand.

# Bibliography

- [1] Google Earth website, May 2012. <<http://www.google.com/earth/index.html>>.
- [2] OpenStreetMap website, May 2012. <<http://www.openstreetmap.org/>>.
- [3] Osmosis documentation, May 2012. <<http://wiki.openstreetmap.org/wiki/Osmosis>>.
- [4] Public Transport project, May 2012. <[http://merle.felk.cvut.cz/redmine/projects/agentpolis/wiki/Public\\_Transport](http://merle.felk.cvut.cz/redmine/projects/agentpolis/wiki/Public_Transport)>.
- [5] AgentPolis simulation framework, May 2012. <<http://merle.felk.cvut.cz/redmine/projects/agentpolis/wiki>>.
- [6] AgentPolis Developer's Guide and Alite description, May 2012. <[http://merle.felk.cvut.cz/redmine/projects/agentpolis/wiki/AgentPolis\\_Developers\\_Guide](http://merle.felk.cvut.cz/redmine/projects/agentpolis/wiki/AgentPolis_Developers_Guide)>.
- [7] ANDREI BORSHCHEV, A. F. From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools, 2004. <<http://www.econ.iastate.edu/tesfatsi/systemdyndiscreteeventabmcompared.borshchevfilippov04.pdf>>.
- [8] ANDREY GLASCHENKO, G. R. P. S. A. I. Multi-Agent Real Time Scheduling System for Taxi Companies, 2009. <[http://www.ifaamas.org/Proceedings/aamas09/pdf/03\\_Industrial\\_Track/13\\_70\\_it.pdf](http://www.ifaamas.org/Proceedings/aamas09/pdf/03_Industrial_Track/13_70_it.pdf)>.
- [9] BEKTAS, T. The multiple traveling salesman problem: an overview of formulations and solution procedures, 2006. <<http://www.sciencedirect.com/science/article/pii/S0305048304001550>>.
- [10] BONABEAU, E. Agent-based modeling: Methods and techniques for simulating human systems, 2001. <<http://www.pnas.org/content/99/suppl.3/7280.long>>.
- [11] CLOUDMADE. Prague OpenStreetMap, December 2011. <[http://downloads.cloudmade.com/europe/eastern\\_europe/czech\\_republic/prague](http://downloads.cloudmade.com/europe/eastern_europe/czech_republic/prague)>.
- [12] JABER JEMAI, K. M. A Neural-Tabu Search Heuristic for the Real Time Vehicle Routing Problem, 2007. <<http://www.springerlink.com/content/g20410t4j5th214r/>>.

- [13] MAURIZIO BIELLIA, A. B. – ROSSIC, R. Trends in Models and Algorithms for Fleet Management, 2011. <<http://www.sciencedirect.com/science/article/pii/S187704281101384X>>.
- [14] NICOLAS KNAKA, B. P. S. K. An Agent-Based Simulation Tool for Modelling Sustainable Logistics Systems, 2006. <[http://www.iemss.org/iemss2006/papers/s7/225\\_Page\\_2.pdf](http://www.iemss.org/iemss2006/papers/s7/225_Page_2.pdf)>.
- [15] PAGE, B. Environmental Informatics for Sustainable Logistics, 2004. <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.4063&rep=rep1&type=pdf>>.
- [16] SNEZANA MITROVIC-MINIC, G. L. R. K. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows, 2004. <<http://www.sciencedirect.com/science/article/pii/S019126150300095X>>.
- [17] VICTOR PILLAC, A. M. C. G. Dynamic Vehicle Routing Problems: State Of The Art and Prospects, 2010. <<http://hal.archives-ouvertes.fr/hal-00623474/>>.
- [18] VOKŘÍNEK JIŘÍ, M. P. A. K. Agents Towards Vehicle Routing Problems, 2010. <[http://www.ifaamas.org/Proceedings/aamas2010/pdf/01FullPapers/16\\_03\\_FP\\_0805.pdf](http://www.ifaamas.org/Proceedings/aamas2010/pdf/01FullPapers/16_03_FP_0805.pdf)>.
- [19] ZAFEER ALIBHAI, B. What is contract net interaction protocol, 2003. <<http://www2.ensc.sfu.ca/research/iDEA/courses/files/Contractotocol1.pdf>>.

# Appendix A

## List of abbreviations

**ABM** Agent-based modelling

**CPU** Central processing unit

**csv** Comma-separated values

**GPS** Global Positioning System

**latlon** Latitude and longitude

**mTSP** Multiple Traveling Salesman Problem

**RFID** Radio Frequency Identification

**VRP** Vehicle Routing Problem





# Appendix B

## User manual

This section describes file structure, configuration and usage of executable files in the project. Two executable files are present, while Google Earth used for visualization has to be installed separately. Their source codes can be found on the included cd or in the following repositories:

```
https://bitbucket.org/staryjir/diplomka  
https://bitbucket.org/staryjir/diplomka-launcher
```

### B.1 Simulation

To run a single simulation execute following command

```
java -jar parcellogistics-jar-with-dependencies.jar [experimentId] [iterationId]
```

where *experimentId* is the number of the experiment to be executed and *iterationId* numbers the execution's iteration.

Appropriate directory `/experiment/exp-[experimentId]/` contains definition of the scenario to be executed. The result will be saved into `/experiment/exp-[experimentId]/results` under *iterationId* directory.

#### B.1.1 Google Earth

To view a running simulation in Google Earth application, download and install it from here [1]. Then before running the simulation `turnOnGeneratingGELinks = true` has to be set in the config settings (config.groovy) of correct experiment directory. After the simulation has started just open a .kml file in Google Earth. With the attribute `showGEDeliveryPoint = false` in the scenario settings (scenario.groovy) you can disable visualization of delivery locations.

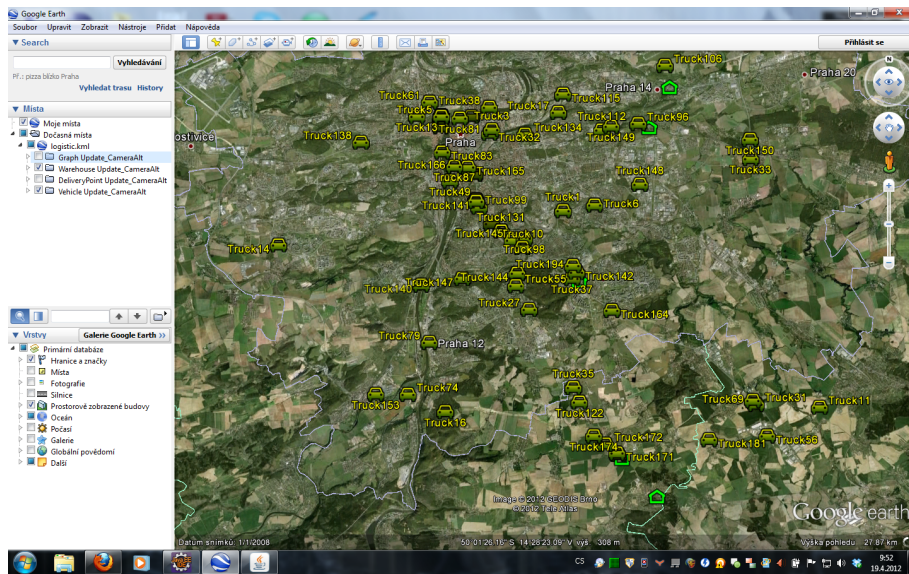


Figure B.1: Google Earth screenshot

## B.1.2 Output

Output is the standard AgentPolis event log in a csv file, beginning with a timestamp in milliseconds and the issuing agent name. For a description of events see the corresponding section (6.3).

## B.2 Launcher

Purpose of the launcher is an automated mass-execution of the simulation with different parameters. The launcher creates directory structure and configuration of each experiment based on its configuration. Every experiment is then executed. To run, just execute the presented jar file. If run from the console, you can see the redirected standard output of the simulation. Please note, that killing this process does not terminate the currently running simulation.

```
java -jar parcellogistics-scripts-jar-with-dependencies.jar
```

### B.2.1 Launcher.config

Contains all possible parameters used for creation of custom scenario. Multiple values are separated by commas. Comments are allowed only on new lines. Please note that scenario file (scenario.groovy) is created for a **cartesian product** of all possible configuration listed here and executed iterationCount times.

- iterationCount  
single number only: *iterationCount* = 10

- `numberOfVehicles`  
number of vehicles in simulation: `numberOfVehicles = 10,20`
- `vehicleCapacity`  
maximal number of deliveries in a vehicle: `vehicleCapacity = 20, 50`
- `vehicleFullLoad`  
vehicle must be fully loaded to leave warehouse: `vehicleFullLoad = false,true`
- `ensureDelivery`  
halts announcing new deliveries until older are served: `ensureDelivery = false,true`
- `deliveryFrequency`  
maximal time interval (in seconds!) that is allowed to pass before new delivery is created. Random number between (0 - `deliveryFrequency` ) is used for each new delivery creation: `deliveryFrequency = 20,100`
- `warehouseCount`  
number of warehouses on the map: `warehouseCount = 10,20`

### B.2.2 Config.groovy

while running launcher it is advised to disable visualization, so the simulation runs at full speed. settings that disable visualization:

```
showEventViewer = false
showVisio = false
turnOnGeneratingGELinks = false
```

To change simulation duration for the whole set of simulations, simply change

```
simulationDurationInMilisec = 3600000
```

to any value in milliseconds. Speed of moving vehicles can be also changed by modifying the value of `agentMoveSpeedInkmph` .

### B.2.3 Output

Apart from output of each simulation in results directory, an aggregated version in file `results.csv` is created in home directory. Each line represents one execution (iteration) of an experiment.

Outputted parameters are as follows: Simulation duration, vehicle count, vehicle capacity, full load attribute, frequency of delivery generation, warehouse count, ensure delivery attribute, average distance (m) per delivery metric, average delivery time (ms) metric, number of deliveries created during simulation, number of delivered items in simulation.



## Appendix C

# Contents of the enclosed CD

The enclosed cd contains executable binaries, results of all performed experiments, source codes of both the java projects and this text. The directory structure is as follows:

```
.
|-- binaries
|   |-- launcher                    -launcher binary and cfg
|   |   |-- experiments
|   |   |   |-- config
|   |   |   |   |-- config.groovy
|   |   |   |   |-- launcher.config
|   |   |   |-- data                -map directory
|   |   |   |-- prague.highway.osm
|   |   |-- parcel_logistics_script-0.0.1-SNAPSHOT-jar-with-dependencies.jar
|   |   |-- parcel_logistics_script-0.0.1-SNAPSHOT.jar
|   |-- simulation                  -simulation binary and config
|       |-- experiments
|           |-- data                -map directory
|           |-- exp-1                -experiment sample
|           |-- exp-2
|           |-- exp-3
|           |-- exp-4
|           |-- exp-5
|           |-- parcellogistics-1.0.0-jar-with-dependencies.jar - simulation binary
|           |-- parcellogistics-1.0.0.jar
|-- readme.txt                      -readme & installation notes
|-- results                          -all simulation results
|   |-- messages.ods                -message count measurement
|   |-- real bid - run1              -real distance bid cost
|   |-- real bid - run2
|   |-- results1-18.csv              -results of all runs together
|   |-- run1                         -one run of a launcher
|   |-- ...
```

```

|-- run18
|   |-- config                               -launcher config for the run
|   |   |-- config.groovy
|   |   |-- launcher.config
|   |   |-- exp-0
|   |   |   |-- config                       -definition of an experiment
|   |   |   |   |-- config.groovy
|   |   |   |   |-- scenario.groovy
|   |   |   |-- results                       -results of 10 iteration
|   |   |   |   |-- 0
|   |   |-- ...
|   |   |   |-- 9
|   |   |   |-- logistic.csv                 -result of a single iteration
|   |-- results.csv                           -result of the launcher run
|-- sources
|   |-- java                                   -source codes of both projects
|   |   |-- parcel_logistics                 -simulation source codes
|   |   |   |-- experiments
|   |   |   |   |-- data                     -map definitions
|   |   |   |   |   |-- dejvice_highway.osm   and extraction scripts
|   |   |   |   |   |-- demo.osm
|   |   |   |   |   |-- prague_highway_half.osm
|   |   |   |   |   |-- prague_highway_original.osm
|   |   |   |   |   |-- prague_highway.osm
|   |   |   |   |   |-- prepare_highway_half.bat
|   |   |   |   |   |-- prepare_highway.bat
|   |   |   |   |-- prepare_data_dejvice.bat
|   |   |   |-- exp-1
|   |   |   |-- exp-2
|   |   |   |-- exp-3
|   |   |   |-- exp-4
|   |   |   |-- exp-5
|   |   |-- pom.xml
|   |-- parcel_logistics_script              -launcher source codes
|   |   |-- experiments
|   |   |   |-- config
|   |   |   |   |-- config.groovy
|   |   |   |   |-- launcher.config
|   |   |   |-- data
|   |   |   |   |-- prague_highway.osm
|   |   |-- pom.xml
|   |   |-- src
|   |-- workspace.zip                         -eclipse workspace snapshot
|-- text                                       -sources for the text
|   |-- csplainnat.bst
|   |-- figures                               -images and figures

```

```
|      |-- hyphen.tex
|      |-- spreadsheet.ods          -calculations for the graphs
|      |-- stary-thesis-2012.tex    -source for this text
|      |-- thesis_reference.bib     -references
|      '-- zadani                   -scanned thesis assignment
|-- text
|  '-- stary-thesis-2012.pdf        -this text
```

2513 directories, 2439 files