Machine Learning Guided Optimization for Demand Responsive Transport Systems*

Louis ZIGRAND[1(✉)][0000−0003−4472−4855],
Pegah ALIZADEH[2][0000−0002−7231−5840],
Emiliano TRAVERSI[1][0000−0003−4673−3982], and
Roberto WOLFLER CALVO[1,3][0000−0002−5459−5797]

[1] LIPN (CNRS – UMR 7030), Université Sorbonne Paris Nord, Paris, France
{zigrand, traversi, wolfler}@lipn.univ-paris13.fr
[2] Léonard de Vinci Pôle Universitaire, Research Center, Paris La Défense, France
pegah.alizadeh@devinci.fr
[3] DIM, Università di Cagliari, Cagliari, Italy

Abstract. Most of the time, objective functions used for solving static combinatorial optimization problems cannot deal efficiently with their real-time counterparts. It is notably the case of Shared Mobility Systems where the dispatching framework must adapt itself dynamically to the demand. More precisely, in the context of Demand Responsive Transport (DRT) services, various objective functions have been proposed in the literature to optimize the vehicles routes. However, these objective functions are limited in practice because they discard the dynamic evolution of the demand. To overcome such a limitation, we propose a Machine Learning Guided Optimization methodology to build a new objective function based on simulations and historical data. This way, we are able to take the demand's dynamic evolution into account. We also present how to design the main components of the proposed framework to fit a DRT application: data generation and evaluation, training process and model optimization. We show the efficiency of our proposed methodology on real-world instances, obtained in a collaboration with Padam Mobility, an international company developing Shared Mobility Systems.

Keywords: Demand Responsive Transport · Surrogate Modeling · Combinatorial Optimization.

1 Introduction

Shared Mobility Systems cover all means of transport that are shared between users, either sequentially or with grouping. In particular, Demand Responsive Transports (DRTs) are shared transport systems where the vehicles adapt their routes dynamically to the demand rather than using fixed routes and timetables. This growing mode of transport, unlike classic public transport, allows the users to book a place in a vehicle by requesting in real time their departure and arrival

* Supported by Padam Mobility under CIFRE Convention 2019/1809 (to L. Z.).

points as well as their desired pick-up or drop-off time [3]. The motivation of using such hybrid models is twofold: they straddle conventional public transport and taxis, as their schedules and routes are quite flexible, and they constitute a viable alternative to individual transport due to a lower cost of use. Still, the management of a DRT system needs efficient decision tools to handle the users' requests, the routing of the vehicles as well as the quality of the service [14, 28].

1.1 Context

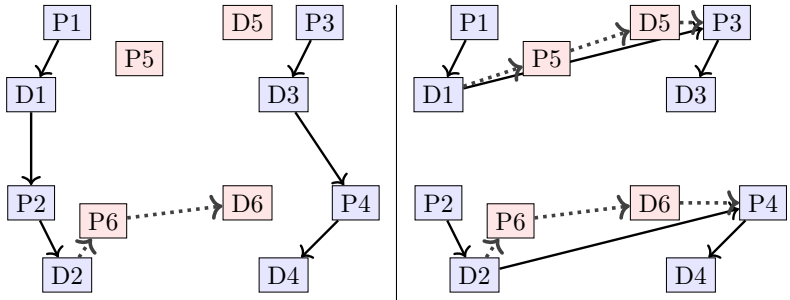
This work is performed jointly with Padam Mobility, a well-established company that provides technological support to DRT services. Their production, which has been operational in around 50 territories for 5 years, generates a significant amount of data: each territory involves hundreds of service points and thousands of travel requests monthly. They operate as follows: at any time, a user can submit a new request to the system via an application. A request stands for a departure location, an arrival location and a desired time of pick-up or drop-off. If the request needs to be served on the same day, it is named an *online* request; otherwise, it is named an *offline* request (also called *in-advance* in the literature). Each night, Padam Mobility optimizes the initial vehicles' routes for the next day by giving the offline requests to an Offline Optimization Framework (OOF). The day after, the vehicles start following the routes scheduled by this offline planning. During the day, each time that an online request pops up, an insertion algorithm decides either to accept the request and suggest a trip proposition to the user, or to reject it immediately. Then, if a proposition is validated by the user, the algorithm updates the routes with regards to the accepted request.

1.2 Motivation

The optimization of the vehicles offline scheduling can be modeled as a Dial-a-Ride Problem (DaRP) [13]. A DaRP can be static or dynamic: in the first case, all requests are known in advance, while the system handles requests as they occur in the second case . Our work is on the Dynamic DaRP but the design of the offline planning can be viewed as a Static DaRP tackled with a scenario based optimization algorithm to take into account the future online requests. The vast majority of the algorithms present in the DaRP literature use objective functions related to the minimization of the time traveled [22]. As for Padam Mobility, it is the Total Duration of the Rides (TDR) i.e., the accumulated time on the road of all vehicles during the service that is being minimized.

We display in Fig. 1 a basic example of what motivated our work. Here, we describe six requests by their pick-up (P) and drop-off (D) locations. In each scheduled path, the first four, in blue, are offline requests while the last two, in red, are the online ones. The solid lines represent the offline planning of the vehicles while the dotted ones show the possibility to insert the online requests within those initial schedules.

In terms of operational cost, Fig. 1a shows an optimized offline planning regarding the offline requests and the TDR. However, this set of routes cannot



(a) TDR-optimized schedule (b) Flexible schedule

Fig. 1. Visual comparison between two offline schedules

serve every online request due to the implied detour. On the other hand, the initial routes displayed in Fig. 1b are sub-optimal in terms of TDR but allow the system to serve all the online requests. This is a very simple but visual case where minimizing the TDR can lead to bad routes when online requests start to arrive. Since the online requests are unknown to our system, we are interested in studying the historical demand's data for a specific city. This will allow us to design more flexible offline schedules regarding the online requests, like the ones shown in Fig. 1b, and thus to increase their acceptance rate.

1.3 Contribution

In this work, we mainly focus on improving the algorithms used to optimize the offline planning of a DRT service, in order to increase its online requests acceptance rate. To the best of our knowledge, there exist only a few works that deal with the scheduling of routes without knowing all the requests ahead of time (see Section 2) and none of them considers such a procedure. Therefore, the main contributions of this work are the following:

- In Section 3, we propose to build a Machine Learning model able to estimate, for a given set of routes of vehicles, the expected number of online requests that such initial set of routes will be able to serve the day after. We then optimize the offline planning of a DRT service using this objective function.
- The data used to train the proposed model is produced via a Simulation Framework, based on the historical data of Padam Mobility. To analyse the challenging data with Machine Learning approaches, we propose in Section 4 a framework that models the data generation, the training process and the prediction phase with a generic approach.
- The results compared to the existing objective functions in Section 5 show the efficiency of our approach. We experimentally demonstrate that using traditional myopic approaches do not provide routes with the necessary flexibility to react to the arrival of online requests.

We stress the fact that the proposed methodology is not tailored only for Padam Mobility as it could be adapted to many DRT systems.

2 Related work

Our methodology is at the frontiers of 3 computer science fields: Combinatorial Optimization, Simulation-based Optimization (SbO) and Machine Learning. The first provides the application, the DaRP, and the general techniques to solve it. Then, looking at the problem under the lenses of SbO allows us to view the problem in a new light with the definition of a new objective function. Finally, Machine Learning techniques applied to Surrogate Modeling provide a solution to computational limitations met with SbO, with means to learn a more suitable objective function. In the rest of this section, we present the literature related to each of the mentioned fields that are the most related to our approach.

Dial-a-Ride Problem There is a large body of research directed at the Static DaRP in Combinatorial Optimization [13]. The exact approaches [7], which are mainly based on the Branch and Bound method, can guarantee the optimality of the solution but are expensive in terms of time and resources. Heuristic approaches [17] are far less expensive but can return sub-optimal solutions instead of globally optimized ones. Among them, the Adaptive Large Neighborhood Search (ALNS) method has been shown to perform well on several instances of the Static DaRP [11, 28]. In all those approaches, the most classic objectives are either linked to operating costs or to the service quality, such as the number of unserved requests, the total duration of passenger transport, the total duration of the rides, the number of vehicles required, etc. [22]. The uncertainty related to the online requests in the dynamic version of the DaRP can be tackled by working on the algorithms in charge of the offline planning and the insertion of the online requests. Given the center of interest of this work, we solely focus on the first axis. One way to make space in the offline schedule is to add fake requests from a clustering of historical data to the set of in-advance requests and solve a Static DaRP [26] but this model proved too simple to improve the flexibility of the rides. Two-stage [6] and multi-stage [24] stochastic programming models have been proposed for the Dynamic and Stochastic Vehicle Routing Problem to optimize the offline schedule but these approaches provide limited enhancement due to the simplified models used within the recourse phase.

Simulation-based Optimization SbO designs a subfield of Operations Research where the evaluation of a solution is not computed with an explicit mathematical formula but by the means of simulations [2]. Such models can provide a better description of real-world situations than simplified procedures, if any exists, but at the price of needing bigger computation resources. Hence, SbO often goes with Surrogate-based Optimization [4]. To the best of our knowledge, there has been no research on using SbO to handle the offline planning of DRT services.

Surrogate Modeling Also known as Metamodeling, it forms a subfield of Machine Learning that consists in representing a complex model f with a simpler model $\hat{f}$ at the price of some approximation. The purpose of the latter can be, for instance, to enable a faster computation of new values of f. As simulations are

usually expensive in both computer resources and time, this paradigm is often found in the context of SbO [2, 4] and is one of many interactions between the Machine Learning and Combinatorial Optimization fields [5]. In particular, how to obtain simple but accurate models in the fewest possible simulations has been studied in various works, such as for non-differentiable objective functions [9], complex multi-objective optimization [20] or large-scale problems [21]. For a more in-depth analysis of Surrogate Modeling, we refer the reader to [1].

Machine Learning Techniques As the resulting model should be able to discriminate between good and bad solutions, we face here a ranking problem. This kind of problem is well known and can be tackled with various "Learning to Rank" techniques [18]. In our case, as the output of a simulation is not deterministic (see Section 4.3), we do not have a reliable ground truth of the ranking between the solutions. Consequently, we cannot apply conventional pairwise nor listwise algorithms to handle our problem. Thus, our objective is to learn a pointwise ranking function where the diversity of the training set will allow us to statistically describe the best solutions. To do so, the most common Supervised Machine Learning techniques in the literature are Gaussian Process Regressions (GPRs), Artificial Neural Networks (ANNs) and Radial Basis Functions (RBFs). A GPR model, also known as Kriging method, is based on the idea of considering the function f as a Bayesian model [25]. These models have been used in various applications, such as the optimization of the gait for quadrupedal and bipedal robots [19] or the optimization of hydrofoil shape design [23]. An ANN is a set of connected processing units that receive, transform and transfer information from one to another. They have been shown to outperform classic regression models (see for example [16]). General guidelines about how to model a stochastic simulator with ANNs are discussed in [10] and applied to a job shop problem. Recently, Convolutional Neural Networks (CNNs) have notably proven to be effective in modeling fluid dynamics applications [12, 31]. RBFs follow the idea that the value of an entry depends solely on its distance to predefined reference points [15]. A particular variant, Radial Basis Function Networks (RBFNs), designs a kind of neural network where RBFs are used in place of activation functions and has shown promising results [30].

3 Machine Learning Guided Optimization

To better illustrate the generality of our methodology, namely Machine Learning Guided Optimization (MLGO) framework, we keep this section as general as possible. Ideally, we would like to solve the following optimization problem:

$$\max_{x \in X} f(x) \tag{1}$$

where X represents the domain of the variables x and f is a generic objective function defined over the domain X.

However, the evaluation of f can only be done through simulations and such a process is too time consuming to be used in practice in an optimization algorithm

where thousands of calls to f would be made. To overcome this drawback, we propose to use a surrogate model $\hat{f}$ of f. A key aspect in our method is therefore to ensure that the obtained model is sufficiently accurate and simple.

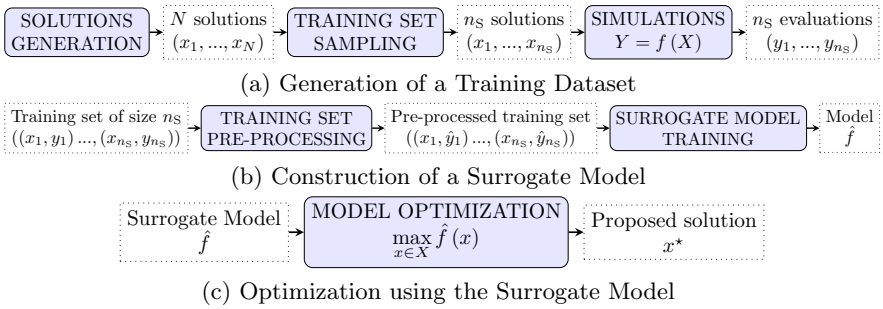


Fig. 2. Overview of the MLGO Framework

We present in Fig. 2 an overview of the MLGO approach in 3 steps:

- Fig. 2a – *Training Dataset*. The goal of this step is to create the dataset required to train the model. It starts by generating a pool of solutions of our problem from which a set of n_S distinct solutions $(x_1, \ldots, x_{n_S})$ is sampled (see Section 4.2). Then, those solutions are evaluated with f (see Section 4.3) to obtain their realized values $(y_1 = f(x_1), \ldots, y_{n_S} = f(x_{n_S}))$.
- Fig. 2b – *Model Training*. This second step takes the training dataset from the previous step, $((x_1, y_1), \ldots, (x_{n_S}, y_{n_S}))$, pre-processes it and computes the surrogate model $\hat{f}$ through a training phase (see Section 4.4).
- Fig. 2c – *Optimization*. Finally, $\hat{f}$ is defined as the objective function within an optimizer (see Section 4.5) to obtain a solution to the problem.

4 MLGO applied to DRT Systems

We describe in this section how the main components of the MLGO framework are developed to fit a DRT application and ours in particular.

4.1 Model and Notations

Before going into detail, we present some notations and definitions that will be used in the rest of the paper. We note V the set of n_{V} vehicles available, $\mathrm{B} = [\mathrm{lat}_{\min},\ \mathrm{lat}_{\max}] \times [\mathrm{long}_{\min},\ \mathrm{long}_{\max}]$ a bounding box defining the considered territory and $\mathrm{D} \in \mathrm{B}$ the coordinates of the vehicles depot. The time horizon T is discretized into a sequence of n_{T} regular time steps $(t_1 ... t_{n_{\mathrm{T}}})$.

Definition 1. *An online request is represented by a pick-up time $t_p \in T$ and place $\left(lat_p, long_p\right) \in B$, a drop-off time $t_d \in T$ and place $(lat_d, long_d) \in B$, and*

an arrival time $t_a \in T$. An offline request has the same elements, except for the absence of a t_a as the user's request arrived in the past and is already served.

Definition 2. *An online (resp. offline) scenario is a set of online (resp. offline) requests that want to (resp. must) be served by the fleet V of vehicles.*

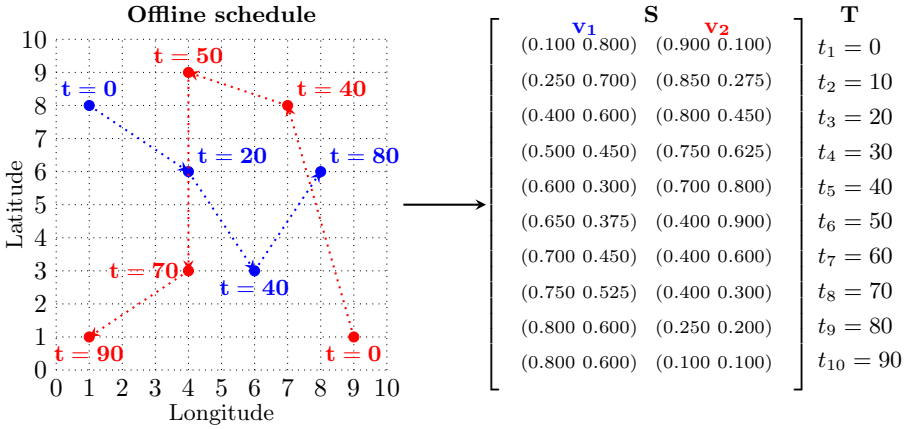


Fig. 3. Modeling of the offline schedule of two vehicles as a tensor

Definition 3. *A solution x corresponds to a set of routes for the n_V vehicles, leaving from D after t_1 and returning to D before t_{n_T}. A solution is formally modeled by a tensor $\mathbf{S} \in \mathbb{R}^{V \times T \times 2}$ where, $\forall\,(v,t) \in V \times T$, $\mathbf{S}_{v,t}$ is the normalized position (with respect to B) of the vehicle v at time step t. A solution is said feasible for a given offline scenario if its planning can serve all its offline requests.*

Definition 3 implies that, in general, a randomly chosen tensor $\mathbf{S} \in \mathbb{R}^{\mathrm{V} \times \mathrm{T} \times 2}$ does not represent a feasible solution. It is therefore not an easy task to obtain a set of diversified solutions (see Section 4.2 for more details). We provide in Fig. 3 a simple but visual example of the tensor representation of a solution.

We recall that, in this work, we suppose the offline scenario to be given and the online scenario to be uncertain, which matches the situation where the OOF is executed each night for the next service. This implies that the generation of solutions and the optimization (see Fig. 2) take as input the same offline scenario: in both cases, we need to produce one or more feasible solutions while the feasibility of a solution only depends on the offline scenario considered.

Also, the objective function f used in Equation (1) and Fig. 2 is the expected rate of accepted online requests returned by a simulator, being in our case the one developed by Padam Mobility (see Section 4.3).

4.2 Generation of Feasible Solutions

In the literature, most of the works on surrogate modeling consider problems where generating a sample of the space of solutions can easily be done with techniques such as Latin Hypercube Sampling [15] or Full Factorial Sampling [23]. Nonetheless, such methods can be used only if no restriction holds on the solutions. In our case, the high number of constraints defining the feasible region of the underlying DaRP (see Definition 3) requires the use of more sophisticated techniques to generate a solution.

To generate a diversified set of feasible solutions, we chose to launch the Adaptive Large Neighborhood Search (ALNS) developed by Padam Mobility (see Section 4.5) on the considered offline scenario without any objective function for a few minutes and save each new schedule found during the search. Once a large enough number of solutions has been obtained by this procedure, we use a stratified sampling strategy based on the KMeans clustering algorithm, the tensor representation of the solutions (see Definition 3) and the Euclidean distance to cover the solution space as homogeneously as possible. In practice, this procedure allowed us to produce diversified training sets of 1800 samples, validation sets of 200 samples and testing sets of 1000 samples for all our instances.

4.3 Simulation Framework

The Simulation Framework must take as input a feasible solution x and return the expected number of accepted online requests $f(x, s)$ for an online scenario s.

Any procedure that does this job can be used. In our case, Padam Mobility has its own simulator that provides a good enough representation of how a DRT service behaves through a working day. In particular, it induces some randomness in the behaviour of the virtual users when they are facing equivalent propositions for their requests to be served. As a change of a few minutes on a serving time can have an impact on future requests and considering practical experiments, we will then consider the output of a simulation as stochastic but stable. More precisely, ~ 10 runs of a solution through the simulator is enough to fully evaluate it.

The set of online requests that will actually arrive along the day can be viewed as a random variable. In this work, we make the hypothesis that we have at our disposal a stochastic model to generate plausible scenarios of online requests for a given day. Recent works, such as [27] or [29], sustain this assumption.

We therefore decide to design a new objective function by first selecting n_σ online scenarios from a given pool of online scenarios (see Section 5). Then, we simulate once the working day for each of the selected online scenario, starting from a given offline scenario, and we return the average percentage of accepted online requests over the n_σ online scenarios.

4.4 Surrogate Model

The Surrogate Model must be a model $\hat{f}$ that imitates the original and unknown objective function f: for a feasible solution x and a set of online scenarios S, $\hat{f}(x) \sim f(x, S)$. In principle, any supervised approach can be used to obtain $\hat{f}$.

In our case, we present in Section 5.1 the various Machine Learning models considered within this study as well as the methodology used to decide which one should be used for the optimization. Nonetheless, the design choices of $\hat{f}$ should follow two main principles: firstly, it must be a relatively minimalist and compact model so that the training phase is not significantly time consuming. Secondly, it must scale well with the size of the instances without losing in quality.

In terms of data pre-processing, the input values are the tensor representation of the schedules (see Definition 3) while the output values of the training set are standardized. Then, as we experimentally observed that our instances present Gaussian-like distributions of values over their generated pool of offline schedules, the weights of the Weighted Mean Square Error loss function used to train the models are designed so that each value has a similar impact on the loss.

4.5 Offline Optimization Framework

The Offline Optimization Framework (OOF) must consider an offline scenario $s_{\texttt{Off}}$ and return an optimized schedule $x = \text{OOF}(s_{\texttt{Off}}, \mathcal{O})$ for an objective $\mathcal{O}$.

In this work, we used a version of ALNS [11, 28] modified by Padam Mobility to handle their own application, in order to respect their customized constraints. The ALNS method is a Local Search metaheuristic. This variety of algorithms explores the search space of feasible solutions by applying local changes until no improvement can be found and therefore a (locally) optimal solution is identified.

In our context, the advantage of this method is that we can keep the structure of the in-house ALNS and change only the part related to the evaluation of a solution by using the surrogate model presented in Section 4.4 in place of the currently used objective function, based on the minimization of the TDR.

5 Experiments

To assess the quality of our framework, we consider 25 challenging instances provided by our industrial partner. In Table 1, we display the following information relative to the selected instances: the identifier of each *Instance*, used in later references, the service *Duration* i.e., the time length during which users' requests can be served, the number of *Vehicles*, the number of historical *Offline requests* and the number of historical *Online requests*. Regardless of the service duration, the discrete time horizon T for the proposed instances is always of size 100.

Table 1. Presentation of the considered instances

Instance	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
Vehicles	2	2	3	2	2	3	3	2	2	2	12	12	5	6	12	11	12	9	8	13	11	11	9	12	11
Duration (in h)	7	7	7	7	7	7	7	7	7	7	12	15	13	13	15	15	15	15	13	13	15	15	13	15	15
Surface (in km^2)	46	46	46	46	46	46	46	46	46	46	197	197	144	144	197	197	197	197	144	144	197	197	144	197	197
Offline requests	30	41	35	30	41	48	45	42	44	52	41	49	45	58	62	45	51	63	91	86	82	92	109	86	86
Online requests	32	24	32	41	36	31	42	50	54	60	74	68	79	85	92	109	104	107	80	88	156	154	179	204	204

We note $\mathbf{I}$ the set of instances. Regarding the Simulation Framework presented in Section 4.3, we consider two sets of scenarios of online requests:

- *Historical Configuration (HC)*
  This option represents the optimistic point of view, where the online requests are perfectly known in-advance. Hence, we use $n_\sigma = 1$ scenario made of the historical online requests that occurred during the actual service.
- *Robust Configuration (RC)*
  This option represents a noisy forecast of the upcoming online requests. We generate $n_\sigma = 5$ scenarios based on the historical set of online requests with a randomized order of arrival. Furthermore, the departure and destination positions are moved randomly in a 500 meters radius as well as the requested time in a 30 minutes time window. This manually added noise represents the generation of scenarios based on a forecasted Origin-Destination matrix with areas of 1 km^2 and time steps of 30 minutes [29].

5.1 Choice of a Machine Learning model for the Optimization

In this section, we first describe the Machine Learning techniques that we have implemented and tested to approximate the Simulation Framework. Then, we present the methodology used to evaluate the different models in order to choose which one should be used within the optimization process.

Presentation of the Surrogate Models

Radial Basis Function Network (RBFN) We implement the variant of this model where the centers of the Radial Basis Functions are computed based on a clustering technique, using the *KMeans* method of the scikit-learn Python package[1], before being fed to an ANN, using the TensorFlow Python package[2].

Gaussian Process Regression (GPR) We use the model provided by the scikit-learn Python package [1] named *GaussianProcessRegressor*.

Feedforward Neural Network (FNN) Our FNN structure consists of a first layer to flatten the tensors provided in input as vectors of size $200 \times n_{\mathrm{V}}$, 4 hidden layers with 1024 neurons, and a final layer with a single neuron to compute the output of the model. All nodes of the FNN have a linear activation function. The hyperparameters used for the training phase are: 5000 epochs, a batch size equal to the Training Set size, and an Adam optimizer with a learning rate of 1E-5. We implemented this model with the TensorFlow Python package [2] .

[1] See https://scikit-learn.org/ for more information.
[2] See https://www.tensorflow.org/ for more information.

Convolutional Neural Network (CNN) Our architecture consists of 5 successive Convolutional Layers with a Kernel size of (1, 3), a Stride of (1, 2), a Dropout rate of 0.5 and a Rectified Linear Unit (ReLU) as activation function. The size of the tensor after each layer is respectively: $n_{\mathrm{V}} \times 49 \times 32$, $n_{\mathrm{V}} \times 24 \times 64$, $n_{\mathrm{V}} \times 11 \times 128$, $n_{\mathrm{V}} \times 5 \times 256$ and $n_{\mathrm{V}} \times 1 \times 512$. The output of the last Convolutional Layer is then flattened as a vector of size $512 \times n_{\mathrm{V}}$ and fed to a Fully Connected Layer of the same size with a linear activation function. Finally, the last layer is the Output Layer with a single neuron and a linear activation function to compute the output of the model. The hyperparameters used for the training of the CNN are: 10000 epochs, a batch size equal to the Training Set size, and an Adam optimizer with a learning rate of 1E-4. We implement this model with TensorFlow[2].

Ensemble Learning Model (ELM) We define this model as a weighted sum of the previously cited models: as each Machine Learning technique learns differently through its learning phase, such model can take the best of them [8]. In practice, we used the following combination as it proved to be the most stable one, in terms of noise reduction: $\mathrm{ELM} = 0.5 \times \mathrm{FNN} + 0.35 \times \mathrm{CNN} + 0.15 \times \mathrm{GPR}$.

For the sake of fairness, the hyperparameters of each model have been optimized to obtain the highest possible accuracy, while maintaining a size that allows to train the methods in a reasonable amount of time.

Evaluation of the Surrogate Models

Notations For each instance $I \in \mathbf{I}$, we note $\mathtt{Train}_I$ the training set and $\mathtt{Test}_I$ the test set generated for I. We note $\mathbf{M}$ the set of considered surrogate models.

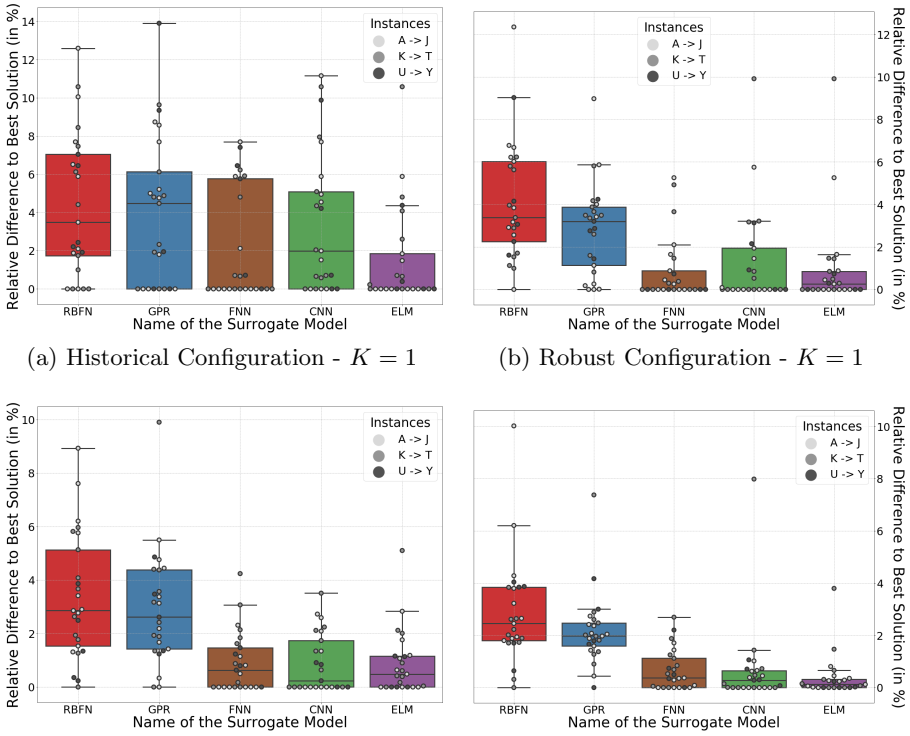
Methodology In order to evaluate the generalization capacity of the different surrogate models, we use the following process:

- For each instance $I \in \mathbf{I}$, we define $\mathtt{Train}_I^{10\%}$ the subset of $\mathtt{Train}_I$ that contains the solutions with a value in the top 10% values of the set:

$$\mathtt{Train}_I^{10\%} = \left\{ (x;y) \in \mathtt{Train}_I \;\middle|\; y \geq 0.9 \times \left(\max_{(x';y') \in \mathtt{Train}_I} y' \right) + 0.1 \times \left(\min_{(x';y') \in \mathtt{Train}_I} y' \right) \right\}$$

- For each surrogate model $M \in \mathbf{M}$, we train M on $\mathtt{Train}_I \setminus \mathtt{Train}_I^{10\%}$ and evaluate the extended test set $\mathtt{Test}_I \cup \mathtt{Train}_I^{10\%}$ with the obtained model. Then, we note $\left(y_{I,k}^M \right)_{1 \leq k \leq K}$ the full evaluation by the Simulation Framework of the top K solutions of the extended test set of I according to M for a given $K \in \mathbf{N}^\star$. If more than one solution can be fully evaluated at the end of the optimization, this K represents some tolerance on the model forecast.
- Finally, $\forall\ I \in \mathbf{I}$, $\forall\ M \in \mathbf{M}$, we define $g_{I,K}^M$ the Relative Difference to the Best Solution of order $K \in \mathbf{N}^\star$ as follows: $g_{I,K}^M = \frac{y_{I,K}^{\mathrm{Best}} - \left(\max_{1 \leq k \leq K} y_{I,k}^M \right)}{y_{I,K}^{\mathrm{Best}}}$, where

$y^{\text{Best}}_{I,K} = \max_{\substack{M' \in \mathbf{M} \\ 1 \leq k \leq K}} y^{M'}_{I,k}$. The idea behind $g^{M}_{I,K}$ is to measure how well each model has learned what made a solution a good one in comparison to the others, up to a given tolerance, defined by the number of solutions considered K.



(a) Historical Configuration - $K = 1$

(b) Robust Configuration - $K = 1$

(c) Historical Configuration - $K = 5$

(d) Robust Configuration - $K = 5$

Fig. 4. Evaluation of the Surrogate Models

Results We present in Fig. 4 the aggregated results of the previously cited surrogate models over the 25 considered instances. In both Fig. 4a and Fig. 4d, we can see that the ELM clearly provides the best results overall, in spite of a few outliers. In Fig. 4c and Fig. 4b, the situation is slightly more contrasted with similar performances of the FNN, CNN and ELM models. On the other hand, ELM clearly provides the best results in Fig. 4d, which is the most robust case, where we use the Robust Configuration, paired with a tolerance of 5 instances. Therefore, as the ELM model provides the most stable results overall, we decide to use it within the OOF. We note ELM (HC) (resp. ELM (RC)) the ELM trained with the Historical (resp. Robust) Configuration.

5.2 Computational Results

The experiments were led on a computer equipped with an AMD® Ryzen 7 3700x for CPU, allowing us to parallelize the evaluation through the simulator of the training set, a GeForce RTX 2060 SUPER for GPU, making the training of our ANNs be quick, and 16 Go of RAM while running on Ubuntu 20.04 LTS. Based on this configuration, we can evaluate the computational gain of our surrogate model in contrast with using the actual simulator within the OOF:

- Instances A to J: it takes a few seconds to run a simulation while it takes ~ 15 min to build our model then ~ 20 ms to make an evaluation. Thus, hours of computation time are saved after a thousand iterations of the optimizer.
- Instances K to Y: it takes more than ten seconds to run a single simulation while it takes ~ 30 min to build our model then ~ 40 ms to evaluate a solution afterwards. Hence, the time saved during the optimization phase becomes sizable even faster than for smaller instances.

To sum up, the design of our framework is a time-saver in contrast with a SbO design, notably by letting our model learn the stochastic behaviour of the Simulation Framework through the diversification of the training set and by making use of the parallelization capacity of modern computers.

5.3 Optimization Results

In this section, we compare the performances of the schedules obtained using the ELM with the ones that we obtain using the objective functions present in the literature: the Total Duration of the Rides (TDR), the Total Detour Time (TDT), the Forward Slack Time (FST), the Onboard Time (OT) and the Onboard Deviation (OD), all detailed in Table 2 (see [22, 28] for more details).

Table 2. Classic objective functions from the literature

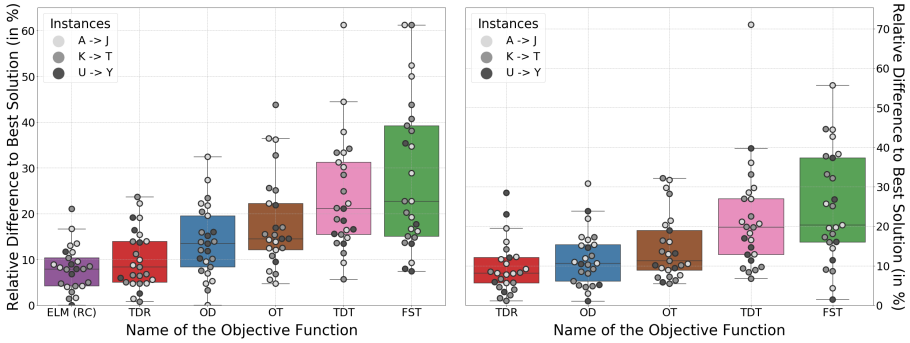
Name	Direction	Formula
TDR	min	$\sum_{\text{vehicle}} [\text{Total Travel Time}]_{\text{vehicle}}$
TDT	min	$\sum_{\text{user}} \left[\frac{\text{Actual Travel Time}}{\text{Direct Travel Time}}\right]_{\text{user}}$
FST	max	$\sum_{\text{s}\in\text{stops}}{}^{a} [\text{Maximal Arrival Time}]_{\text{s}} - [\text{Arrival Time}]_{\text{s}}$
OT	min	$\sum_{\text{user}} [\text{Actual Travel Time}]_{\text{user}}$
OD	min	$\sum_{\text{s}\in\text{segments}}{}^{b} [\text{Passengers Onboard}]_{\text{s}} \times [\text{Travel Time}]_{\text{s}}$

[a] A stop is a location and time where a vehicle handles a request.
[b] A segment is a trip of a vehicle between two consecutive stops.

For each instance $I \in \mathbf{I}$, we run the OOF on its offline scenario with the objectives shown in Table 2 and with our fully trained ELM (HC) and ELM (RC)

models. We set a time limit of 30 min for the optimizer to run and, to limit the impact of randomness, each combination of instance and objective function is solved 10 times with different random seeds. Then, for each objective $\mathcal{O}$, we note $x_I^{\mathcal{O}}$ the best solution found according to $\mathcal{O}$ for instance I and we evaluate it with 25 runs of the Simulation Framework for both configurations of online scenarios. Finally, for $C \in \{\text{HC}, \text{RC}\}$, we note $y_{I,C}^{\mathcal{O}}$ the full evaluation with the Simulation Framework of $x_I^{\mathcal{O}}$ for the set of online scenarios defined in C.

To analyze the performance of the objective functions in both configurations $C \in \{\text{HC}, \text{RC}\}$, we define $y_{I,C}^{\star} = y_{I,C}^{\text{ELM (C)}}$ as a reference value: our ELM models have been designed to maximize it. Then, we define $g_{I,C}^{\mathcal{O}}$ the Relative Difference to Best Solution of objective $\mathcal{O}$ in configuration C as $g_{I,C}^{\mathcal{O}} = \frac{y_{I,C}^{\star} - y_{I,C}^{\mathcal{O}}}{y_{I,C}^{\star}}$.



(a) Historical Configuration (b) Robust Configuration

Fig. 5. Evaluation of the Objective Functions

We present in Fig. 5 the aggregated results of our study over all instances. Fig. 5a presents how the solutions associated to each objective function would have performed in the reality. In the case of our ELM (RC) model, the observed gap can be interpreted as the cost of robustness. In spite of this, we can see that our model is more stable and provides better results than the traditional objectives overall. On the other side, Fig. 5b shows how much the newly proposed objective function performs better in comparison to the more classic objectives present in the literature in an uncertain scenario. More precisely, the ELM (RC) model is on average 9.5% better than TDR, its closest competitor, in such a context. The gap that we can find for most objectives and instances shows the lack of robustness of these functions and the place for improvement that we aim at filling with our method. Therefore, this study proves that the classic objective functions seem to be particularly unfit for the Dynamic DaRP.

6 Conclusion

In this paper, we present a new approach to model and optimize the offline planning of a Demand Responsive Transport (DRT) system, namely Machine Learning Guided Optimization. The idea of this framework is to train a Surrogate Model to estimate the expected number of online requests that a given offline schedule will be able to serve. Integrated within an optimizer, this model can then be used as an objective function to optimize the offline planning, in order to maximize the expected number of accepted online requests in the day after. We apply this methodology to a real DRT case study in collaboration with Padam Mobility, an international company specialized in Shared Mobility Systems. To obtain an effective algorithm, several aspects need to be considered: the definition of a proper dataset, an ad-hoc predictive model, and an optimisation algorithm. In this work, we show how to take care of each aspect. The experimental section of the paper clearly proves that the traditional approaches in the literature are outclassed by our framework, which provides the best performances overall. We hope that the presented work could motivate other researchers to investigate similar paradigms for other applications related to transportation problems.

Acknowledgement.

We thank Anh-Dung NGUYEN and Samir NAIM from Padam Mobility for all the useful discussions and their continuous support throughout this project.

References

1. Alizadeh, R., Allen, J.K., Mistree, F.: Managing computational complexity using surrogate models: a critical review. Research in Engineering Design **31**(3) (2020)
2. Amaran, S., Sahinidis, N.V., Sharda, B., Bury, S.J.: Simulation optimization: a review of algorithms and applications. Annals of Operations Research **240**(1) (2016)
3. Ambrosino, G., Nelson, J., Romanazzo, M.: Demand responsive transport services: Towards the flexible mobility agency. ENEA (2004)
4. Barton, R.R., Meckesheimer, M.: Metamodel-based simulation optimization. Handbooks in operations research and management science **13**, 535–574 (2006)
5. Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: a methodological tour d'horizon. Eur. J. Oper. Res. (2020)
6. Bernardo, M., Pannek, J.: Robust solution approach for the dynamic and stochastic vehicle routing problem. Journal of Advanced Transportation **2018** (2018)
7. Cordeau, J.F.: A branch-and-cut algorithm for the dial-a-ride problem. Operations Research **54**(3), 573–586 (2006)
8. Dietterich, T.G.: Ensemble methods in machine learning. In: International workshop on multiple classifier systems. pp. 1–15. Springer (2000)
9. Engilberge, M., Chevallier, L., Pérez, P., Cord, M.: Sodeep: a sorting deep net to learn ranking loss surrogates. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10792–10801 (2019)
10. Fonseca, D.J., Navaresse, D.O., Moynihan, G.P.: Simulation metamodeling through artificial neural networks. Eng. Appl. Artif. Intell. **16**(3) (2003)

11. Gschwind, T., Drexl, M.: Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. Transportation Science **53**(2) (2019)
12. Guo, X., Li, W., Iorio, F.: Convolutional neural networks for steady flow approximation. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 481–490 (2016)
13. Ho, S.C., Szeto, W.Y., Kuo, Y.H., Leung, J.M., Petering, M., Tou, T.W.: A survey of dial-a-ride problems: Literature review and recent developments. Transportation Research Part B: Methodological **111**, 395–421 (2018)
14. Huang, A., Dou, Z., Qi, L., Wang, L.: Flexible route optimization for demand-responsive public transit service. Journal of Transportation Engineering, Part A: Systems **146**(12), 04020132 (2020)
15. Ilievski, I., Akhtar, T., Feng, J., Shoemaker, C.: Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates. In: Proceedings of the AAAI Conference on Artificial Intelligence (2017)
16. Johnson, V.M., Rogers, L.L.: Accuracy of neural network approximators in simulation-optimization. J. Water Resour. Plan. Manag. - ASCE **126**(2) (2000)
17. Kirchler, D., Wolfler Calvo, R.: A granular tabu search algorithm for the dial-a-ride problem. Transportation Research Part B: Methodological **56**, 120–135 (2013)
18. Liu, T.Y.: Learning to rank for information retrieval. Springer (2011)
19. Lizotte, D.J., Wang, T., Bowling, M.H., Schuurmans, D.: Automatic gait optimization with gaussian process regression. In: IJCAI. vol. 7, pp. 944–949 (2007)
20. Lv, Z., Wang, L., Han, Z., Zhao, J., Wang, W.: Surrogate-assisted particle swarm optimization algorithm with pareto active learning for expensive multi-objective optimization. IEEE/CAA Journal of Automatica Sinica **6**(3), 838–849 (2019)
21. Mairal, J.: Optimization with first-order surrogate functions. In: International Conference on Machine Learning. pp. 783–791. PMLR (2013)
22. Parragh, S.N., Doerner, K.F., Hartl, R.F.: A survey on pickup and delivery problems. Journal für Betriebswirtschaft **58**(2), 81–117 (2008)
23. Ploé, P.: Surrogate-based optimization of hydrofoil shapes using RANS simulations. Ph.D. thesis, École centrale de Nantes (2018)
24. Saint-Guillain, M., Deville, Y., Solnon, C.: A multistage stochastic programming approach to the dynamic and stochastic vrptw. In: 12th CPAIOR. Springer (2015)
25. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems **25** (2012)
26. Tensen, I.: Stochastic optimization of the dial-a-ride problem. Dealing with variable travel times and irregular arrival of requests in the planning of special transport services. Master's thesis, University of Twente (2015)
27. Toqué, F., Côme, E., El Mahrsi, M.K., Oukhellou, L.: Forecasting dynamic public transport origin-destination matrices with long-short term memory recurrent neural networks. In: 2016 IEEE 19th ITSC. pp. 1071–1076. IEEE (2016)
28. Vallée, S., Oulamara, A., Cherif-Khettaf, W.R.: Maximizing the number of served requests in an online shared transport system by solving a dynamic darp. In: International Conference on Computational Logistics. pp. 64–78. Springer (2017)
29. Wang, Y., Yin, H., Chen, H., Wo, T., Xu, J., Zheng, K.: Origin-destination matrix prediction via graph convolution: a new perspective of passenger demand modeling. In: Proceedings of the 25th ACM SIGKDD. pp. 1227–1235 (2019)
30. Yao, W., Chen, X., Huang, Y., van Tooren, M.: A surrogate-based optimization method with rbf neural network enhanced by linear interpolation and hybrid infill strategy. Optimization Methods and Software **29**(2), 406–429 (2014)
31. Zhang, Y., Sung, W.J., Mavris, D.N.: Application of convolutional neural network to predict airfoil lift coefficient. In: AIAA/ASCE/AHS/ASC Conference (2018)