ETH zürich

263-5051-00L
Projects in ML Research

Stochastic Vehicle Scheduling

Fabio Hehli, Tobias Rahn, Leonard Schröter, Han Xi
ETH Zurich

1 The Vehicle Scheduling Problem

In the vehicle scheduling problem (VSP), we are given a set of tasks, where **each task is a delivery job** with a starting time and location and a destination. We have to complete all task by dispatching our delivery vehicles. The goal is to **minimize a linear function of the total travel time and the number of vehicles deployed.**
The problem is **modeled as a weighted directed graph** with one vertex per task and a special vertex for the vehicle depot. Tasks are connected if there is enough time to complete the first one and then go to the second one in time. The problem can then be **solved by a linear programming solver**.

2 The **Stochastic** Vehicle Scheduling Problem

The VSP assumes a deterministic environment for our delivery vehicles. It does not consider factors like traffic jams or vehicle breakdowns. To address these shortcomings, the **stochastic VSP models travel times as random**. The goal is to minimize a cost made up of the number of vehicles dispatched and delay cost.
The stochastic VSP can be solved through a MILP formulation, but this approach **does not scale beyond small problem sizes**.

3 Method

A simple way to address the stochastic VSP is to incorporate the delays into the travel times, which yields a deterministic VSP that can be solved efficiently (cf. Section 1). We choose the “amended travel times” via Machine Learning techniques. We thus learn a transformation from stochastic to deterministic VSP instances. The resulting algorithm (given in [1]) works as follows:

1. An instance of the stochastic VSP is encoded with a feature vector per arc (e.g. travel time, starting in depot, slack quantiles, slack CDF values).
2. Each arc’s features are mapped to an “amended” edge weight.
3. The weights are used as travel times and the instance is solved with the deterministic VSP solver.

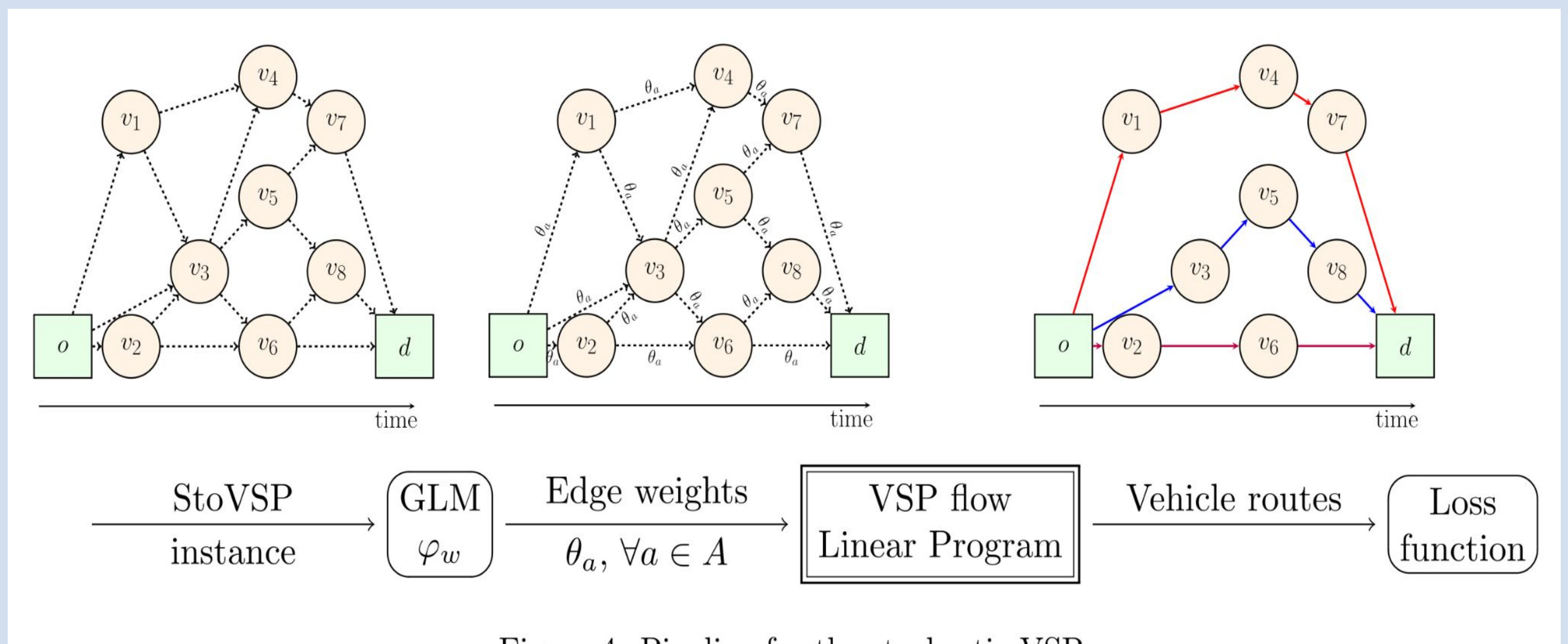
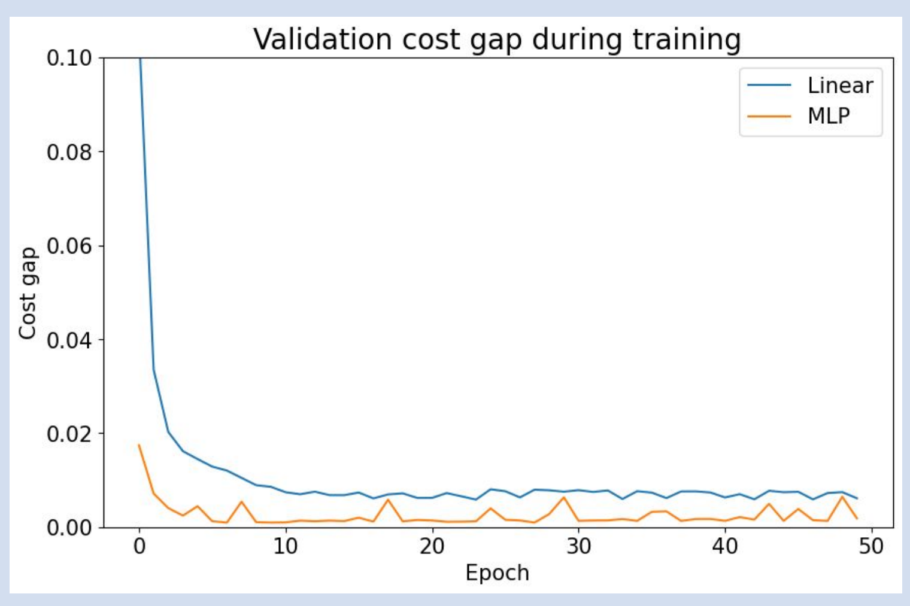


Figure 4: Pipeline for the stochastic VSP

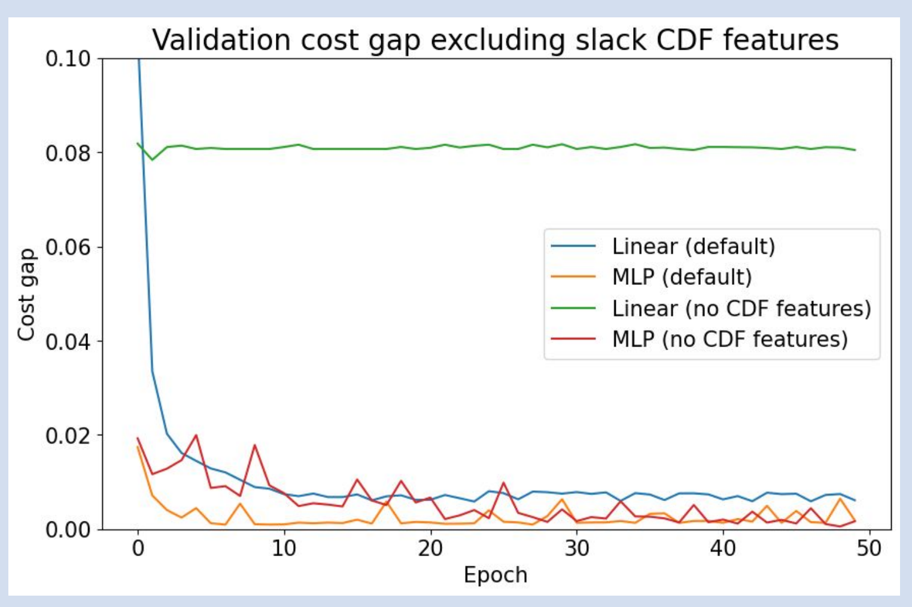
The end-to-end solver is made differentiable by using the Fenchel-Young loss [2]. It is applicable to optimization problems of the form $y^*(\theta) = \arg\max_{y \in \mathcal{C}} \theta^\top y$ over convex $\mathcal{C}$ (like the deterministic VSP). It makes the map $\theta \mapsto y^*(\theta)$ differentiable by *smoothing* the objective function. That is, we instead use the perturbed optimizer $y^*_\varepsilon(\theta) = \mathbb{E}_{Z \sim \mathcal{N}(0,I)}[\arg\max_{y \in \mathcal{C}} (\theta + \varepsilon Z)^\top y]$, which is differentiable and satisfies a lot of nice properties. For instance, $\lim_{\varepsilon \to 0} y^*_\varepsilon(\theta) = y^*(\theta)$.

4 Results

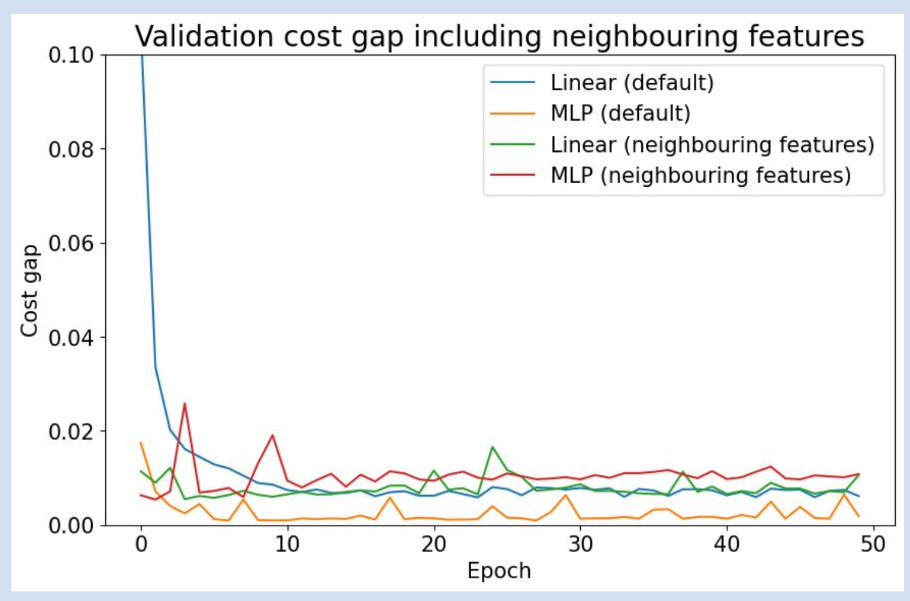
The **cost gap** describes the percentage difference in cost between the current and optimal solution. The delays are obtained by sampling congestion variables for each region in the task grid. All training was done on instances with 25 tasks and 10 scenarios.



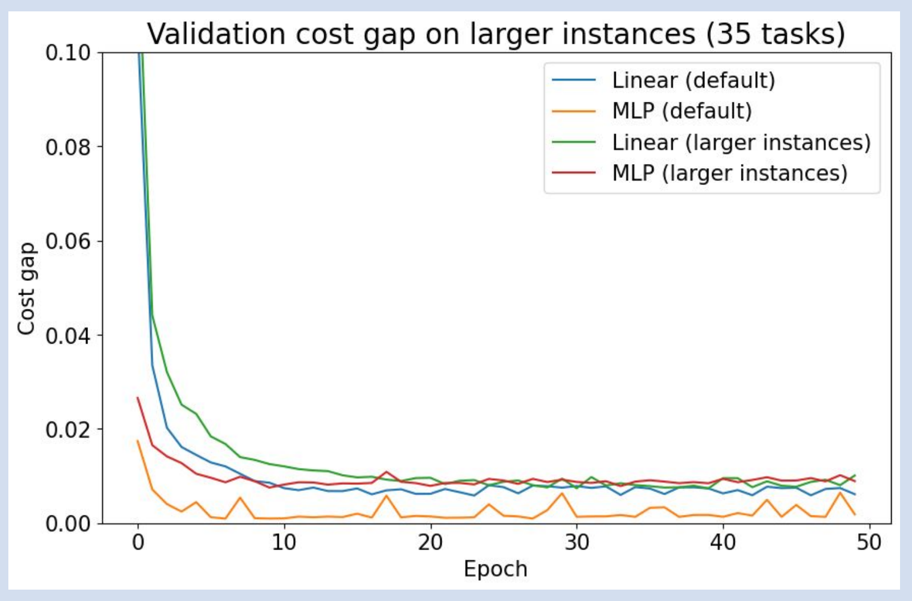
Using a two-layer MLP to learn the edge weights works better than a simple linear layer.



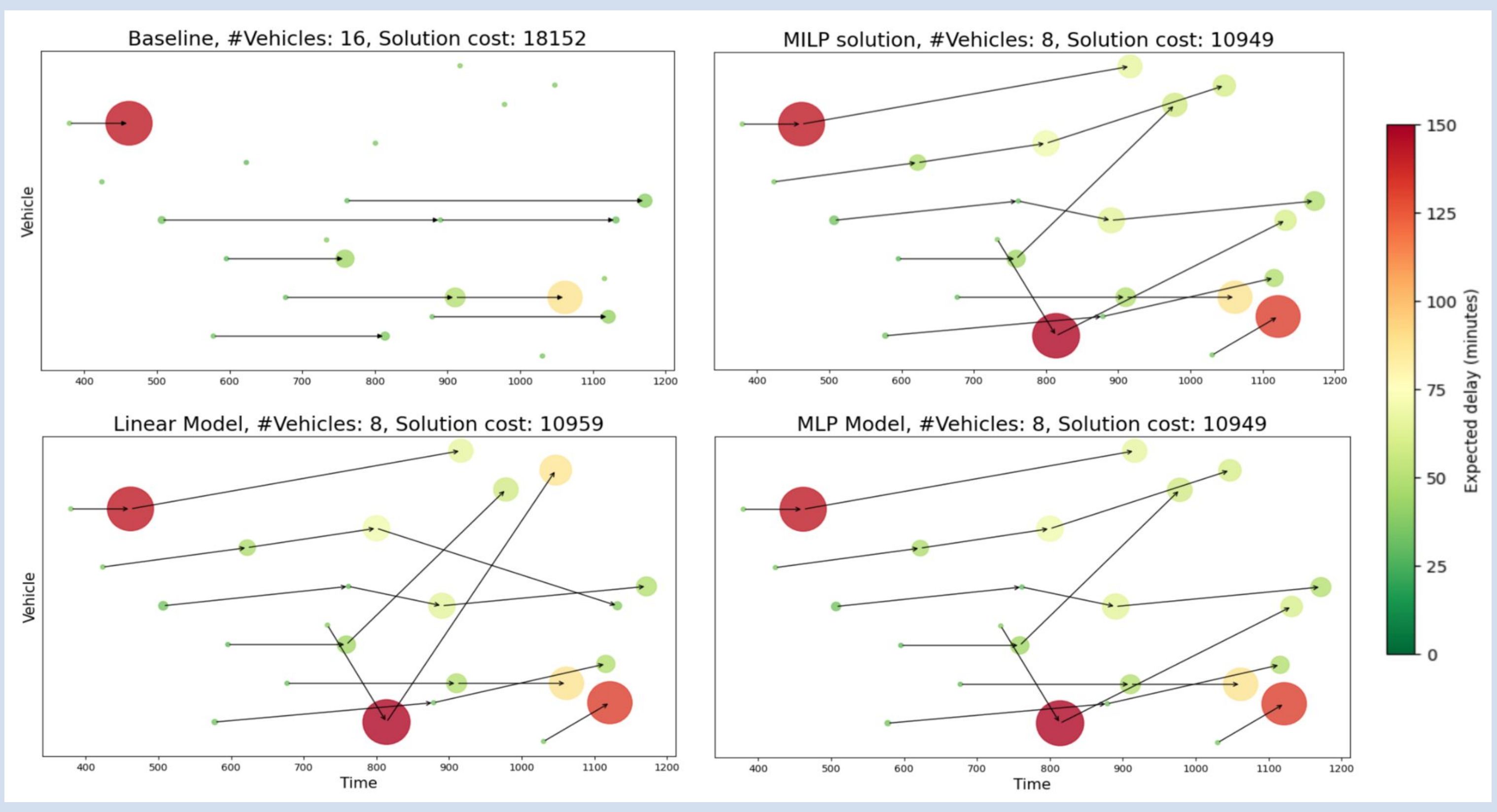
With a linear layer, CDF features are important. This indicates that the model needs to be non-linear.



Including neighbouring features performs as well as for linear layers, but worse for MLP. The MLP is likely overfitting.



The model trained on 25 tasks generalizes to larger instances with 35 tasks, but seems to loose a bit of performance.

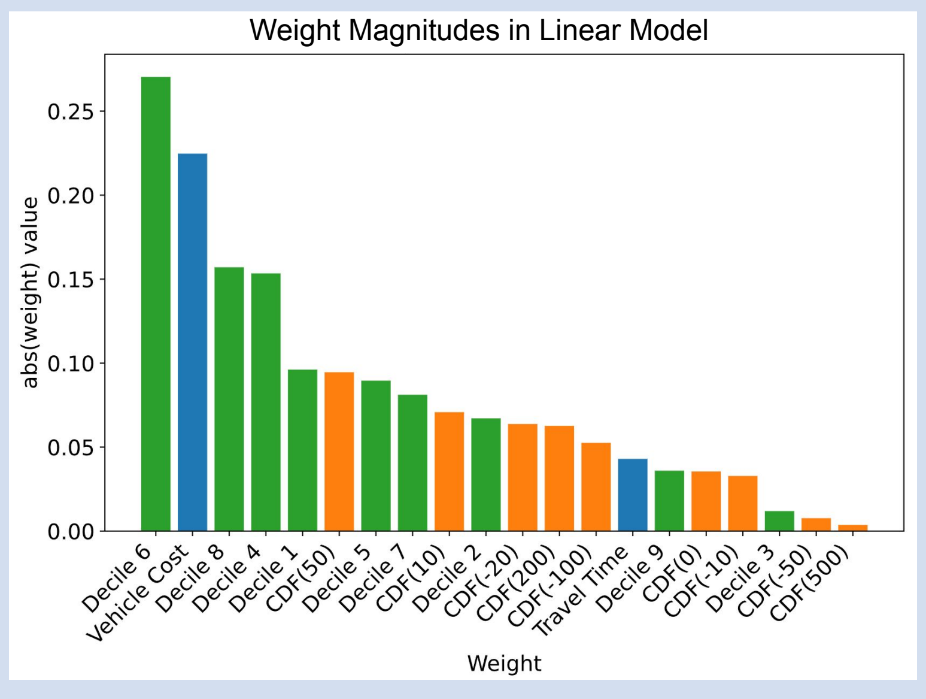


Comparison of predictions of four different models on the same 25-task instance. Each circle represents a task, with color value and size corresponding to its total delay (intrinsic + propagated) in the predicted solution. Tasks are ordered with respect to start time along the x axis. Each arrow path represents a vehicle path in the predicted solution.

Method	Mean (s)	Std (s)	Speedup
MILP Solver	102	114	1x
Learned Pipeline	2.33	0.37	43.8x

Runtime comparison for solving the SVSP using the MILP Solver vs. the learned pipeline (25 tasks).

- We were able to reproduce the results of the paper with respect to the average cost gap our model achieves.
- As seen above, non-linearity in the model is important.
- Models trained on smaller instances generalize to larger instances.
- (See figure on right) Vehicle cost seems to be more important than travel time; The decile features have higher magnitudes than the CDF features.



Deciles and the CDF are with respect to the slack variable

References

1. Dalle, G., Baty, L., Bouvier, L., & Parmentier, A. (2022). Learning with combinatorial optimization layers: a probabilistic approach. arXiv preprint arXiv:2207.13513.
2. Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J. P., & Bach, F. (2020). Learning with differentiable perturbed optimizers. Advances in neural information processing systems, 33, 9508-9519.

Contributions

Fabio: Wrote training code and runtime measurement script. Together with Tobias wrote sections 1,2,3 of poster.
Tobias: Focused on implementing the LP and MILP solver. Together with Fabio wrote sections 1,2,3 of poster.
Leonard: Worked on dataset generation code with Han. Fine-tuned training code and fixed bugs, such that we can reproduce the paper results. Ran experiments with different configurations (first half of section 4).
Han: Contributed to the dataset generation code together with Leonard, bug fixes. Wrote code for visualizing the solutions of different models (section 4) and for visual debugging.