Simulating Dynamic Vehicle Routing Problems with Athos

Benjamin Hoffmann, Michael Guckert,
KITE - Kompetenzzentrum für Informationstechnologie
Technische Hochschule Mittelhessen, Germany
{benjamin.hoffmann,michael.guckert}@mnd.thm.de

Kevin Chalmers,
Neil Urquhart
School of Computing
Edinburgh Napier University, Scotland
{k.chalmers,n.urquhart}@napier.ac.uk

KEYWORDS

Domain-specific language, Agent-based modelling, Evolutionary Algorithms, Vehicle Routing With Time Windows.

ABSTRACT

Complex routing problems, such as vehicle routing problems with additional constraints, are both hard to solve and hard to express in a form that is accessible to the human expert and at the same time processible by a computer system that is supposed to produce a solution of sufficient quality. The formulation must be formal enough to avoid ambiguities and also comprehensible enough to be created, discussed and shared by domain experts. In this paper, we present the domain specific language Athos in which complex routing problems can be expressed in a computationally independent, human-readable form. Athos is then transformed into code that can be run in an adequate target platform. Suitable methods for solving problems are available and applied to the given problem. We present a case study in which we use a genetic algorithm to solve instances of a vehicle routing problem with time windows and demonstrate the end to end process to produce a solution in the Athos environment. Moreover, we show how the Athos system goes beyond optimisation of static routes and can be used as a tool to simulate the impact of traffic and congestion on the tours. We call this extended problem a dynamic vehicle routing problem with time windows.

INTRODUCTION

Even after many years of research, routing problems are still a focus of current research efforts. Generally belonging to the class of NP-hard problems, the search for efficient heuristics for routing problems remains a challenge. With logistics networks growing in size and complexity, efficiency and sustainability become central issues for the transportation industry.

Rapidly changing requirements need software solutions that can be easily adapted and extended. If using general purpose languages, changing software requires systems engineers and domain experts to interact and communicate, which often is a source of misunderstanding leading to weak models and error prone systems Dalal and Chhillar 2012. A low level of abstraction in general purpose languages prevents reuse of larger building blocks and therefore implementation has to start from scratch each time a new problem has to be solved. Using a proprietary platform creates dependencies to standards imposed by a commercial software vendor that may not always be in line with the needs of the current project.

We suggest using a model driven approach to overcome this dilemma. In this paper, we discuss how domain experts can specify traffic related optimisation problems declaratively with our DSL Athos. Athos models are accessible to humans and can be transformed into executable programs to be run in appropriate target platforms (e.g. NetLogo, Repast Symphony, or potentially any other environment).

In this paper, we demonstrate Athos and its features by showing how it can be used to model both a static and a dynamised version of the Vehicle Routing Problem with Time Windows (VRPTW). In the dynamic version of the VRPTW mutual influence of traffic participants is considered. The problem runs in a network of roads defined in the model and uses a genetic algorithm to heuristically compute a solution. The models are as computationally independent as possible and therefore the algorithm and its implementation are not part of the model but of the infrastructure of the target platform.

ATHOS

Athos is a Domain Specific Language (DSL) designed for the domain of dynamic transportation problems. The language is implemented in Xtext[1]. The Athos architecture contains a generator that transforms Platform-Independent Models (PIMs) into Platform-Specific Models (PSMs); i.e. executable code to be run in a target platform. By using appropriate template code, any agent based platform can theoretically be generated. We currently support NetLogo and use it as our main development environment. Prototype support for Repast Simphony has also been implemented and tested. Code of an algorithmic nature (e.g. routing algorithms and heuristics for optimisation) is implemented in Java and made available to the target platform via libraries. The NetLogo implementation, for instance, uses extensions that can be accessed from the generated NetLogo models via the NetLogo-Extension-API. Athos deliberately does not support imperative programming as we aim to provide a purely declarative modelling language. Currently available

[1] https://www.eclipse.org/Xtext/

meta-heuristics libraries in Athos are an implementation of an ACS (Ant Colony System) for solving TSP-like problems Hoffmann, Guckert, et al. 2018 and an Evolutionary Algorithm for solving more complex vehicle routing problems like the problem presented in this paper..

The VRPTW describes the task of visiting a set of customers for delivery or pick-up of products, depending on context. Each visit must conform to time windows specified for each customer. If the vehicle does not arrive within the limits of the time window, it either has to wait until the window opens (early arrival) or the schedule is not feasible (late arrival). Visits at a customer may also consume a given amount of service time. Vehicles start and end their journeys in one of possibly many depots. If more than one depot is used, the problem is referred to as a multi-depot problem. The problem may be formulated for a fixed number of vehicles or the number of vehicles may be part of the objective function of the optimisation problem. VRPTW can be used to optimise a single objective (e.g. overall distance travelled) or multiple objectives such as vehicles and distance travelled Dabia, Demir, and Woensel, van 2014 or vehicles, distance travelled, financial costs and environmental impact (CO_2-production). VRPTW is highly-relevant to real-world problems both in an operational N. B. Urquhart, Hart, and Judson 2015 and a planing context N. Urquhart and Fonzone 2017. The user of the VRPTW-solving software will be a domain expert, e.g. a logistics analyst or transport planner. Many industrial contexts will include additional specific constraints defined by the business context, i.e. working conditions of staff, types of vehicle in use, environmental or financial considerations. The implementation of such additional, possibly very versatile, constraints leads to an increased workload for professional programmers using a conventional, general-purpose programming language. A DSL as Athos will potentially reduce development time and give domain experts a tool to easily modify and extend a model without having to access software developers.

Obviously, any problem instance of a VRPTW requires travel time data between customers and depots. Depending on the optimisation criterion used, it may also be necessary to retain distance or emissions data as well. Such data can be held in an origin-destination matrix (see for example Dantzig, Ramser, and Hubert 1959), or simply be calculated using the Euclidean distance between customers. However, real-world examples N. Urquhart and Fonzone 2017; N. B. Urquhart, Hart, and Judson 2015 usually use an underlying street graph and apply path-finding algorithms to find routes between customers.

Summarising the discussion, we see that a DSL supporting the domain of vehicle routing must be capable of expressing the many different formulations of a VRP including differing vehicle types, time windows, service times, capacity constraints and driving time constraints. At the same time, it must also support the modelling of the underlying street graph.

The following step by step example illustrates the modelling features of the language and how a VRPTW can be described. The product to be delivered is soap each item having a weight of 1 weight unit. The model contains a single agent type named *delivery* with two states *awt* and *die*. The agents of this type wait for an optimal tour to be computed and then start delivering goods according to the tour received thereby satisfying demands of the customers on that tour. Travel time will be computed using the default duration function that uses length and speed of the agents and counts in the defined congestion factors so that a high amount of traffic in the network has an impact on the delivery.

```
model VRPTW_Example
 world xmin 0 xmax 75 ymin 0 ymax 75
 products product soap weight 1.0
 agentTypes
 agentType staticDelivery congestionFactor 60.0 maxWeight 200.0
    behaviour awt awaitTour when finished do die;
    behaviour die vanish;
 functions
 durationFunction normal  length default
```

Behavioural patterns of agents are defined by means of *behaviour blocks.* For each agent type a single behaviour block with an arbitrary number of *behaviour states* can be defined. These behaviour states correspond to the states of an implicit finite automaton. A state consists of an possibly externally visible action of the agent and an internal representation of transitions and their effect. Events can be defines as stimuli that trigger a transition and entail a change of state.

The Athos meta-model reflects all of the elements of the language. Figure 1 shows how agent states are represented there. *AgentTypes* are each linked to exactly one *AgentBehaviourBlock*, which contains one or more *AgentBehaviourStates*. The state of an agent corresponds to exactly one observable behaviour of the agent that the agent produces when being in the respective state. This observable behaviour is an instance of *AgentBehaviourDescription*. Athos has a set of built-in *AgentBehaviourDescriptions* which will continuously be extended in future versions.

Additionally, an *AgentBehaviourState* contains an arbitrary number of *AgentBehaviourTransitions*, which trigger a change to a target state depending on a condition. Possible target states are other named states in the *AgentBehaviourBlock* (in Figure 1 or anonymous states defined in the *AgentBehaviourTransition*.

```
agentTypes
agentType staticDelivery congestionFactor 60.0 maxWeight 200.0
    behaviour awt awaitTour when finished do die;
    behaviour die vanish;
```

The underlying network here is a complete graph in which the length of the edges equals the Euclidean distance between the end nodes. Nodes are defined with their coordinates. As the graph is defined to be complete, the list of edges is empty. The length of the edges is used as travel time. Note that this is only an academic example – any other duration function could be defined and used here.

```
complete network
 nodes
 node n1 (35.0, 35.0)
 ...
 node n51 (47.00, 47.00)
edges
```

Sources and demands are defined using nodes of the network. The keyword *ea* indicates that an evolutionary algorithm is to be used to compute the tours for the vehicles leaving the

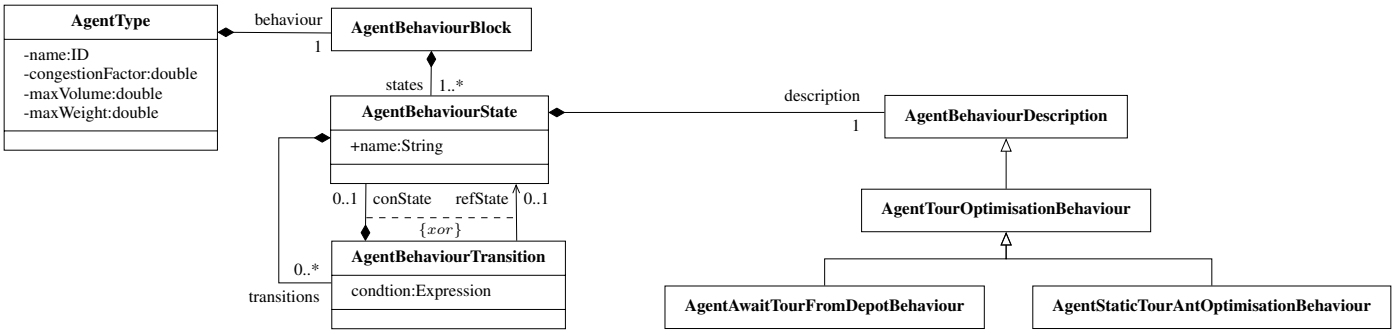


Fig. 1. Athos meta-model for modelling agent behaviour.

depots. Additional parameters for the algorithm are provided. For each demand quantities, time windows, and service time are defined.

```
sources
    n1 isDepot soap sprouts (staticDelivery)   agentsStart route
     ea (n2, n3, n4, ..., n48, n49, n50, n51) popSize 30
demands
     n1 hasDemand soap absQuantity 0.00 earliestTime  0
        latestTime 230 serviceTime 0
     ...
     n51 hasDemand  soap absQuantity 13.00 earliestTime 124
        latestTime 134 serviceTime 10
```

Agents of a given type can be monitored by using metrics in which indicators can be defined that can either accumulate values or set values. These indicators are collected for each agent and can be viewed for each single agent or condensed into overall statistics. As any other feature of Athos, metrics have a representation in the meta model analogous to that of the behaviours. However, for the sake of brevity this is not discussed here.

```
defineMetrics
    metrics for staticDelivery (
        class metric distanceCovered
            when (isAtCustomer?)
            add distanceTo last customer
        individual metric ticksEarly
            when (isAtCustomer? && earliestTime > currentTime)
            add earliestTime − currentTime
    )
```

Beyond the optimisation of tours with the built in evolutionary algorithm, Athos can run simulations of dynamic delivery scenarios with these optimised schedules in which the effect of traffic and congestion in the network can be measured up with the individually defined metrics. While a static VRPTW optimises tours once according to a defined objective function, a dynamic VRPTW goes beyond that and is sensitive to dynamic aspects of traffic in the underlying network. We discuss and compare a static and a dynamic VRPTW in the case study presented in the next section.

CASE STUDY

In this section, we will present a case study that compares the static and dynamic variants of the VRPTW problem. We will analyse how a changed traffic situation influences the success of the planned tours. First we will use Athos to define a static VRPTW. In addition, we will define some metrics to see how the vehicles perform when the traffic situation stays exactly the same throughout the entire simulation. In a second step, we will transform the VRPTW into a dynamic VRPTW by adding additional noise-agents that will induce congestion effects inside the network. We will use the defined metrics to see how congestion influenced the outcome of the calculated VRPTW solution. Note that the complete Athos program, the generated NetLogo programs (together with the required extension) as well as some videos showing the simulation can be obtained from https://athos.mnd.thm.de/public/ecmscasestudy.html.

Figure 2 illustrates the graph used in this case study. The graph is an artificially simplified version of an urban area with the following characteristics:

- At its core, the area features a beltway. The roads here are highly dependent on each other so that congestion on one road directly expands to other roads of the beltway.
- The depot is located at the very centre of this beltway. Access roads to the depot also belong to the beltway and thus are also affected by any congestion on the beltway.
- The centre of this area also features a network of highways with high capacity. These highways are independent so that congestion on one highway does not have ripple effects on adjacent highways.
- Suburban areas are accessible through roads of less capacity. These roads are more susceptible to congestion when used by a queue of cars or cars with high congestion factors like heavy-goods vehicles.

```
model UrbanArea
world xmin 0 xmax 40 ymin 0 ymax 22
<<definition of agent types>>
functions
durationFunction highway
    length + 1.5 * accCongestionFactor default
durationFunction road length * 3 + 4 * accCongestionFactor
network
<<definition of nodes>>
edge undirected e01 from n0 to n1
    length 0.0 cfactor 1.0 path "cityRing" function highway
edge undirected e12 from n1 to n2
    length 0.0 cfactor 1.0 path "cityRing" function highway
edge undirected e16 from n10 to n11
    length 0.0 cfactor 1.0 function highway
edge undirected e17 from n10 to n18
    length 0.0 cfactor 1.0 function road
<< definition of other edges >>
```

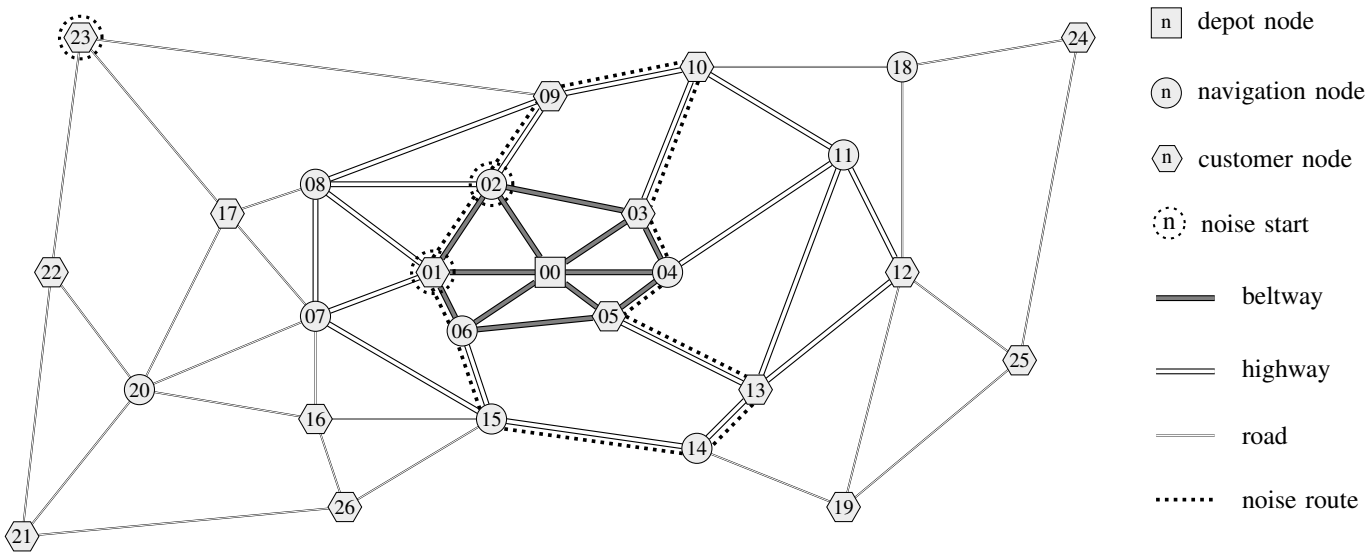


Fig. 2. Artificial graph used in the case study.

The first two lines of the listing above give a name to the model and define its global boundaries. The definition of the agent types used in the case study is omitted in the listing but will be presented shortly. Lines 4 to 7 contain the functions that are associated to the two different road types used in the case study. The function associated to edges which represent highways was aptly named "highway". The meaning of this function is that the length of a highway defines the minimum amount of time a vehicle needs to cross it. For example, a highway of length 200 would require 200 ticks to be crossed by a vehicle, given that the vehicle has a congestion factor of zero and the summed up congestion factor of all other vehicles on the road is also zero. This is due to the fact that the congestion factor of each vehicle on a road additionally increases the time required to cross the respective road.

Roads that belong to the same path share their accumulated congestion factor. In the listing, edge `e01` and `e12` belong to the same path. To account for the fact that normal roads take longer to travel and are more susceptible to congestion than highways, the `road` function multiplies the length of the road by three and the accumulated congestion factors by four.

```
1  << environment definition >>
2  agentTypes
3  agentType delivery congestionFactor 0.0 maxWeight 180.0
4      behaviour awt awaitTour when finished do wait;
5      behaviour wait idle for 1000 when finished do die;
6      behaviour die vanish;
7  <<network definition>>
8  sources
9   n0 isDepot soap sprouts (delivery)
10      agentsStart route ea (n3, n5, n9, n10, n12, n13, n16, n17,
11      n19, n21, n22, n23, n24, n25, n26) popSize 30 at 0
12 demands
13   n0 hasDemand soap  absQuantity 0.00
14      earliestTime 0 latestTime 500 serviceTime 0
15   n1 hasDemand soap  absQuantity 15.0
16      earliestTime 15 latestTime 120 serviceTime 5
17 << more demand specifications >>
```

The above listing shows the specification of the homogeneous fleet of vehicles that can be sent out by the depot. Lines 3 to 6 specify that each of those vehicles has a congestion factor of zero. This was done so that these vehicles do not contribute to congestion on the roads. Each vehicle can provide a customer with 180 units of a given product. Line 4 specifies that this type of vehicle waits at a depot until it is assigned a tour. As the metrics that we will define in the next step do only apply for existing agents, line 5 and 6 tell a vehicle to idle at the depot for 1,000 ticks before leaving the simulation. In line 9, node `n0` is defined to be a depot from where vehicles of type `delivery` start. Customers to be served are defined inside the brackets. Note that in this case, the code is intentionally no longer computationally independent, because the `ea` keyword explicitly specifies the application of an evolutionary algorithm to solve the problem. This allows to further specify certain parameters used in the respective algorithm – in this case, the population is set to a size of 30. The depot is supposed to find a set of tours for its vehicles at the very beginning of the simulation. This is specified in line 11 with the keyword `at` followed by the value zero. Finally, lines 12 to 17 contain the demand specifications. Note that the depot node also appears here in order to define the latest point in time for the return of the vehicles. Table I summarises the constraints related to the customers in this case study.

```
1  << environment and delivery agent type >>
2  agentType noiseAgent congestionFactor 150.0 maxWeight 100.0
3   behaviour roam route exact (n1, n2, n9, n10, n3, n4, n5,
4                               n13, n14, n15, n6) repeat 5 times
5     when finished do die;
6   behaviour die vanish;
7  << functions and network definition >>
8  sources
9  << n0 still depot >>
10      n1 sprouts (noiseAgent) at 2
11      n2 sprouts (noiseAgent) at 4
12      n23 sprouts (noiseAgent) at 3
```

The Athos code presented so far allows us to run the simulation in a "noise-free" mode: The depot calculates a solution for the VRPTW and sends out the vehicles to serve the customers. In order to transform the static VRPTW into a dynamic version, we will run a second batch of simulations in which additional noise-agents will increase travel times on the city ring and some of the roads/highways in the network. To this end, the above listing introduces another agent type with a congestion factor of 150. Due to its high congestion factor, this type of agent considerably slows down traffic on its current road (or path of roads). The agent follows a predefined route (specified in lines 3 and 4) 5 times and then disappears. Lines 9 to 11 define the nodes and the exact point in time where an instance of this type of agent appears. Note that even though node `n23` is not part of the specified route of nodes for that type of agent, it can still appear at this node. The agent then uses the fastest way to the first node of the tour specified for this agent type (`n1`).

The final part of the simulation specification for this case study is the definition of a set of metrics. In this case study, we are interested in the cumulative distance travelled by the delivery vehicles. Moreover, the exact time vehicles had to wait due to an early arrival at a customer might give some insight on the efficiency of the calculated tour. Also, it might be important to now the accumulated time by which vehicles arrived to late at a customer and the total number of time windows violated and time windows met. These metrics are specified in the following listing.

```
1  << as before >>
2  defineMetrics updateRate 10
3      metrics for delivery  (
4          class metric distanceCovered
5              when ( isAtCustomer? )
6              add distanceTo last customer
7          class metric ticksEarly
8              when (isAtCustomer? && earliestTime > currentTime)
9              add earliestTime - currentTime
10         class metric ticksLate
11             when (isAtCustomer? && latestTime < currentTime)
12             add currentTime - latestTime
13         class metric windowsViolated
14             when (isAtCustomer? && latestTime < currentTime)
15             add 1
16         class metric windowsMet
17             when (isAtCustomer? && currentTime <= latestTime)
18             add 1
19     )
```

Table II and Table III summarise the results of ten simulation runs each for the problem without noise agents and the problem with noise agents that introduce dynamism through reduced travel speeds. As was to be expected, in the simulations without noise-agents, no time windows were violated. In addition to the 16 customers defined for the problem, the metric also counts the timely return of a vehicle to the depot as a met time window. Since the evolutionary algorithm used to solve this problem is not deterministic, some runs feature solutions with three and some with four vehicles resulting in 19 or 20 met time windows.

The introduction of noise-agents changes the situation dramatically. The noise agents effectuate the movement speed on their respective roads in a way that the delivery vehicles do no longer meet all time windows. In fact, nearly half of the defined time windows are violated. In each of the ten simulations with noise-agents the accumulated ticks by which time windows were missed is around 760.

In both cases the distance travelled by the vehicles was nearly which was to be expected. The slower movement on the roads caused the vehicles to arrive at their customers later than originally calculated which is also reflected in the amount of ticks that vehicles arrived too early at their customers which is reduced by around 6.7 ticks.

In our future work, we will use Athos to further research into dynamic VRPTW to provide strategies that provide satisfactory solutions even in case of sudden traffic surges.

TABLE I. CONSTRAINTS OF THE VRPTW.

Cstm	Location	Demand	Earliest	Latest	Service
01	(18.0, 9.0)	15	15	120	5
03	(21.0, 11.0)	20	10	120	7
05	(20.0, 7.5)	50	20	90	10
09	(18.0, 15.0)	45	80	220	15
10	(23.0, 16.0)	25	90	250	10
12	(30.0, 9.0)	30	35	260	5
13	(25.0, 5.0)	60	40	140	7
16	(10.0, 4.0)	35	45	160	8
17	(7.0, 11.0)	40	0	140	50
19	(28.0, 1.0)	15	10	130	9
21	(0.0, 0.0)	50	5	120	5
22	(1.0, 9.0)	30	10	90	15
23	(2.0, 17.0)	40	20	60	5
24	(36.0, 17.0)	35	25	50	8
25	(34.0, 6.0)	20	20	60	9
26	(11.0, 1.0)	15	30	45	10

RELATED WORK

Steil et al. (Steil et al. 2011) discuss an approach that encompasses all aspects involved in the domain of patrol routing algorithms. It covers all stages in the development of patrol routes from the specification (expression) and simulation-based assessment (execution and evaluation) to the translation of patrol routes to the real-world (engagement). Accordingly, they call their approach the 4Es approach.

The 4Es approach is similar to the approach presented in this paper in that it integrates the expression, simulation and evaluation in an appropriate environment. Moreover, it also uses a DSL for the expression of routing algorithms. Their DSL, called Turn, allows to define the next destination of an agent in a road network by means of set reduce functions that can be chained to successively reduce the set of all nodes until only one node is left which is then selected as an agent's next destination. This way the agents in the simulation follow a pre-defined routing strategy. The system evaluates routing strategies by application of four distinct metrics. These metrics provide information on the time it took a first-responding agent to get to the node where an event occurred, the percentage of

TABLE II. RESULTS FOR TEN RUNS OF A NOISE-FREE SIMULATION.

Metric/Run	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg.**
Total distance	101.90	105.10	105.10	101.90	101.90	107.65	102.79	107.65	101.90	107.65	104.35
Ticks early	27	48	48	27	27	27	25	27	27	27	31
Ticks late	0	0	0	0	0	0	0	0	0	0	0
Windows violated	0	0	0	0	0	0	0	0	0	0	0
Windows met	19	20	20	19	19	19	20	20	19	20	19.5

TABLE III. RESULTS FOR TEN RUNS OF A NOISE-FULL SIMULATION.

Metric/Run	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg.**
Total distance	104.86	101.71	101.71	101.71	101.71	101.71	101.71	104.86	107.42	104.86	103.23
Ticks early	39	18	18	18	18	18	18	39	18	39	24.30
Ticks late	766	764	766	762	760	764	761	764	776	763	764.6
Windows violated	9	9	9	9	9	9	9	9	10	9	9.1
Windows met	11	10	10	10	10	10	10	11	10	11	10.30

nodes in the network visited by agents per day, the number of nodes visited that were in a state in which an event of interest is likely to occur (so-called hot nodes), an the amount of time agents spent at such hot nodes.

Despite the mentioned similarities, the work of Steil et al. differs considerably from our work when looked at in more detail. First of all, the two research efforts target two different domains. As is pointed out by the authors, patrol routing problems share some features with vehicle-routing problems but greatly differ in what practitioners ultimately aim to achieve. In VRPs, in the majority of cases, the objective is the minimisation of a given cost function. By contrast, patrol routing dispatchers often do not search for a solution that optimises any specific value. Instead they search for routes that bring about satisfactory values for certain metrics as long as the routes followed by the agents are somewhat unpredictable and are non-deterministic. In contrast to the Turn DSL, Athos allows an explicit definition of a list of nodes to visit in the exact specified order or the definition of a set of nodes which have to be visited in an order that optimises a user-defined function. Most importantly, in the simulation framework of Steil et al., there is no concept of velocity or congestion. Agents move along the underlying graph among neighbouring nodes one node per time step. Due to the fact that there are no congestion effects or any changes in the movement speeds of agents, their approach cannot be used to simulate dynamic vehicle routing problems where travel times are subject to fluctuations depending on the current traffic situation.

Another platform for traffic and transport simulations is MATSim (Horni, Nagel, and Axhausen 2016). MATSim is a mutli-agent microsimulation system based on the co-evolutionary principle. This means that the agents in the system are equipped with a set of plans to follow. Throughout multiple iterations the agents try and evaluate the outcome of the plans at their disposal. In each cycle a plan is selected, applied and evaluated. With a given probability, agents modify different dimensions of their plans. For example, agents can vary the time they leave a given location, choose a different route or switch to a different mode of transport. Each agent seeks to optimise its individual outcome.

Macijewski and Nagel present a MATSim-based approach to evaluate algorithms for the DVRP (Maciejewski and Nagel 2012). At the same time, their work intends to plan demand-responsive transport services (DRT severcies) using the MATSim framework. In their approach MATSim is used to calculate time-dependant travel times for a given scenario. The calculated data is then merged with additional demand and supply data which results in a (dynamic) VRPTW. This way, the authors designed four scenarios. The scenarios were designed to closely resemble traffic situations common to urban environments. Two of the scenarios analysed courier services while the other two scenarios investigated taxi services. They calculated solutions in two ways: First, they solved the problem using time-dependant travel times. Second, they produced solutions based on average travel times. The latter results were then applied to the problem with time-dependant travel times. The authors found that for the couriers services the routes calculated on the basis of the time dependent travel times considerably outperformed those based on average travel times. Their explanation is that knowledge of time-dependant travel times allows for the calculation of routes that avoid congested roads. Interestingly, for the taxi services travel times for the solutions of both approaches were rather similar which the authors explain with the nature of the demands to taxi-services which does not lend itself to careful planning. For both scenarios, the solutions based on average times violated the defined time windows when applied to the problem that featured time-dependent travel times.

GAMA (Grignard et al. 2013) is a sophisticated simulation-platform. It features the GAML DSL which was designed for modelling agent-based simulations. The DSL's meta-model elements can be divided into elements for the definition of aspects related to entities, space and time. A species element is to GAML what a class is to object-oriented languages like Java. A distinct feature of the GAML meta-model is that it allows agents to form containment hierarchies that can be

used to model different levels of detail in a simulation. The species definition is also used to equip agents with skills like movement and attributes like movement speed. The species of an agent also defines a set of actions and reflexes. Actions represent behaviour that an agent executes when asked to. By contrast, a reflex represents behaviour that the agent executes in every step of the simulation given that all guard conditions hold. Even though GAML could also be used to model transport and routing problems, it requires modelling in a language not specifically tailored towards this domain. Thus, models are on a less abstract level which makes harder to comprehend and communicate by domain experts.

CONCLUSIONS AND FUTURE WORK

We have demonstrated how Athos can be used to model dynamic vehicle routing problems and how solutions can be computed. Besides the evolutionary algorithm presented here Athos provides other heuristics for solving a variety of complex routing problems (see Hoffmann, Guckert, et al. 2018 and Hoffmann, Chalmers, et al. 2019).

At the moment we extend the Athos environment with a flexible interface to Open Street Map [2] so that the definition of the underlying network can be generated from OSM data rather than be coded manually. Beyond that our concern is to measure general usability aspects of the language by letting domain experts assess the applicability of Athos. While we currently aim at improving the modeling capabilities of Athos our long-term intention is to develop an integrated instrument that allows a domain expert to describe and solve real world traffic related routing problems without any need for algorithmic decisions. The system will choose appropriate methods and heuristics and generate efficient best-practice code for the target platform.

REFERENCES

Dabia, S., E. Demir, and T. Woensel, van (2014). *An exact approach for the pollution-routing problem*. English. BETA publicatie : working papers. Technische Universiteit Eindhoven.

Dalal, Sandeep and Rajender Singh Chhillar (2012). "Case Studies of Most Common and Severe Types of Software System Failure". In: *International Journal of Advanced Research in Computer Science and Software Engineering* 2.8, pp. 341–347.

Dantzig, George, Bernard Ramser, and John Hubert (1959). "The Truck Dispatching Problem". In: *Management Science* 6.1, pp. 80–91.

Grignard, Arnaud et al. (2013). "GAMA 1.6: Advancing the art of complex agent-based modeling and simulation". In: *International Conference on Principles and Practice of Multi-Agent Systems*, pp. 117–131.

Hoffmann, Benjamin, Kevin Chalmers, et al. (2019). "Athos - A Model Driven Approach to Describe and Solve Optimisation Problems: An Application to the Vehicle Routing Problem with Time Windows". In: *Proceedings of the 4th ACM International Workshop on Real World Domain Specific Languages*. RWDSL '19. Washington D. C., DC, USA: ACM, 3:1–3:10.

[2] www.openstreetmap.org

Hoffmann, Benjamin, Michael Guckert, et al. (2018). "A Domain-Specific Language For Routing Problems". In: *European Conference on Modelling and Simulation, ECMS 2018, Wilhelmshaven, Germany, May 22-25, 2018, Proceedings*, pp. 262–268.

Horni, Andreas, Kai Nagel, and Kay W. Axhausen (2016). "Introducing MATSim". In: *The Multi-Agent Transport Simulation MATSim*. Ed. by Andreas Horni, Kai Nagel, and Kay W. Axhausen. Ubiquity Press, pp. 3–8.

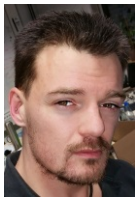
Maciejewski, Michał and Kai Nagel (2012). "Towards Multi-Agent Simulation of the Dynamic Vehicle Routing Problem in MATSim". In: *Parallel Processing and Applied Mathematics*. Ed. by Roman Wyrzykowski et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 551–560.

Steil, Dana A. et al. (2011). "Patrol Routing Expression, Execution, Evaluation, and Engagement". In: *IEEE Transactions on Intelligent Transportation Systems* 12.1, pp. 58–72.

Urquhart, Neil B., Emma Hart, and Alistair Judson (2015). "Multi-Modal Employee Routing with Time Windows in an Urban Environment". In: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, pp. 1503–1504.

Urquhart, Neil and Achille Fonzone (2017). "Evolving Solution Choice and Decision Support for a Real-world Optimisation Problem". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '17. ACM, pp. 1264–1271.

AUTHOR BIOGRAPHIES



BENJAMIN HOFFMANN is a research assistant at Technische Hochschule Mittelhessen in Friedberg from which he also received his master's degree. He is also a PhD student at Edinburgh Napier University. His research activities are in domain-specific languages, model-driven software development and optimisation problems.

MICHAEL GUCKERT is a Professor of Applied Informatics at Technische Hochschule Mittelhessen and head of KITE - AACC (Kompetenzzentrum für Informationstechnologie - Advanced Analytics Cognitive Computing) . He received a degree in Mathematics from Justus Liebig University Giessen and a PhD in Computer Science from Philipps University Marburg. His research areas are multi agent systems, model driven software development and applications of artificial intelligence.

KEVIN CHALMERS is a Associate Professor at Edinburgh Napier University where he leads the Computer Science and Software Engineering subject area. He gained his PhD from Edinburgh Napier University in 2009, examining the application of mobile concurrency models to ubiquitous computing. His research is focused primarily on concurrency and parallelism and how different technologies can support this.

NEIL URQUHART is a lecturer in Computing Science at Edinburgh Napier University where he is Programme Leader for the Computing Science. He gained is PhD from Edinburgh Napier University in 2002, writing a thesis examining the use of Software Agents and Evolutionary Algorithms to solve a real-world routing optimisation problem. His research interests include Evolutionary Computation and Agent-based Systems and their application to real-world optimisation problems