Transformer network for the Dial-A-Ride problem

Pawel Mlyniec, Mauro Staver, Maxim Ryakhovskiy
School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—The Dial-a-Ride Problem (DARP) is a complex combinatorial problem considered in the operational research field. Given a set of drivers and a stream of tasks, where each task is a ride between two points under some time constraints, one has to find a mapping that assigns a driver to the given task. This problem is highly relevant in today's economy as delivery services have gained large popularity and become widespread. It is usually solved by using hand-crafted heuristics inspired by solutions for the Travelling Salesman Problem. In this paper, we propose a transformer network capable of cloning a supervision policy on DARP. This provides evidence that such an architecture is capable of understanding the intricacies of DARP and thus similar architectures can likely be applied to other NP-hard problems in operational research.**

I. Introduction

The Dial-a-Ride (DARP) is an Optimisation problem for flexible transport services. This problem formulation can be applied to shared taxi services, food delivery, and other delivery services. Each target needs to be picked up and dropped off at a certain position with constraints on pick up and drop off time. Due to strict time and space constraints, DARP belongs in an NP-hard class of problems.

The traditional heuristic solutions to such problems are quite problem-specific and take years to develop. Therefore, solving NP-hard problems by learning would be a breakthrough for a large number of industrial challenges. A recent paper [1] demonstrates that the transformer architecture can find optimal solutions to TSP instances of sizes 50 and 100. This is encouraging because it shows that transformers can capture the complexity of NP-hard combinatorial problems. Therefore, it makes sense to try to apply a similar architecture to DARP. This has been attempted in a recent unpublished paper [2] at EPFL and we will continue on this work.

The paper introduces a transformer architecture capable of cloning heuristic strategies using supervised learning. The open-source code-base associated with the paper was broken and highly complex, so we first fixed and refactored its main parts. The code is now highly modular, well documented, and should be easily built upon. Using the newly-refactored code-base, we were able to reproduce the results of cloning the Nearest Neighbor strategy on small instances of DARP, as described in the mentioned paper.

The DARP can be formulated as follows [3]:

Input

- A stream of m tasks $T = (t_i)_{i\in[1,m]}$ where $t_i = (p_{up}, p_{off}, \tau_{up}, \tau_{off})$ represents the time steps until which the target needs to be picked up τ_{up} or dropped off τ_{off} and the pickup p_{up} and drop off p_{off} positions.
- A set of n drivers $A = (a_i^k)_{i\in[1,n]}$ where $a_i^k = (p_a^k, C_i^k, \tau_{free})$ represents, at each time step k, the drivers position p_a^k, available capacity C_i^k and the next time step where the driver can choose a new target τ_{free}.

Output

- a mapping stream P associating each target to a driver, following all time, space and capacity constraints. $P = (< t_i, a^i_{F(i)} >)_{i\in[1,m]}$ with $F :\to [1, n]$

A stream of tasks represents passengers that need to be picked up and dropped off at certain locations under fixed time constraints. Meanwhile, the amount of drivers is limited and each driver is constrained by its capacity, current passengers, and current position. The output is a stream of associations between drivers and tasks which respects all constraints.

II. Methods

We first generate a supervision dataset by using random synthetic problem instances and some heuristic strategy. This dataset is fed into a transformer model as input for supervised learning. At the end the model is evaluated on small instances of the cordeau2006 benchmark dataset.

A. *Environment*

A Gym type environment is used to simulate a problem instance defined with passed parameters. Given some action, the environment applies all logical time and space constraints, performs the action, and returns the observation of the next time step. The observation contains all environment information at the current time step: positions of drivers, current capacities, target information, etc. Using environments, we construct a supervised dataset which maps each observation to an action chosen by the strategy we are trying to imitate. The environment is designed as a turn-based game where players are drivers, so an action is simply the id of the target that the current player is assigned to, or 0 if no targets should be assigned.

B. *Data Processing*

The observations need to be translated into appropriate embedding vectors that can be understood by the transformer model. We follow the same technique for generating embedding vectors as described in the original paper.

Each element of the observation is either an Integer or a point in two-dimensional space. All Integer values are processed by using a lookup table in order to obtain an embedding vector of the desired size. The two-dimensional points are passed through a dense MLP layer which returns the embedding vector of the desired size for positional data.

For the sake of homogeneity, each observation gets translated into three embedding vectors: one for positional data, one for time data, and one for other general environment data like target/driver ids, current player id, etc. These three vectors are then added before being processed by the transformer model.

C. *Supervised Learning*

We created the supervision dataset on random environment instances containing 2 drivers and 16 targets, with the the maximum capacity of each driver set to 3 and maximum allowed ride time of targets set to 30 time units. The speed of drivers is simply 1 space unit per time unit. Using a supervision dataset of 10,000 observation-to-action mappings, we commence supervised training on our transformer model for 500 epochs. We found this to be enough to get meaningful results on imitating the Nearest Neighbor strategy. For more complex strategies, both numbers should be higher.

III. Model

Transformers [4] were first introduced in 2017 by the Google research group. The goal was to design a highly performant model for sequential NLP tasks with the ability to handle long-range word dependencies. Unlike RNNs, transformers do not necessarily process the data in order. Rather, the attention mechanism provides context for any position in the input sequence which allows a global perception of the given DARP instance. Using this mechanism, each encoding of a driver will take into account other drivers encodings, which should result in globally optimal decision making. Another advantage of using transformers rather than recurrent or conventional networks is the high parallelizability they offer.

A. *Model Architecture*

In this project, we use the transformer model proposed by Ardoin et al. The model is highly similar to a bare classic encoder block. Systems like BERT [5] demonstrate that using only encoder blocks is almost as efficient as using full encoder + decoder transformers. The combined embedding vectors representing the problem are passed through the encoder resulting in an encoded version of the given problem instance. This encoded feature vector is then passed to a classifier which outputs the probability of each target being assigned to the current driver. The classifier's architecture represented in the scheme is a classical Linear Layer. We train this architecture (Fig 1) using cross-entropy loss according to the given supervision choice of action.

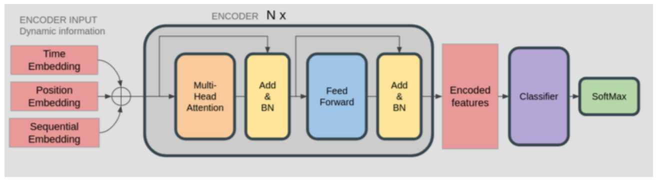

Figure 1. Transformer architecture used to solve DARP

IV. Experiments

We ran multiple experiments based on the smallest cordeau instance with 2 drivers and 16 targets, with the maximum capacity of each driver set to 3 and maximum allowed ride time of targets set to 30 time units. The speed of drivers is simply 1 space unit per time unit.

On each experiment, a new supervision dataset of 10,000 observation-to-action mappings was created based on randomly populated environment instances and the Nearest Neighbor policy. This dataset is then split into a training set and a validation set based on a 90-10 split. All experiments were run for 500 epochs and supervised using cross-entropy loss. To learn strategies more complex than Nearest Neighbor, we recommend using larger supervision datasets and more epochs.

The exact parameters used in experiments can be found in the main startup script located in our open-source repository [6]. After each training epoch, models were evaluated on the validation set.

A. *Reproducing results*

Using default parameters on a simple *a2-16* problem instance (2 drivers and 16 targets), we were able to quickly obtain around 100% accuracy on the test dataset. Indeed, it only took around 50 epochs to reach such accuracy, as seen in Fig 2. where the top plot shows accuracy on the training data and the bottom one on the test data. This is in line with the results obtained by the original author of the project we are building upon. From this result, we conclude that our pipeline works correctly and we can move to further experiments.

B. *Learning rate*

Results on experiments on a larger *a3-24* instance (3 drivers and 24 targets), proved to be largely affected by the learning rate. We tested learning rates of 0.001, 0.0005, and 0.0001 while keeping all other parameters as default (described above). Surprisingly, only the learning rate of 0.0001 resulted in an accuracy converging to around 100%. On the other hand, using other learning rates caused the

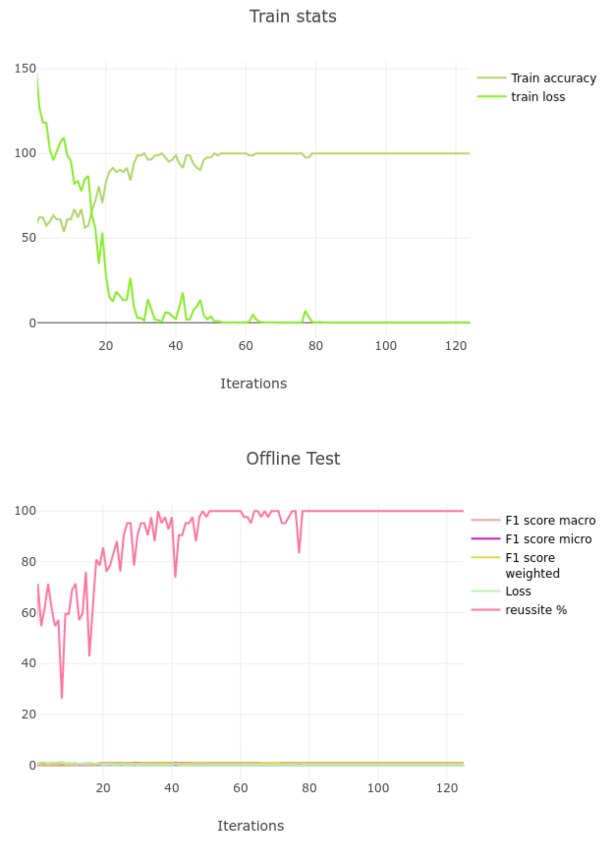

Figure 2. Train and Test accuracy for 2 drivers and 16 targets problem

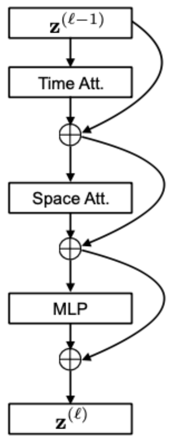

Figure 3. Timesformer encoding block architecture

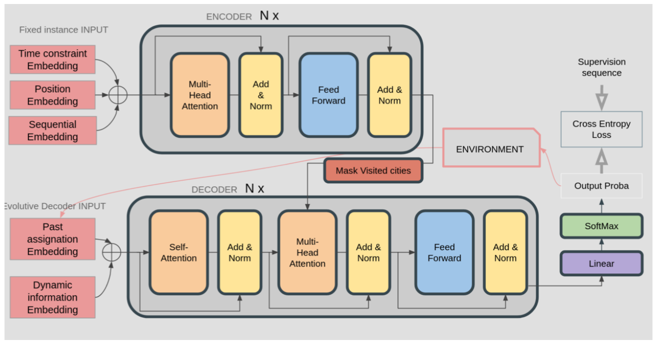

Figure 4. Encoder-Decoder architecture for DARP

accuracy to fluctuate around the initial value of around 80% with no progress. Therefore, we kept the learning rate of 0.0001 in all other experiments.

C. *Transformer hyperparameters*

We performed a manual grid search on small problem instances to find the best transformer hyperparameters. The best configuration seems to be 4 attention heads, 6 layers, and a dropout rate of 0.1. This configuration is highly similar to the default parameters proposed in the original transformer paper [4] and offers the fastest convergence to high accuracy.

D. *Results*

By tuning certain parameters, as described above, we were able to perfectly reproduce the NN strategy on small problem instances. This demonstrates that the model is capable of capturing the complexity of a given problem instance and that data processing is coherent with the model architecture. That being said, the NN strategy is a naive strategy, and reproducing more complex strategies might not yield the same results.

V. Possible improvements

- **TimeSformer** [7] – is a novel transformer model used for video analysis. As our model, it is also only composed of encoding blocks visible in Fig 3. The difference lays in separate attention computation for time and space. Bertasius et al. have demonstrated that divided space-time attention performs better than combined one. DARP also has time-space constraints, hence it should be worth trying to use the Timesformer architecture.
- **Encoder-Decoder architecture** – using the full power of the transformer architecture could allow improved model prediction. A similar architecture [1] as the one described in Fig 4. was used to solve TSP instances with great success.
- **data representation and processing** – the limitation on the accuracy for larger DARP instances could come from the lack of variety in generated instances. A good data augmentation process can greatly increase the variety and hopefully increase the overall accuracy.

VI. Conclusion

We have managed to reproduce results in line with the previous paper. Our model seems to capture the intricacies of the given DARP instance quite well which is encouraging for future research on this topic. Even though our results are promising, the nearest neighbor supervision strategy is trivial and the model is far from usable in real-world scenarios. A lot of work still needs to be done to allow efficient replication of more complex strategies. Optimizing a transformer architecture to make it competitive on larger problem instances could revolutionize the Operational Research field and possibly open a door for a new wave of successful applications of transformers in the industry.

VII. Issues

While working on the project we encountered several issues. The open-source codebase associated with the paper we were building upon was broken, highly complex, and unorganized. After a while, we managed to set a meeting with the author of the code where he walked us through the code and gave us some tips on how to continue. Our next problem was obtaining the *Gurobi* license for every node on the cluster and the fact that the runtime of one proper experiment was around three days. Considering all, we have given up on trying to replicate the *Restricted Fragment* supervision algorithm and instead went for a simpler approach: replication of the *Nearest Neighbor* algorithm.

For simple instances we managed to replicate the results but for bigger instances of the problem, the model is not always converging to the result.

VIII. Acknowledgements

Thanks to the author of the previous paper Thibaud Ardoin for agreeing to meet with us and being kind enough to offer extensive support in dealing with his code. Thanks to prof. Alahi for mentoring and guiding us through this project.

References

[1] X. Bresson and T. Laurent, “The transformer network for the traveling salesman problem,” *ArXiv*, 2018.

[2] A. A. Thibaud Ardoin, “The dial-a-ride problem (not) solved with transformers,” 2021.

[3] J.-F. Cordeau and G. Laporte, “The dial-a-ride problem (darp): Variants, modeling issues and algorithms,” *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2003.

[4] N. P. J. U. L. J. A. N. G. L. K. Ashish Vaswani, Noam Shazeer and I. Polosukhin, “Attention is all you need,” *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019.

[6] https://github.com/staverm/DARPwTransformers.

[7] G. Bertasius, H. Wang, and L. Torresani, “Is space-time attention all you need for video understanding?” 2021.