HAL open science

SULFR: Simulation of Urban Logistic For Reinforcement

Guillaume Bono, Jilles Dibangoye, Laëtitia Matignon, Florian Pereyron, Olivier Simonin

To cite this version:

Guillaume Bono, Jilles Dibangoye, Laëtitia Matignon, Florian Pereyron, Olivier Simonin. SULFR: Simulation of Urban Logistic For Reinforcement. Workshop on Prediction and Generative Modeling in Reinforcement Learning, Jul 2018, Stockholm, Sweden. pp.1-5. hal-01847773v2

HAL Id: hal-01847773

https://inria.hal.science/hal-01847773v2

Submitted on 7 Sep 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SULFR: Simulation of Urban Logistics for Reinforcement

Guillaume Bono [1] **Jilles Steeve Dibangoye** [1] **Laëtitia Matignon** [1 2] **Florian Pereyron** [3] **Olivier Simonin** [1]

Abstract

In urban logistics, various sources of uncertainty can invalidate pre-planned routes. In this context, a routing strategy that uses available information from the environment could help improve the overall performance of the routing process by dynamically choosing the next client at the online execution time. While static and deterministic testbeds for vehicle routing exist, their stochastic and dynamic counterparts are still missing. This paper proposes an interface to the micro-traffic simulation package *SUMO* that implement a generative model of stochastic and dynamic vehicle routing problems. We formalize the latter using a reinforcement learning framework for semi-Markov decision processes. The resulting testbeds make it possible to compare single- and multi-agent reinforcement learning algorithms in customizable routing environments. We report our preliminary tests to evaluate a hand-crafted policy on some basic scenarios.

1. Introduction

In urban logistics, many professionals could testify that pre-compiled plans for vehicle routing can rarely be followed by the book all day long. Many sources of uncertainty can invalidate these plans – examples including unpredictable traffic conditions, lack of parking areas, unavailable clients, or unexpected packaging. Drivers usually adapt and repair plans using their own experience to find the best compromises. For the drivers to act conditionally on the available information at the online execution time, it is crucial that they learn upon information collected from both individual and collective experiences. The literature of single- and multi-agent reinforcement learning (Sutton & Barto, 1998) addresses such problems through repeated interactions between automated learning agents and an environment; the later often serves as a generative model of experiences. While static and deterministic environments for vehicle routing problems exist, their stochastic and dynamic counterparts are still missing.

The literature of generative models for the stochastic and dynamic vehicle routing problems is still in its infancy. Many models are either handcrafted (Cordeau & Laporte, 2003; Novoa & Storer, 2009; Parragh et al., 2010) or built upon static and deterministic artificial scenarios (Solomon, 1987). These models mimic stochastic and dynamic environments either by adding random noise to some features of the scenario, such as travel times, or clients demand, or by sampling events, e.g., clients requests, from an arbitrary distribution. Other models rely on databases of historical traces from logistic companies, railroad, taxis, or healthcare transport services (Simão et al., 2009; Bouzaiene-Ayari et al., 2016; Schilde et al., 2014; Chen et al., 2016). Recently, the *Flow* framework developed by (Wu et al., 2017) provides an interface that allows one to run reinforcement learning algorithms using OpenAI *rllab* framework (Duan et al., 2016) to control, at a low level, autonomous vehicles deployed within the micro-traffic simulator *SUMO* (Krajzewicz et al., 2012). This generative model is close to ours but focuses only on traffic flows' regulation.

This paper presents a novel interface to single- and multi-agent reinforcement learning for Stochastic and Dynamic Vehicle Routing Problems (SD-VRPs). Our *SULFR* interface builds upon the Simulation for Urban MObility (SUMO) to recast SD-VRPs into Semi-Markov Decision Processes (SMDPs) (Sutton et al., 1999). It aims at providing a generative model to sample trajectories of states and rewards in customizable scenarios, which can be imported from real-world data, or designed to highlight desirable properties for a routing policy.

Prior attempts to formalize vehicle routing problems into MDPs (Coltin & Veloso, 2014; Maxwell et al., 2010; Ichoua et al., 2006; Sáez et al., 2008) assume synchronized vehicle decisions, i.e., all vehicles' actions last a single decision step, and time elapsed in each route is stored separately for each vehicle. Unfortunately this model may raise synchronization issues when taking online decision through a stochastic

[1]Univ Lyon, INSA Lyon, INRIA, CITI, F-69621 Villeurbanne, France [2]Univ Lyon, Universit Lyon 1, LIRIS, CNRS, UMR5205, Villeurbanne, F-69622, France [3]Volvo Group, Advanced Technology and Research. Correspondence to: Guillaume Bono <guillaume.bono@inria.fr>.

Accepted at the FAIM workshop "Prediction and Generative Modeling in Reinforcement Learning", Stockholm, Sweden, 2018.
Copyright 2018 by the author(s).

policy relying on MDPs. Our generative model builds upon Semi-MDPs instead. In such a framework, agents act based on meta-actions (also called options) instead of low-level actions (Sutton et al., 1999; Mahadevan et al., 1997). We only have to consider a few options, for example "deliver a package" or "recharge the battery". We use *SUMO* as a generative model for the lower-level physics and navigation of each vehicle, which enables us to update its state and compute rewards signal designed for SD-VRPs.

We organized the remainder of this paper as follow. Section 2 presents the SMDP formalism and how we recast SD-VRPs into SMDPs. Section 3 shows the structure of our SUMO interface managing agents in customizable urban scenarios. Section 4 describes the use of this interface to evaluate hand-crafted policies. Finally, we end the paper with concluding remarks and future work.

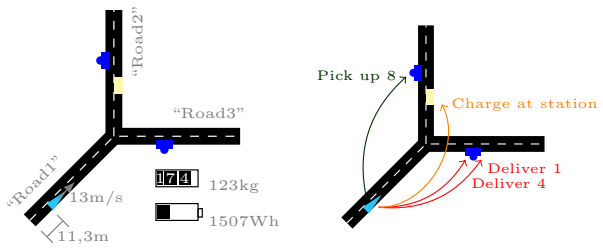

Figure 1. Best viewed in color. *Left*: An agent (cyan triangle) facing two clients' locations (blue) and a charging station (yellow). Some components of its state are highlighted in (gray). *Right*: Available options for the agent: it can deliver package 1 and 4 to the client on the right (red), pick up package 8 at the other client (green) or recharge its battery at the station (orange). Detailed course of actions to reach "Road2" or "Road3", park at the targeted stop and wait for loading, unloading or full charge is given by the low-level policies corresponding to each option.

2. Problem Formalization

2.1. Semi-Markov Decision Process

We define a Semi-Markov Decision Process as a Markov Decision Process with *options*, as introduced by (Sutton et al., 1999).

An SMDP $\mathrm{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, p, r, b_0, \gamma \rangle$ is defined by: $\mathcal{S}$ the set of possible states; $\mathcal{A}$ the set of possible atomic actions taken at each time step; $\mathcal{O}$ the set of options available, each defined as a tuple $o = \langle \mathcal{I}, \pi, \beta \rangle$, where $\mathcal{I} \subseteq \mathcal{S}$ is the subset of states in which o can be initiated, π_s^a is the probability of choosing atomic action a in state s while executing o, β_s is the probability of o terminating in state s; $p_{ss'}^a = \mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a)$ the probability of transition from state s to state s' when executing action a; $r_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$ the expected immediate reward $R_{t+1} \in \mathbb{R}$ when executing a in state s; $b_{0s} = \mathbb{P}(S_0 = s)$ the initial state distribution; $\gamma \in [0, 1[$ the discount factor applied to future rewards.

Note that every atomic action a can be viewed as an option with pure policy $\pi_s^{a'} = 1$ if $a' = a$, 0 otherwise, and termination probability $\beta_s = 1$ in all states s. We can factorize a transition rule and reward signal over options $p_{ss'}^o$ and r_s^o by marginalizing probabilities over states encountered during option execution.

We define a policy over options $\mu : (s, o) \mapsto \mathbb{P}(o|s)$ as the probability of starting option o in state s. The objective is to find the optimal policy over options μ^* maximizing expected discounted cumulated reward $\mathbb{E}[\sum_{t=0}^{T} \gamma^t R_{t+1}]$.

Next we describe how we recast the Stochastic and Dynamic Vehicle Routing Problem into a SMDP. More specifically, we consider a capacitated VRP with stochastic and dynamic pick up and deliveries, time windows, stochastic travel times and electric vehicles (Toth & Vigo, 2002).

2.2. States

First, let us introduce some useful notations: $\mathcal{E}$ is the set of edges in the graph describing the road network. L_e denotes the length of road $e \in \mathcal{E}$. Finally $T > 0$ is the time horizon of each episode.

Any state $s \in \mathcal{S}$ can be decomposed into agents' states $\langle s^1, \ldots, s^n \rangle$ and environment state s^{env}.

State s^i of agents $i \in \{1, \ldots, n\}$ consists of many attributes, including:

$$\begin{aligned}
\texttt{road} &\in \mathcal{E} && \text{: String id of current road} \\
\texttt{road_pos} &\in [0, L_{\text{road}}] && \text{: Position on current road} \\
\texttt{x, y} &\in \mathbb{R}^2 && \text{: Cartesian coordinates} \\
\texttt{speed} &\in [0, \text{max_speed}] && \text{: Speed} \\
\texttt{load} &\in [0, \text{max_load}] && \text{: Available payload capacity} \\
\texttt{battery} &\in [0, \text{max_bat}] && \text{: Battery level}
\end{aligned}$$

The environment state s^{env} contains:

$$\begin{aligned}
\texttt{step} &\in [0, T] && \text{: Simulation time step} \\
\texttt{road_tt[e]} &> 0 && \text{: Avg. travel time for all edge } e \in \mathcal{E} \\
\texttt{req_status[j]} & && \text{: Status for all requests } j \in \{1, \ldots, m\} \\
&\in \{\text{hidden}, \text{pending}, \text{assigned}, \text{served}, \text{expired}\} &&
\end{aligned}$$

A state is terminal whenever one of the vehicles' batteries fails, or if all requests are served or expired, or if the time step reaches its limit T.

2.3. Actions and Dynamics

Low-level atomic actions $a \in \mathcal{A}$ are also factored $a = \langle a^1, \ldots, a^n \rangle$. Action a^i of agent i has two components: acceleration and lane changing commands. Our approach abstracts these actions into higher-level options with predefined internal policies. We use internal policies provided by SUMO, such as one based on the stochastic car-following model introduced by (Krauß, 1998).

As for the dynamics $p^a_{ss'}$ of the process, it is hard to explicitly state them, as part of the state results from global behaviors emerging from a complex combination of local interactions between all simulated vehicles. This is one of the source of uncertainty we are looking for when using a micro-traffic simulation, and we rely on SUMO to sample states during options execution.

2.4. Options

We consider 4 classes of options that compose routes of the vehicles:

1. `PickUp(agent, request)` option.
   "`agent`" travels to the inital location of "`request`" using the fastest route, parks at the client, and waits for the package to be loaded. This option can be initiated only if the request is available, and does not overload the vehicle. It ends when package is fully loaded.
2. `Deliver(agent, request)` option.
   "`agent`" travels to the destination of "`request`" using the fastest route, parks at the client, and waits for the package to be unloaded. This option can be initiated only if the request has already been picked up by the vehicle. It ends when package is fully unloaded.
3. `Charge(agent, charger)` option.
   "`agent`" travels to the charging station "`charger`" using the fastest route, parks at the station, and waits for its battery to be fully recharged. This option can be initiated only if the battery of the vehicle is not full. It ends when battery is fully recharged.

We illustrate in Figure 1 a situation where the agent has 4 available options. More classes of options can be added using the provided interface and the abstract `Option` class.

2.5. Rewards

The learning agent receives feedback at each time step. The overall immediate reward consists of:

- Cost on energy consumed in the last time step,
- Reward on package delivery,
- Fees for late pick up or delivery.

3. Interfacing with Micro-Simulation

We depict in Figure 2 the software architecture used in our SMDP interface *SULFR* to SUMO. Road networks can be manually generated following geometric patterns or imported from existing cartographic data. For example, we can import data from the *Open Street Map* community-maintained online database. Traffic conditions consist in

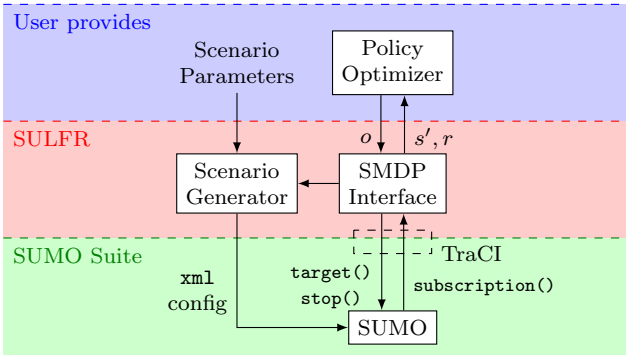

Figure 2. The Software architecture of our *SULFR* interface to SUMO. User chooses a scenario and plugs in the algorithm optimizing the policy. *SULFR* generates SUMO configuration files to launch the simulator, and translates options into target and stop commands for the agents in the simulation. It triggers callbacks at each simulation step to update states and reward signals. Algorithm is notified when an option terminates.

pre-defined routes for uncontrollable vehicles. Different routes can be randomly generated at the beginning of each episode to model the uncertainty in the road congestion, or again imported from actual traffic flow measurement data. Clients' locations and charging station can also be randomly distributed along edges of the road network, or imported from real-world locations. These inputs are given to the simulator as `xml` files by the Scenario generator.

SUMO uses extra modules called *devices* to augment the capabilities of simulated vehicles. The vehicle type of the agents includes a *rerouting* and a *battery* device. The former enables online target assignment and partial route planning, while the latter just add an electric consumption model to the vehicle.

The interface then starts the simulation, and subscribes to state updates at every simulation time step. New options however are only sent back to an agent in the simulation when its previous option has reached a termination state. A detailed overview of the object-oriented structure of the interface is shown on Figure 3.

4. Application example

To demonstrate the policy evaluation capabilities of our *SULFR* interface, we created 3 basic scenarios (see Figure 4): A *Grid* and a *Spider* road networks with random traffic and clients' locations, and a more realistic *Bologna* scenario imported from existing work (Bieker et al., 2014). We generated 10 requests at random clients' locations, and granted a reward of 100 for successful deliveries, penalized by a late fee of 1/s. The energy cost was equal to 1/Wh. We evaluated 2 handcrafted policies on options: A random policy denoted μ^{rand} that choose an option uniformly among all

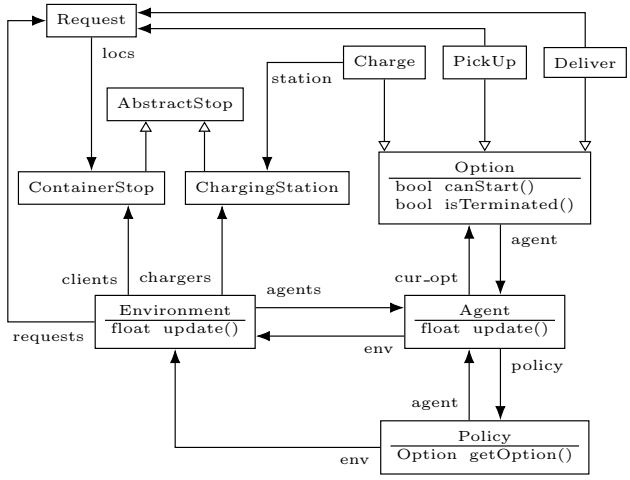

Figure 3. A simplified class diagram of the *SULFR* interface. Only key methods are included: `update()`, called at all simulation time step to update the state of the agents and the environment; `isTerminated()`, also called at every time step, checking for termination condition of the options; `canStart()`, used to filter available options when the previous one has ended. Policies have access to the whole state of the agents and the environment.

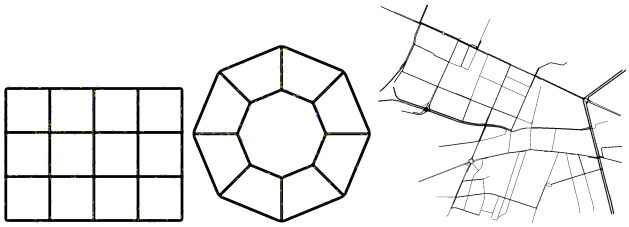

Figure 4. Grid (left), *Spider* (center) and *Bologna* (right) road networks used as test benches. *Grid* and *Spider* were automatically generated and populated with random flows of traffic. *Bologna* was imported from a previous project based on SUMO, and represent a district of Bologna, Italy.

available ones in the current state, and a nearest neighbour policy μ^{NN} sending agents to pick up or deliver the closest available requests.

We ran simulations for each policy and on each scenario for 50 episodes, with stochastic customers and variable traffic conditions. We plotted on Figure 5 the evolution of the policy evaluation over episodes. As the number of episodes grows, we can see that the return stabilizes to the expected value of the policies' performances. Figures 5 reveals that the very basic *NN* policy outperforms the *rand* one on all scenarios. However its advantage is almost negligible on the more complicated and realistic *Bologna* scenario.

Reinforcement learning can require quite a high number of samples to reliably estimate the policy. Running parallel simulations on multi-threaded architecture might significantly speed up the policy evaluation process. Still, we measured average running time in the order of a minute for episodes of 600 time-steps (i.e. 10min of simulation). More development needs to be done to speed up the simulation process, but also make the execution fully reliable, as we are experiencing some minor instabilities in the current implementation. Nonetheless, these experiments show how the simulator can be used as a generative model to sample trajectories and rewards to evaluate a policy.

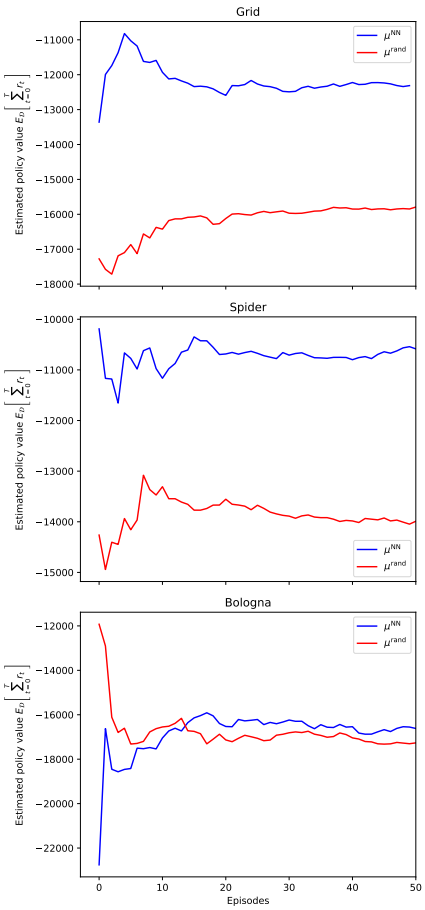

Figure 5. Evolution of estimated policy value over episodes for both policies *rand* (red) and *NN* (blue) on all scenarios. After a few episodes, policy estimation stabilizes to its expected value.

5. Conclusion and Perspective

We presented the *SULFR* interface to the SUMO micro-traffic simulator made for single- and multi-agent reinforcement learning in SD-VRPs as SMDPs. We demonstrated how this interface can be used to sample trajectories of states and rewards in customizable scenarios. We plan to extend our model to use interruptible options, and to improve the running time by optimizing simulator calls and sampling trajectories using multiple simulations in parallel.

References

Bieker, L., Krajzewicz, D., Morra, A. P., Michelacci, C., and Cartolano, F. Traffic simulation for all: a real world traffic scenario from the city of bologna. In *SUMO 2014*, May 2014.

Bouzaiene-Ayari, B., Cheng, C., Das, S., Fiorillo, R., and Powell, W. B. From single commodity to multiattribute models for locomotive optimization: A comparison of optimal integer programming and approximate dynamic programming. *Transportation Science*, 50(2):366–389, 2016.

Chen, M.-C., Hsiao, Y.-H., Reddy, R. H., and Tiwari, M. K. The self-learning particle swarm optimization approach for routing pickup and delivery of multiple products with material handling in multiple cross-docks. *Transportation Research Part E: Logistics and Transportation Review*, 91:208 – 226, 2016.

Coltin, B. and Veloso, M. Online pickup and delivery planning with transfers for mobile robots. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5786–5791, May 2014.

Cordeau, J.-F. and Laporte, G. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579 – 594, 2003.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1329–1338, New York, New York, USA, 20–22 Jun 2016. PMLR.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40(2):211–225, 2006.

Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L. Recent development and applications of sumo - simulation of urban mobility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

Krauß, S. Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. Technical report, 1998.

Mahadevan, S., Marchalleck, N., Das, T. K., and Gosavi, A. Self-improving factory simulation using continuous-time average-reward reinforcement learning. In *Proceedings of the 14th International Conference on Machine Learning*, pp. 202–210. Morgan Kaufmann, 1997.

Maxwell, M. S., Restrepo, M., Henderson, S. G., and Topaloglu, H. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*, 22(2):266–281, 2010.

Novoa, C. and Storer, R. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 196(2):509 – 515, 2009.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. Variable neighborhood search for the dial-a-ride problem. *Computers and Operations Research*, 37(6):1129 – 1138, 2010.

Sáez, D., Cortés, C. E., and Núñez, A. Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. *Computers and Operations Research*, 35(11):3412 – 3438, 2008. Part Special Issue: Topics in Real-time Supply Chain Management.

Schilde, M., Doerner, K., and Hartl, R. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, 238(1):18 – 30, 2014.

Simão, H. P., Day, J., George, A. P., Gifford, T., Nienow, J., and Powell, W. B. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, 43(2):178–197, 2009.

Solomon, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

Sutton, R. and Barto, A. *Reinforcement Learning: An Introduction*. A Bradford book. Bradford Book, 1998.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, August 1999.

Toth, P. and Vigo, D. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002.

Wu, C., Kreidieh, A., Parvate, K., Vinitsky, E., and Bayen, A. M. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *CoRR*, abs/1710.05465, 2017.