Learning to Deliver: a Foundation Model for the Montreal Capacitated Vehicle Routing Problem

Samuel J. K. Chin, Matthias Winkenbach
Massachusetts Institute of Technology, Center for Transportation & Logistics, 77 Massachusetts Ave, Cambridge, MA 02139, jkschin@mit.edu, mwinkenb@mit.edu

Akash Srivastava
MIT-IBM Watson AI Lab, 314 Main St, Cambridge, MA 02142, Akash.Srivastava@ibm.com

Abstract. In this paper, we present the Foundation Model for the Montreal Capacitated Vehicle Routing Problem (FM-MCVRP), a novel Deep Learning (DL) model that approximates high-quality solutions to a variant of the Capacitated Vehicle Routing Problem (CVRP) that characterizes many real-world applications. The so-called Montreal Capacitated Vehicle Routing Problem (MCVRP), first formally described by Bengio et al. (2021), is defined on a fixed and finite graph, which is analogous to a city. Each MCVRP instance is essentially the sub-graph connecting a randomly sampled subset of the nodes in the fixed graph, which represent a set of potential addresses in a real-world delivery problem on a given day. Our work exploits this problem structure to frame the MCVRP as an analogous Natural Language Processing (NLP) task. Specifically, we leverage a Transformer architecture embedded in a Large Language Model (LLM) framework to train our model in a supervised manner on computationally inexpensive, sub-optimal MCVRP solutions obtained algorithmically. Through comprehensive computational experiments, we show that FM-MCVRP produces better MCVRP solutions than the training data and generalizes to larger sized problem instances not seen during training. Even when compared to near-optimal solutions from state-of-the-art heuristics, FM-MCVRP yields competitive results despite being trained on inferior data. For instance, for 400-customer problems, FM-MCVRP solutions on average fall within 2% of the benchmark. Our results further demonstrate that unlike prior works in the literature, FM-MCVRP is a unified model, which performs consistently and reliably on a range of problem instance sizes and parameter values such as the vehicle capacity.

Key words: deep learning, foundation model, vehicle routing, transformers, combinatorial optimization

1. Introduction

The Capacitated Vehicle Routing Problem (CVRP) is one of the most well-researched Combinatorial Optimization (CO) problems in the Operations Research (OR), transportation, and logistics literature, given its many commercial and non-commercial applications. To this day, it continues to be an active area of academic research with advances being made in two main directions. First, researchers seek to develop new methodological approaches either to solve existing variants of the problem more efficiently, and for larger problem instances, or to uncover new best known solutions to benchmark problem instances from the literature (Uchoa et al. 2017, Arnold et al. 2019). Second, inspired by increasingly sophisticated routing challenges faced by the transportation and logistics industry, authors define and investigate new and more advanced variants of the problem to capture additional constraints and nuanced problem characteristics (cf., Vidal et al. 2020).

Methodologically, the OR literature on the CVRP and its many variants can be grouped into (i) exact methods, including Mathematical Programming (MP), (ii) heuristic approaches, and (iii) metaheuristic approaches. The dominant exact method to solve CO problems such as the CVRP is MP. However, the NP-hard nature of these problems (cf., Cook et al. 2011) quickly renders such models intractable for realistically sized problem instances. Therefore, for solving large-scale problem instances often faced in real industrial applications, most state-of-the-art solution methods rely on heuristic and metaheuristic approaches (Arnold and Sörensen 2019, Vidal 2022). Most notably, the Lin-Kernighan-Helsgaun-3 (LKH-3) heuristic by Helsgaun (2017) and Hybrid Genetic Search (HGS) (Vidal 2022) are currently among the best performing solution approaches to the CVRP (cf., Table 5 in Vidal 2022).

Machine Learning (ML) methods, on the other hand, dominated the fields of Computer Vision (CV) and Natural Language Processing (NLP) in recent years. Further, ML has found numerous highly visible applications – from strategic game playing (see, e.g., Silver et al. 2018) to molecular biology (see, e.g., Jumper et al. 2021). More recently, publicly discussed generative Artificial Intelligence (AI) models such as Chat Generative Pre-trained Transformer (ChatGPT) have enabled human-like conversations with a machine. The rapidly advancing algorithmic performances are constantly pushing the boundaries of achievable state-of-the-art results. As these problems in general have an extremely large state space, a natural next question in research is to what extent ML can be applied to CO problems and complement or even replace existing OR methods (Bengio et al. 2021). Specifically, Pointer Networks (PNs), first proposed by Vinyals et al. (2015), demonstrate how a Neural Network (NN) can approximate solutions to the Traveling Salesman Problem (TSP). This work has since inspired numerous approaches to solving routing problems with Deep Learning (DL) (see, e.g., Nazari et al. 2018, Lu et al. 2019, Kool et al. 2019, Kwon et al. 2020). The idea of leveraging NNs to solve CO problems such as the TSP is not entirely new though. For instance, Hopfield and Tank (1985) were the first to solve small TSP instances with a so-called Hopfield network.

1.1. Contributions of This Work

There are four main contributions of this work. First, we show that the recent methodological advances in the field of Large Language Models (LLMs) can be applied successfully to solving combinatorial optimization problems in the transportation and logistics domain, such as vehicle routing. Specifically, we propose the Foundation Model for the Montreal Capacitated Vehicle Routing Problem (FM-MCVRP), a novel supervised DL model that approximates near-optimal solutions to the Montreal Capacitated Vehicle Routing Problem (MCVRP), a particularly relevant variant of the CVRP, which mimics many real-world delivery problems (cf., Bengio et al. 2021). Unlike recent work from Kool et al. (2019) and Kwon et al. (2020) who build on the Transformer architecture (Vaswani et al. 2017) in a PN framework, FM-MCVRP is to the best of our knowledge the first DL model leveraging the Transformer architecture in an LLM framework for routing problems. Building on the Text-to-Text Transfer Transformer (T5) model by Raffel et al. (2020), which has been largely applied to NLP problems, we model the underlying routing problem as an analogous NLP task that predicts the nodes to visit defined on a fixed graph instead of using a PN.

Second, our proposed model is a unified model that accepts a range of problem sizes and vehicle capacities. This makes our model highly applicable to real-world problems instead of only performing well on stylized problem instances with tightly controlled parameters. Previous works, such as Kool et al. (2019) and Kwon et al. (2020), train a single model for each given combination of problem size and vehicle capacity, which seems impractical for real-world operations as companies do not have the capability to manage a large number of ML models that are trained for each possible combination of problem size and capacity they might encounter. Moreover, we can show that our proposed model can solve up to 800-node problem instances, compared to Kool et al. (2019) and Kwon et al. (2020) who solve up to 100-node problem instances as larger instances result in divergence in the training process (see Appendix C). However, many real-world delivery problems, especially in last-mile logistics, contain more than 100 customers.

Third, we show that our proposed model is able to outperform the algorithmically obtained solutions it was trained on. This finding has strong managerial implications. Given the size and complexity of many real-world routing problems, and due to a lack of expertise and ability to invest into high-performing algorithms explicitly tailored towards their specific problem, many companies rely on route planning tools based on relatively simple algorithms, decision rules, and human experience. Hence, many companies continue to amass large quantities of potentially sub-optimal route sequences that are obtained either algorithmically or by observing how routes are actually executed by drivers. Since the FM-MCVRP is agnostic to the method used to obtain the solutions on which it is trained, it allows companies to improve route quality by leveraging existing routing data. This means that they can gradually improve the quality of their routes by learning from readily available data without having to invest into designing and maintaining an explicit algorithm tailored to their business needs. Notably, our model is able to find solutions that are competitive to and may even outperform state-of-the-art benchmark methods on certain problem instances, despite being trained on inferior, yet computationally inexpensive solutions.

Fourth, we show that FM-MCVRP generalizes well to larger problem instances. Specifically, we train our model on 20 to 400-node problem instances before testing it on 600 and 800-node problem instances. We show that FM-MCVRP is able to consistently maintain its favorable performance characteristics for these larger problem instances. The ability to generalize to larger, and previously unseen instance sizes while maintaining a high solution quality has important implications for many real-world routing problems. The number of stops on a route may vary widely depending on the commercial, geographical, and operational context (e.g., product and customer characteristics, stop density, vehicle technology, etc.). While a company's training data may be skewed towards certain instance sizes, the trained FM-MCVRP can still yield good results for previously unseen or less prominent instance sizes. Similarly, a company may be able to train FM-MCVRP on historical data from routes it currently performs and leverage the trained model to plan routes even as the demand profile of it clients or the operational and technological context of its routes changes over time.

The remainder of this paper is structured as follows. In Section 2, we review relevant literature for solving or approximating solutions to the CVRP. We cover exact methods, heuristics and metaheuristics, DL approaches, and finally decoding strategies for DL approaches. In Section 3, we formally define the CVRP and revisit the formal definition of the MCVRP presented by Bengio et al. (2021). In Section 4, we detail our methodology and present our FM-MCVRP model. In Section 5, we describe the setup of our computational experiments. The results and insights from these experiments are presented in Section 6. In Section 7, we provide the managerial insights. We conclude with a summary of our work and an overview of what we consider fruitful directions for future research in Section 8.

2. Related Literature

In the following, we provide an overview of current state-of-the-art methods to solve the CVRP and related problems both from an OR and from an ML perspective. To frame the discussion of CVRP heuristics, we follow the definition from Toth and Vigo (2002) which broadly distinguishes three categories: construction, two-phase, and improvement methods. Two-phase methods can be further classified into *cluster-first, route-second* methods and *route-first, cluster-second* methods. The performance of the algorithms within each category differ and it is important to be cognizant of the class to which an algorithm belongs to and compare algorithms within its class. Given the nature of our proposed method, we focus our literature review on construction methods.

We first cover the best performing exact methods in Section 2.1 before providing an overview of state-of-the-art heuristic and metaheuristic approaches in Section 2.2. In Section 2.3, we then survey the most promising DL approaches to common routing problems. Section 2.4 provides additional insights into state-of-the-art decoding strategies, which are a critical determinant of the performance of learning-based approaches to CO problems. In Section 2.5, we distill the research gap that this paper is intended to fill.

2.1. Exact Methods

Exact methods in OR primarily involve MP, and so-called matheuristics, which integrate MP into metaheuristic frameworks for enhanced problem-solving efficiency. In this section, we first discuss MP approaches, followed by matheuristic approaches. A recent advancement in MP approaches to finding exact solutions to the CVRP has been the introduction of Branch-Cut-and-Price (BCP) by Pecin et al. (2017). The authors combine a myriad of innovations previously proposed in the literature on branch-and-price methods with so-called Limited Memory Subset Row Cuts (lm-SRCs), which enable significant efficiency gains in the pricing subproblem. They use their method to efficiently solve CVRP instances of up to 360 customers, which is a relevant instance size for many real-world industry applications. Pessoa et al. (2020) further extend the work of Pecin et al. (2017) for a more general definition of the Vehicle Routing Problem (VRP). For a recent in-depth review of exact BCP methods, we refer the reader to Costa et al. (2019).

Matheuristics on the other hand combine MP and metaheuristic approaches (see Section 2.2). Queiroga et al. (2021) propose a matheuristic coined Partial Optimization Metaheuristic under Special Intensification Conditions (POPMUSIC) for the CVRP. Using best known solutions from the CVRP benchmark instances of Uchoa et al. (2017) as an initial solution, their method is able to find new best known solutions for several instances by breaking them down into subproblems that are solved using a modified BCP algorithm as a heuristic. More recently, Skålnes et al. (2023) propose another state-of-the-art matheuristic. The authors use a construction heuristic to create an initial solution which, in turn, is refined by an improvement heuristic. The refined solution then serves as a starting solution for an exact branch-and-cut algorithm. The improvement and branch-and-cut steps are repeated until an optimal solution is found or another termination criteria is met. The method of Skålnes et al. (2023) is able to obtain the currently best known solutions for CVRP instances with more than 10,000 customers from the dataset of Arnold et al. (2019). For a comprehensive review of prior works on matheuristics, we refer the reader to Archetti and Speranza (2014).

2.2. State-of-the-Art Heuristics and Metaheuristics

The CVRP is an NP-hard problem rendering large-scale problem instances that correspond to many real-world applications intractable. Therefore, efficient approximate solution techniques are required. State-of-the-art approaches to solving large CVRP instances frequently involve a local search heuristic incorporated in a metaheuristic framework (Vidal 2022). As a CVRP solution consists of multiple TSP tours, TSP local search heuristics are commonly employed in solving CVRPs. Commonly employed TSP local search heuristics include the 2-OPT, 3-OPT (Croes 1958), the Lin-Kernighan (LK) heuristic (Lin and Kernighan 1973) and the Lin-Kernighan-Helsgaun (LKH) heuristic (Helsgaun 2000). Note that LK and LKH are the generalizations of the 2-OPT and 3-OPT local search operator to the k-opt operator. As LKH is only applicable to the TSP, Helsgaun (2017) propose LKH-3, an extension of LKH specifically for CVRPs. Common metaheuristic frameworks that have been successfully applied to the CVRP are Simulated Annealing (SA) (Bertsimas and Tsitsiklis 1993), Genetic Algorithms (GAs) (Holland 1992), and Large Neighborhood Search (LNS) (Shaw 1998).

HGS (Vidal 2022) is a current state-of-the-art metaheuristic that is predominantly an improvement method. It derives its remarkable performance from integrating a local search heuristic (SWAP*), which is essentially a classic *swap* operator without an insertion in place, in a GA framework. The HGS algorithm maintains a pool of feasible and infeasible solutions at all times. Following a typical GA logic, it first selects two solutions (genes) from this pool and combines the two solutions with an ordered crossover (Oliver et al. 1987) to obtain a new solution. The algorithm then conducts a controlled neighborhood search that explores both feasible and infeasible solutions to find a new local minimum. In the event where the resulting solution continues to be infeasible, a repair operator is applied with 50% probability. After that, the solution is either added to the feasible or the infeasible solution pool.

Knowledge Guided Local Search (KGLS) (Arnold and Sörensen 2019) and Slack Induction by String Removal (SISR) (Christiaens and Berghe 2020) are two other metaheuristic approaches that achieve competitive results (cf., Table 5 and 6 in Vidal 2022) on the benchmark instances from Uchoa et al. (2017). KGLS relies on penalization of arcs during the local search process. The penalties are computed based on features that were obtained from studying the common characteristics of good and bad solutions. SISR proposes a *destroy*, a *repair*, and a fleet minimization procedure, where the *destroy* operator has a novel property of spatial slack and the *repair* operator is categorized as a greedy insertion with blinks (Christiaens and Berghe 2020).

Among the heuristics and metaheuristics discussed in this section, HGS is the highest performing in terms of solution quality obtained on the benchmark instances of Uchoa et al. (2017) under a fixed time limit (cf., Table 5 and 6 in Vidal 2022).

2.3. Deep Learning Approaches

Most recently discussed advances in AI, such as ChatGPT and others, are built on DL methods. In this section, we briefly provide a few definitions to help the reader make the translation from terms and concepts originally developed in the context of other DL applications to the context of the CVRP. We then introduce the *Attention* mechanism and the seminal *Transformer* (Vaswani et al. 2017) that forms the basis of almost all state-of-the-art NLP and CV methods currently discussed. Following that, we introduce two other main NLP architectures: the Language Model (LM) architecture and the Prefix Language Model (Prefix LM) architecture. Finally, we discuss recent state-of-the-art DL approaches to solving the CVRP, and their relative performance compared to state-of-the-art heuristics and metaheuristics.

Definitions. In the DL literature, the term *embedding* refers to a vector representation of an object transformed from its original features. In the context of NLP, CV, and the CVRP, the object corresponds to a word, image, or node, respectively. In routing, a node could be expressed as a vector representing its (x, y)-coordinates, demand, and other relevant features. Note that while two closely co-located nodes with different demand quantities can be considered dissimilar when simply using a norm over its original features, transforming these nodes into their embeddings may uncover more deeply rooted similarities in their node characteristics, making them appear more similar to one another in the specific context of the underlying routing problem.

In the NLP literature (see, e.g., Vaswani et al. 2017), a *token* is generally analogous to a word in a sentence. In the context of routing, a token corresponds to a node to be visited at a specific position in a given route sequence.

Attention and Transformers. So-called *Attention* is now a commonly used mechanism in DL that was first introduced by Bahdanau et al. (2014), which selectively places emphasis on specific parts of the input sequence. In their seminal *Transformer* paper, Vaswani et al. (2017) propose an extension to this mechanism (see Appendix A), which we are referring to in this paper. We can further distinguish two variants of the attention mechanism: *fully visible* attention, where all tokens are able to *attend* to each other, and *causal* attention (see Raffel et al. 2020), where tokens in earlier parts of a sequence are unable to *attend* to tokens in later parts of the sequence. Finally, the Attention mechanism can be augmented with Multi-Head Attention (MHA), which applies the Attention mechanism with different learned weight matrices (Vaswani et al. 2017).

The key building blocks of a Transformer model are an *encoder* and a *decoder* and thus it is also commonly known as the encoder-decoder architecture. The *encoder* is made up of a series of encoder blocks, which themselves contain MHA and feed-forward layers. Essentially, it is a function that takes as input a matrix of features and transforms the matrix into embeddings. The *decoder*, in turn, consists of a series of decoder blocks, which themselves contain MHA, causal MHA and feed-forward layers. It is a function that takes the encoder embeddings and node features in the partial solution as input and outputs nodes in an autoregressive manner. Expressed in the context of vehicle routing, the encoder learns a representation of all nodes in the underlying graph of the MCVRP and the decoder seeks to capture the distribution of solutions in the graph.

NLP model architectures. The field of NLP broadly consists of three main model architectures: the encoder-decoder as described above, the LM, and the Prefix LM. The LM architecture, which is commonly known as the decoder-only architecture, is first proposed in the Generative Pre-trained Transformer (GPT) model by Radford et al. (2019). In this model, *causal* attention is applied to the model such that a token at any given position can only see the previous tokens in the input and not future tokens.

The Prefix LM is essentially a modified version of the decoder-only architecture. Here, the model has a prefix section where all tokens have *fully visible* attention. For more details, we refer the reader to Raffel et al. (2020). In the context of a MCVRP, the classic encoder-decoder and Prefix LM are the most relevant architectures, as the problem is defined on a fully connected graph in which all nodes are fully visible (i.e., connected) to each other. While this is consistent with an encoder-decoder or a Prefix LM architecture, the causal mask in an LM architecture would limit the nodes' visibility of each other.

Deep Learning for routing problems. Vinyals et al. (2015) are the first authors to present DL methods that attempt to approximate solutions to the TSP. Their method, PNs, is a Recurrent Neural Network (RNN) based method inspired by Sequence-to-Sequence (Seq2Seq) models in NLP. It solves the TSP by taking an input sequence of nodes and outputs a sequence of nodes through attending to the input nodes.

Kool et al. (2019) connect the the idea of PNs presented by Vinyals et al. (2015) with the Transformer model presented by (Vaswani et al. 2017), and are the first to demonstrate the use of Transformers to solve the CVRP. The input to the encoder in the model from Kool et al. (2019) is a set of nodes and the solution to the CVRP is produced in an autoregressive manner, similar to the original Transformer paper by Vaswani et al. (2017). Further, the authors use REINFORCE (Williams 1992), a policy gradient algorithm from Reinforcement Learning (RL), to train the Transformer.

Kwon et al. (2020) extend the work from Kool et al. (2019) and use the exact same architecture, with the two main differences being that the authors use a modified REINFORCE (Williams 1992) algorithm and unit square transformations like rotations and reflections to introduce equivalent representations of a given graph.

The methods from Kool et al. (2019) and Kwon et al. (2020) parameterize their model with Transformers and train their models with RL and are thus referred to as Deep Reinforcement Learning (DRL) methods. In addition, they both use the PN mechanism and attempt to solve random CVRP instances. In general, their methods do not yet achieve state-of-the-art results in terms of both solution quality and speed when compared to the methods discussed in Section 2.1 and 2.2. For instance, Kool et al. (2019) finds solutions with an average optimality gap of 3.72% for 100-node CVRPs instances and takes 0.72 seconds on average to run on a GPU while Kwon et al. (2020) finds solutions with an average optimality gap of 0.32% and takes 0.01 seconds on average to run on a GPU. In comparison, HGS obtains an average optimality gap of around 0.40% for CVRP instances with 100 to 330 nodes in less than three seconds (Vidal 2022).

2.4. Decoding Strategies

Recent NN-based approaches to solving routing problems model the conditional probability distribution that describes the likelihood of any given node of the problem to occur at each position of the solution sequence conditioned on the problem nodes and previously decoded solution nodes. As a conditional probability is modeled, sampling methods are often employed to decode a solution in an autoregressive manner. Adopting a sampling method naturally leads to non-deterministic solutions as we obtain a different solution trajectory for every iteration of decoding.

During decoding, the various solution trajectories generated naturally form a tree structure. When exploring this tree during decoding, the search space grows exponentially based on the depth of the tree, which in our context is the length of the solution sequence. This typically renders an exhaustive tree search computationally expensive for larger problem instances. Therefore, finding a good solution sequence with a conditional probability model requires an efficient *decoding strategy* that adequately balances the trade-off between run time and quality of the sequence.

A *greedy decoder* represents a very simple decoding strategy by which only the highest-probability node is considered at each position of the solution sequence. A significant downside of this strategy is that it ignores any path dependencies during decoding. In other words, it does not consider solution trajectories that yield better solutions by choosing low probability nodes early on in the solution sequence to make high-probability nodes accessible in subsequent parts of the solution sequence.

Another common strategy that mitigates the shortcomings of a greedy decoder is a *beam search decoder* (Lowerre 1976), which is a form of modified Breadth-First Search (BFS). In this strategy, a predefined beam size k is chosen and the k highest probability sequences are explored at every level of the search tree while the remaining sequences are discarded. The main limitation of beam search is that it considers node probabilities at each level of the search tree in isolation, without any foresight into the probability distributions in subsequent levels of the tree. In response to this shortcoming, Lu et al. (2022) explore look-ahead heuristics that take into account future node probabilities.

Since the probability distributions of nodes at each position of the route sequence are available, one can also use a *sampling decoder*, where the node realization at each position is sampled from the distribution (see, e.g., Kool et al. 2019). Extensions to sampling include *top-k sampling* (Fan et al. 2018, Holtzman et al. 2018), and *sampling with temperature* (Fan et al. 2018). In Large Language Model Meta AI 2 (LLaMA-2), Touvron et al. (2023) employ *nucleus sampling (NS)* (Holtzman et al. 2020), which is one of the current state-of-the-art sampling-based methods. Since decoding is fundamentally a tree search, several authors also explore Monte Carlo Tree Search (MCTS) based approaches (see, e.g., Leblond et al. 2021, Choo et al. 2022).

2.5. Research Opportunities

In Section 2.1 and 2.2, we discussed state-of-the-art exact and heuristic methods for solving the CVRP. These methods have been extensively researched for the past few decades and involve handcrafting algorithms or local search operators. In Section 2.3, we reviewed state-of-the-art methods that use DL and/or RL to learn a policy that can approximate solutions to the CVRP. From our review of both streams of literature, we see two extreme paradigms emerging. On the one hand, OR methods have been refined with meticulous human engineering over decades to efficiently find solutions to CO problems such as the CVRP. On the other hand, ML methods depend on minimal human engineering but demand significant computational resources.

In the era before AlexNet (Krizhevsky et al. 2012), the canonical paper that sparked the DL revolution, the CV community used handcrafted methods like Scale Invariant Feature Transform (SIFT) (Lowe 2004) and Histogram of Oriented Gradients (HOG) (Dalal and Triggs 2005), to achieve state-of-the-art results. Post AlexNet, only DL methods have been able to achieve state-of-the-art results on current CV problems.

Analogously, an important research frontier is now the successful use of DL on CO problems. Developing a DL method outperforming traditional OR methods on a well-established CO problem could be on the horizon. We attempt to make a first step in this direction by filling a gap in the extant literature and applying the latest research on LLMs to the MCVRP (cf., Bengio et al. 2021). The MCVRP is a special case of the CVRP and constitutes a meaningful problem on which to focus our efforts as it has numerous impactful real-world applications and can be formulated in a manner that aligns well with LLM architectures.

3. Problem Definition

The general CVRP. Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a symmetric undirected graph where $\mathcal{N} = \{0, ..., n\}$ is the set of nodes in consideration and $\mathcal{A} = \{(i, j) : i, j \in \mathcal{N}, i \neq j\}$ is the set of arcs with no self-loops connecting the nodes. In the generalized definition of the CVRP, the arcs can be located anywhere in the service area, i.e., on a plane in two-dimensional Euclidean space. The graph is fully connected and traveling on an arc between node i and node j incurs symmetric costs, $c_{ij} = c_{ji}$. Node 0 is the depot node and all other nodes are demand nodes that have an associated static demand d_i and have to be visited exactly once. A vehicle with capacity C starts at the depot, visits a sequence of demand nodes and returns to the depot under the constraint that the total demand in this route cannot exceed C. Multiple routes are executed until all demand nodes are visited and the total distance traveled is the cost of the CVRP solution.

The Montreal CVRP. In this paper, we consider a special experimental setup for the CVRP referred to as the *Montreal problem* in the literature (see, Bengio et al. 2021). The MCVRP corresponds to many real-world routing problems in which the total set of potential stop locations (e.g., customer addresses) is fixed, given, and finite, while each particular instance of the problem contains only a subset of these potential stop locations with non-zero demands (e.g., the customers requiring service on a given day).

In a formal sense, the MCVRP adheres to the structure of the CVRP. The primary distinction between these two problems centers on the relationship between multiple instances of the respective problem. Specifically, for the CVRP, any set of problem instances operates on independent graphs. This implies that the stop locations, which the vehicle routes are required to cover, can be distributed arbitrarily across the service region, with no inherent overlap in stops among different instances of the problem. In contrast, for the MCVRP, each instance of the problem represents different manifestations of non-zero demands within a subset of the same fixed and given graph $\mathcal{G}'$ of size m', where $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$. Consequently, each instance of the MCVRP is confined to a subgraph $\mathcal{G}$ of size $m \ll m'$. This subgraph is defined as $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} \subseteq \mathcal{N}'$ and $\mathcal{A} = \{(i, j) : i, j \in \mathcal{N}, i \neq j\} \subseteq \mathcal{A}'$.

Following this formal definition of the MCVRP, we generate problem instances by first defining a graph $\mathcal{G}'$ of size $m' = 10,001$, which consists of 10K customer nodes and 1 depot node. Following that, we sample subgraphs $\mathcal{G}$ that range in size from $m = \{21, ..., 401\}$ (i.e., a 21 node problem instance contains 20 customer nodes and 1 depot node). We illustrate the data generation process in greater detail in Section 5.1.

4. Methodology

In natural language, words form sentences and sentences form paragraphs. When a human reads a piece of text, it is trivial for him or her to determine if the sentence is grammatically correct and makes sense. It is thus natural to wonder if the recent successes in LLMs can be directly applied to the MCVRP, where instead of training a model on a large database of natural language text, we train the model on large amounts of problem-solution pairs for the MCVRP produced by a state-of-the-art heuristic. We posit here that the

LLM can capture the probability distribution of solutions obtained from the heuristic and produce the order of visitation for any previously unseen set of nodes sampled from the fixed graph $\mathcal{G}'$. In the following subsections, we describe the relevant theory and our model architecture. Specifically, in Section 4.1, we detail the mathematical formulation of modeling a joint probability with Transformers. In Section 4.2, we then describe our model architecture and objective the model was trained on. In Section 4.3, we elaborate on the Curriculum Learning (CL) strategy, which improves efficiency in the training process and the quality of the model before outlining how solutions are obtained in Section 4.4.

4.1. Modeling Joint Probability with Transformers

Given an instance of a MCVRP, P, we propose a Transformer based encoder-decoder model that aims to learn a stochastic policy to select solutions to P. We first define a feasible candidate solution $\hat{S} = (\hat{s}_1, \ldots, \hat{s}_\ell)$, where ℓ is the length of the solution and $\hat{s}_i \in \{0, ..., m'\}$ are the respective node IDs to be visited at position i. The conditional probability of a feasible candidate solution to the MCVRP instance, $p_\theta(\hat{S} \mid P; \theta)$, is given by

$$p_\theta(\hat{S} \mid P; \theta) = \prod_{i=1}^{\ell} p_\theta(\hat{s}_i \mid \hat{s}_{i-1}, \ldots, \hat{s}_1, P; \theta), \tag{1}$$

where θ are the parameters of the model. Moreover, the individual conditional probabilities $p_\theta(\hat{s}_i \mid \hat{s}_{i-1}, \ldots, \hat{s}_1, P; \theta)$ are parameterized via our Transformer-based decoder model as

$$P_\theta(\hat{s}_i \mid \hat{s}_{i-1}, \ldots, \hat{s}_1, P; \theta) = H_{i,\hat{s}_i}, \tag{2}$$

where H_i is a vector that represents a finite discrete probability distribution over the node IDs in G' at token position i and $H_{i,\hat{s}_i}$ is a scalar that represents the probability of $\hat{s}_i$ being at token position i (see Section 4.2). Our model is trained via a supervised learning procedure on a corpus of training data $(\mathcal{P}, \mathcal{S})$ consisting of pairs of CVRP problems P along with their near-optimal solutions S^* and θ is optimized via Stochastic Gradient Descent (SGD) using AdamW (Loshchilov and Hutter 2019) to minimize cross-entropy loss.

4.2. Model Architecture and Objective

In defining our model architecture and the objective for optimizing the model, we build on the insights gathered by Raffel et al. (2020) and rely heavily on their definitions and notations. Specifically, Raffel et al. (2020) consider two distinct objectives for unsupervised pre-training. First, they pursue a *denoising objective* (see, Devlin et al. 2019) for which the inputs to the model are randomly *masked*, *corrupted*, or left unedited. Here, *masked* means that a placeholder token that is not a word is put at the corresponding position, while *corrupted* means that a random word is put at that position. Note that the denoising objective relies on an encoder-only architecture, which contains a fully-visible mask for the attention mechanism. Here, all tokens in the input are connected to each other. The model then attempts to predict the correct tokens that are masked, corrupted or left unedited at these positions. Second, they pursue an *LM objective* analogous to what we described in Section 4.1. As this models a conditional probability, a causal mask is used in the attention mechanism, such that a token at any given position can only view previous tokens and not future tokens. Through extensive experiments, Raffel et al. (2020) conclude that the combination of an encoder-decoder architecture with a denoising objective yields the highest performance on a set of benchmark NLP tasks (cf. Table 2 in Raffel et al. 2020). Given the generally superior performance exhibited by an encoder-decoder architecture in their analyses, we also adopt this architecture in our work. However, since a denoising objective is not suitable for our problem structure, we rely on an LM objective. We further elaborate on our architecture and the objective in the next paragraphs.

Encoder-decoder architecture. Following the insights from Raffel et al. (2020), we adopt the encoder-decoder architecture. We first introduce two concepts at a high level that are necessary to understand the encoder-decoder architecture: the embedding layer and the output layer. In NLP, words are represented as a one-hot vector and ML models typically operate on feature vectors that contain continuous values. Therefore, it is common to have a learned matrix of size $\mathbb{R}^{|G'|\times D}$ to transform the features into vectors with dimension D, where $|G'|$ is the cardinality of the fixed graph described in Section 3. The encoder-decoder then processes the vectors of size D through multiple encoder and decoder blocks and finally outputs embeddings of size D. To translate this representation back into the original vocabulary space, a learned matrix of size $\mathbb{R}^{D\times|G'|}$ transforms these embeddings and a softmax can be applied to obtain word predictions. The learned matrix of size $\mathbb{R}^{|G'|\times D}$ is commonly referred to as an embedding layer and the learned matrix of size $\mathbb{R}^{D\times|G'|}$ is commonly referred to as the output layer. In practice, these matrices share weights and one is simply the transpose of the other.

In this work, we make two slight modifications to the encoder-decoder architecture as our problem is not an NLP problem. First, as the inputs to our problem are already feature vectors with continuous values, and not one-hot vectors representing a vocabulary, the first embedding layer is unnecessary. Second, the embeddings from the encoder and decoder are both passed through the output layer for prediction of the corresponding node IDs. Therefore, the output layer is shared by both the encoder and the decoder.

Figure 1 provides an illustrative summary of our overall methodological approach.

Following the terminology used in Vaswani et al. (2017) and Raffel et al. (2020), the encoder consists of a series of encoder blocks, which contain fully-visible MHA and feed-forward layers. The decoder consists of a series of decoder blocks, which contain causal MHA, fully-visible MHA and feed-forward layers. In addition, the matrices E, F, G and H in Figure 1 are given by

$$\begin{aligned} F &= \mathrm{softmax}(E\cdot W_o), & E\in\mathbb{R}^{m\times D}, W_o\in\mathbb{R}^{D\times|\mathcal{G}'|}, \\ H &= \mathrm{softmax}(G\cdot W_o+M), & G\in\mathbb{R}^{n\times D}, W_o\in\mathbb{R}^{D\times|\mathcal{G}'|}, M\in\mathbb{R}^{D\times|\mathcal{G}'|} \end{aligned}$$

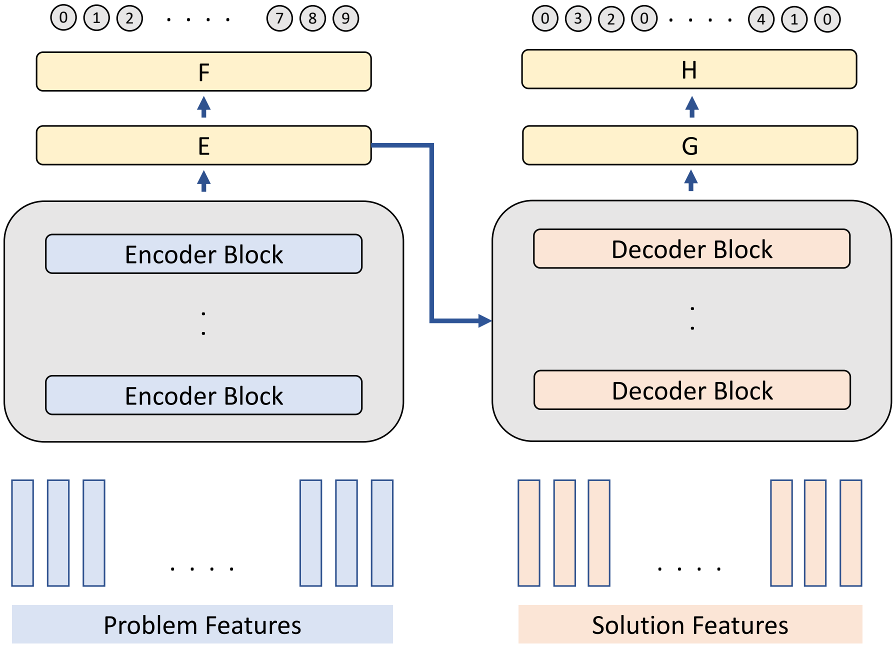


Figure 1 Encoder-Decoder Transformer Architecture for the CVRP.

where E represents the node embeddings output by the encoder after transforming the problem features via a series of encoder blocks; G represents the node embeddings output by the decoder after transforming the solution features via a series of decoder blocks, with E being used as the keys and values in the attention mechanism; W_o represents the shared output layer; m and n represent the total number of nodes in the problem and solution respectively, where $n > m$ as the solution involves returning to the depot; M represents a mask that is applied to prevent nodes that are not in the problem and infeasible nodes from being predicted.

Observe that E is used in the attention mechanism in the decoder, which directly follows the original Transformer paper (see, Vaswani et al. 2017, for more details). Further, observe how $E \cdot W_o$ and $G \cdot W_o$ are matrices of size $\mathbb{R}^{m \times |\mathcal{G}'|}$ and $\mathbb{R}^{n \times |\mathcal{G}'|}$, respectively, and the softmax is applied to the resulting matrices across the dimension corresponding to the size of $|G'|$ (i.e. the matrix of size $\mathbb{R}^{m \times |\mathcal{G}'|}$ has m rows that each sum to one). In particular, each vector in F and H represents a finite discrete probability distribution over the node IDs in G' at a given token position.

Objective. For the encoder, we are unable to use a denoising or an LM objective as the encoder takes as input the problem tokens of a subgraph $\mathcal{G}'$. It is easy to see why this is the case as applying a mask or simply replacing a token with another random token would result in an entirely different problem. Similarly, using an LM objective would impose a structure on the subgraph $\mathcal{G}'$ such that nodes do not have full visibility of each other. For the decoder, we follow Kool et al. (2019) and Kwon et al. (2020) and chose to define a model with an LM objective and thus are unable to use the denoising objective. Future extensions can model the decoder differently and enable the use of a denoising objective. In addition, inspired by CV techniques, we add data augmentations by rotating the entire subgraph $\mathcal{G}'$ by a random angle to enable the model to be invariant to rotations.

Finally, as we are predicting node IDs on both the encoder and decoder, we use the cross-entropy loss for a single token,

$$L(p,q) = -\sum_{n=0}^{|\mathcal{G}'|} p(n)\log(q(n)), \tag{3}$$

where L represents the cross-entropy between the true distribution p and the predicted distribution q, $|\mathcal{G}'|$ represents the cardinality of the subgraph $\mathcal{G}'$, and n represents the respective node IDs. In scenarios with one-hot encoded labels, $p(n) = 1$ for the correct node and 0 for all other nodes. $q(n)$ represents the predicted probability of the token belonging to node n, which is derived from the softmax output of the model. The cross-entropy loss is computed and the model is optimized via SGD with the AdamW optimizer (Loshchilov and Hutter 2019).

Inputs and outputs. The inputs to the encoder are the problem features and the inputs to the decoder are the solution features. We characterize each token (i.e., each node) with 9 features for consistency. These node features are $(x, y, d, t, \kappa, \gamma, \omega, c, a)$. x and y represent the respective coordinates of a node. Their values are normalized into a range of $[0, 1]$. d represents the demand of a node and is normalized into a range of $[0, 1]$, where a value of 1 corresponds to the maximum capacity of the vehicle. t is a binary variable that represents the type of the node, with a value of 1 for the depot and 0 otherwise. Future extensions of this work could include multi-depot scenarios, where each depot is represented by a one-hot vector. κ represents the normalized distance of a node to the depot. γ and ω respectively represent the cosine and sine angle of a node with respect to the depot. c represents the capacity utilization of the vehicle at any point in a particular solution and is in the range of $[0, 1]$, where a value of 1 represents that the vehicle is full. a represents the total normalized demand fulfilled in the entire network so far. Problem tokens can be fully described by $(x, y, d, t, \kappa, \gamma, \omega)$ while the other features are set to zero. Solution tokens are fully described by all 9 features.

4.3. Curriculum Learning Strategy

In this work, we leverage the CL strategy first proposed by Bengio et al. (2009). CL assumes that when trying to learn a task, humans and animals learn better when the examples are not presented randomly but in a certain order. Analogously, we are training an ML model by presenting it with training data in a predefined order. The CVRP lends itself to a natural ordering of the problem instances, where smaller problem instances are easier to solve than larger ones. There are two advantages in adopting the CL strategy. First, Bengio et al. (2009) hypothesize that CL enables faster convergence as well as an increase in the quality of minima obtained. Second, our model accepts inputs and outputs of varying sizes in batch training, which is achieved through padding. As such, the difference between the smallest and largest training sample determines the amount of padding required and this can cause a lot of unnecessary computation. With CL, we are able to reduce the amount of unnecessary computation as training samples with a smaller size have less padding. Further, CL enables us to use a larger batch size in early stages of training as the savings from additional padding permit more training samples in a single batch, allowing the model to potentially see more training samples within the same computational budget.

4.4. Obtaining Solutions

In Section 4.1, we describe how our method models the conditional probability of the next node to visit at a given token position, conditioned on the current partial solution tokens and all the problem tokens. To obtain solutions to new problem instances based on our trained model, we follow the existing literature which distinguishes two general approaches. First, we can employ a simple *greedy sampling (GS)* approach by taking the $\arg\max$ of the conditional probability vector obtained at each token position when decoding autoregressively. This approach is deterministic and computationally efficient as only a single sample of the trajectory is required. Second, we can follow a state-of-the-art sampling technique from NLP and employ *NS* (Holtzman et al. 2019), also known as *top-p sampling*, which first truncates the probability vector up to a threshold p, normalizes this distribution and then samples from it. This approach is non-deterministic and incurs a significantly higher computational cost, which depends on the number of trajectory samples. In return, a higher number of samples typically leads to better performance in terms of the quality of the solutions obtained. In addition, when sampling multiple solutions to a given problem instance, we also randomly rotate the graph to increase variability in the sampling process (i.e., some angles for a given graph may yield better solutions). As mentioned in Section 2.4, other potential decoding strategies include *beam search* (Lowerre 1976), *top-k sampling* (Fan et al. 2018, Holtzman et al. 2018) and *sampling with temperature* (Fan et al. 2018). In this work, we do not pursue these strategies and choose to follow LLaMA-2, a state-of-the-art LLM that uses *NS*. We also include *GS* as a simple baseline to assess the performance of the model.

5. Computational Experiments

In the following, we describe the large-scale experiments we conducted to demonstrate the performance and potential of our method in utilizing an LLM to generate high-quality solutions for MCVRP instances. In Section 5.1, we discuss our data preparation method. In Section 5.2, we discuss the model and training parameters used. Finally, in Section 5.3, we discuss the benchmarks utilized for comparing the performance of FM-MCVRP.

5.1. Generating Data

We primarily follow Nazari et al. (2018) in generating problem instances, a benchmark also adopted by Kool et al. (2019) and Kwon et al. (2020), which are considered canonical works for DRL on the CVRP. However, recall that Kool et al. (2019) and Kwon et al. (2020) train a distinct model for each problem size and fix the capacity. In contrast, our research contribution involves training a single model across various problem sizes ranging from 20 to 400 customer nodes and varying vehicle capacities. We choose this approach to mimic real-world delivery operations, which typically involve routes of varying lengths (e.g., due to varying customer density and drop sizes) and heterogeneous vehicle capacities (e.g., due to mixed fleets). In such a context, it is essential to be able to train and use a single unified model for the entire service area of a given city.

Problem instances. We first generate a graph $\mathcal{G}'$ of size 10K (not including the depot). The locations of the nodes in $\mathcal{G}'$ fall within a unit square, with the depot node being placed in the middle of the unit square. The locations of the customer nodes are randomly sampled from a uniform distribution over the unit square. Subsequently, a problem instance is formed by including the depot and sampling without replacement a set of customer nodes from $\mathcal{G}'$. The demand for each customer node is uniformly sampled from the set $\{1, ..., 9\}$. Note that our work deviates from previous works in the literature in the way the vehicle capacity is defined for a given problem instance. Unlike Nazari et al. (2018), Kool et al. (2019) and Kwon et al. (2020), who associate a specific vehicle capacity with each problem size, we choose capacity C for each problem instance by uniformly sampling within the ranges defined for any problem instance size n in Table 1.

Number of Nodes (n)	**Capacity** (C)
$20 \leq n < 50$	$30 \leq C < 40$
$50 \leq n < 100$	$40 \leq C < 50$
$100 \leq n < 200$	$50 \leq C < 60$
$200 \leq n \leq 400$	$60 \leq C < 70$
$401 \leq n \leq 1,000$	$70 \leq C < 80$

Table 1 The range of capacities corresponding to the number of nodes.

Problem-solution pairs. Following the definition of the MCVRP (see Section 3), we generate 100K unique problem instances on $\mathcal{G}'$ for every possible problem size in the range $\{20, ..., 400\}$ customers, leading to a total of 38.1M instances. For each problem instance, we obtain a solution using HGS (Vidal 2022) with a time limit of 5 seconds. We chose HGS over LKH-3 for our solution generation for two reasons. First, HGS is an algorithm with generally better solution quality over LKH-3 given a fixed time limit, as shown in Vidal (2022). Second, HGS enables us to control the time the algorithm is allowed to run for, whereas the run time in LKH-3 can only be controlled indirectly through the number of trials and runs.

We define T_i as the dataset containing all problem-solution pairs of size i. Further, we define $T_i^{\text{trunc}} \subset T_i$ as the dataset containing a random subset of 1K problem-solution pairs of size i. T_i^{trunc} is necessary for encoder pre-training (see Section 4.2) as we want the model to train on a large variety of problem sizes within a reasonable time.

The decision for a 5 second time limit being imposed on HGS is a hyperparameter that can be tuned. We base our choice of this parameter value on two factors: computational cost and solution quality. We follow Vidal (2022) and measure solution quality by the percentage gap of the solution compared to the best-known solution (BKS). This percentage gap is given by $\text{Gap} = 100 \times (z - z_{\text{BKS}})/z_{\text{BKS}}$, where z is the solution value of the algorithm and z_{BKS} is the BKS value for this problem instance. For data generation, we rely on 48 parallel processes on Intel Xeon Platinum 8260 processors (48 CPU cores) on a total of 8 nodes, giving us a total of 384 parallel processes during data generation. From the perspective of computational cost, generating solutions for 38.1M instances on the aforementioned compute infrastructure under a 5 second time limit per instance results in a total run time of slightly more than one day. From the perspective of solution quality, we did not want to generate solutions that are too close to optimality or the BKS. This is by design as we want to show that our proposed method can learn from a large dataset of good, but sub-optimal and hence relatively inexpensive solutions and outperform the quality of the solutions it has been trained on.

5.2. Model and Training Parameters

Model parameters. Both encoder and decoder in our model use MHA with 12 attention heads, 12 layers, an embedding dimension of 768, a feed-forward layer dimension of 3,072, a dropout probability of 0.1, and the final layer having a dimension of 10,001, representing the 10K potential customer nodes and the depot. With these configurations, our model has 206M parameters in total.

Training parameters. All our models are trained on 16 Tesla V100-PCIE-32GB GPUs on MIT SuperCloud (MITSC) (Reuther et al. 2018). As MITSC has a strict time limit of 96 hours for a job, we designed our training process with these constraints in mind. We leverage the CL strategy described in Section 4.3 and define a curriculum as

$$Cr_i = \bigcup_{j=t}^{i} T_j, \quad Cr_i^{\text{trunc}} = \bigcup_{j=t}^{i} T_j^{\text{trunc}}, \qquad\qquad 20 \leq i \leq 400,$$

where Cr_i and Cr_i^{trunc} contain 100K and 1K problem-solutions pairs for each size from size 20 to size i (inclusive) customer nodes, respectively.

Our training parameters closely follow Raffel et al. (2020) and are summarized in Table 2. At a high level, training can be broken down into two large phases: encoder pre-training (Phase I) and encoder-decoder finetuning (Phase II-A through II-C). As shown in Table 2, all parameters except the curriculum are essentially the same across Phase II. The proposed split is due to the 96 hour limitation on MITSC. Raffel et al. (2020) use a batch size of 128, whereas we use an effective batch size of 256 (batch size of 16 per GPU and a total of 16 GPUs), giving a learning rate scaling factor of $\sqrt{2}$ (see Appendix B). We opted to scale the learning rate only in the finetuning phase as the pre-training learning rates were sufficiently high in encoder pre-training. During pre-training, the learning rate follows the T5 schedule of $\frac{1}{\sqrt{\max(n,k)}}$, where n is the current training iteration and k is the number of warm-up steps. As we use 10K warm-up steps (Raffel et al. 2020), this means that the learning rate is kept constant at 0.01 for the first 10K warm-up steps and thereafter decays exponentially to 0.002. During finetuning, the learning rate is kept constant at $\sqrt{2} \times 10^{-3}$, which follows T5 as well but with an additional scaling factor. We also use the AdamW (Loshchilov and Hutter 2019) optimizer and clip gradients with a norm larger than 1.0. The optimizer, learning rate schedules and batch sizes were carefully chosen based on existing literature and we refer interested readers to Appendix B for details.

Phase	Curriculum	Model	BSZ/GPU	Peak LR	Min LR	Warm-up	Rotation	Time
I	$[Cr_{20}^{\text{trunc}}, \ldots, Cr_{400}^{\text{trunc}}]$	Enc	16	0.01	0.002	10K	No	52hr
II-A	$[Cr_{20}, \ldots, Cr_{50}]$	Enc-Dec	16	$\sqrt{2} \times 10^{-3}$	$\sqrt{2} \times 10^{-3}$	0	Yes	59hr
II-B	Cr_{200}	Enc-Dec	16	$\sqrt{2} \times 10^{-3}$	$\sqrt{2} \times 10^{-3}$	0	Yes	26hr
II-C	Cr_{400}	Enc-Dec	16	$\sqrt{2} \times 10^{-3}$	$\sqrt{2} \times 10^{-3}$	0	Yes	96hr

Table 2 Training parameters used.

5.3. Performance Benchmarks

Throughout our analysis, we compare the performance of our proposed FM-MCVRP against two state-of-the-art heuristics, HGS and LKH-3, and the method presented by Kool et al. (2019), which is a recent DRL approach and referred to as Attention Model (AM) in the following. Note that the publicly available weights for AM are trained for the general CVRP. Therefore, we need to retrain AM for the MCVRP. Since the model in AM caters to a unique problem instance size, we in fact retrain a separate AM for every instance size considered in our analysis. Further details on the retraining process can be found in Appendix C.

For each of these benchmark methods, we discuss model performance for instance sizes of 20, 50, 100, 200, 400, 600 and 800 customer nodes. To assess whether our observed model performance is systematic rather than just a coincidental artifact of the specific problem instances we are solving, we solve 1,000 problem instances for each instance size of 20, 50, 100, 200, and 400 customers, respectively. To avoid excessively large computation times, we reduce the number of instances solved to 100 for the larger problem instances of 600 and 800 customers.

Conditional probability models like our FM-MCVRP and AM generally utilize sampling methods to obtain the best results (see Section 4.4). In this work, we use NS (Holtzman et al. 2020) in FM-MCVRP and the default sampling method in AM, which samples from the conditional probability vector without modifications. While they do not rely on sampling per se, both HGS and LKH-3 use random seeds when finding an initial solution. Thus, the results from all of the methods we seek to compare are non-deterministic. Therefore, to ensure an equitable comparison of their performance, we generate 1, 100, and 1,000 solutions for any given problem instance with each of these methods, respectively. When discussing our results, we report the best found solution in terms of solution value for the currently discussed sample size.

Given that both HGS and LKH-3 are improvement methods, the quality of the solutions they find predominantly depends on the amount of run time they are granted before terminating the solution process. Since our experimental setup is built on the premise that FM-MCVRP is trained in a supervised manner on sub-optimal solutions, we choose to obtain these solutions by imposing a tight computational budget of 5 seconds on HGS.

For all of the LKH-3 results discussed in our analysis, we use the default configuration of the algorithm, as proposed by Helsgaun (2017), which sets the maximum number of trials to the number of customer nodes in the problem instance.

6. Results and Discussion

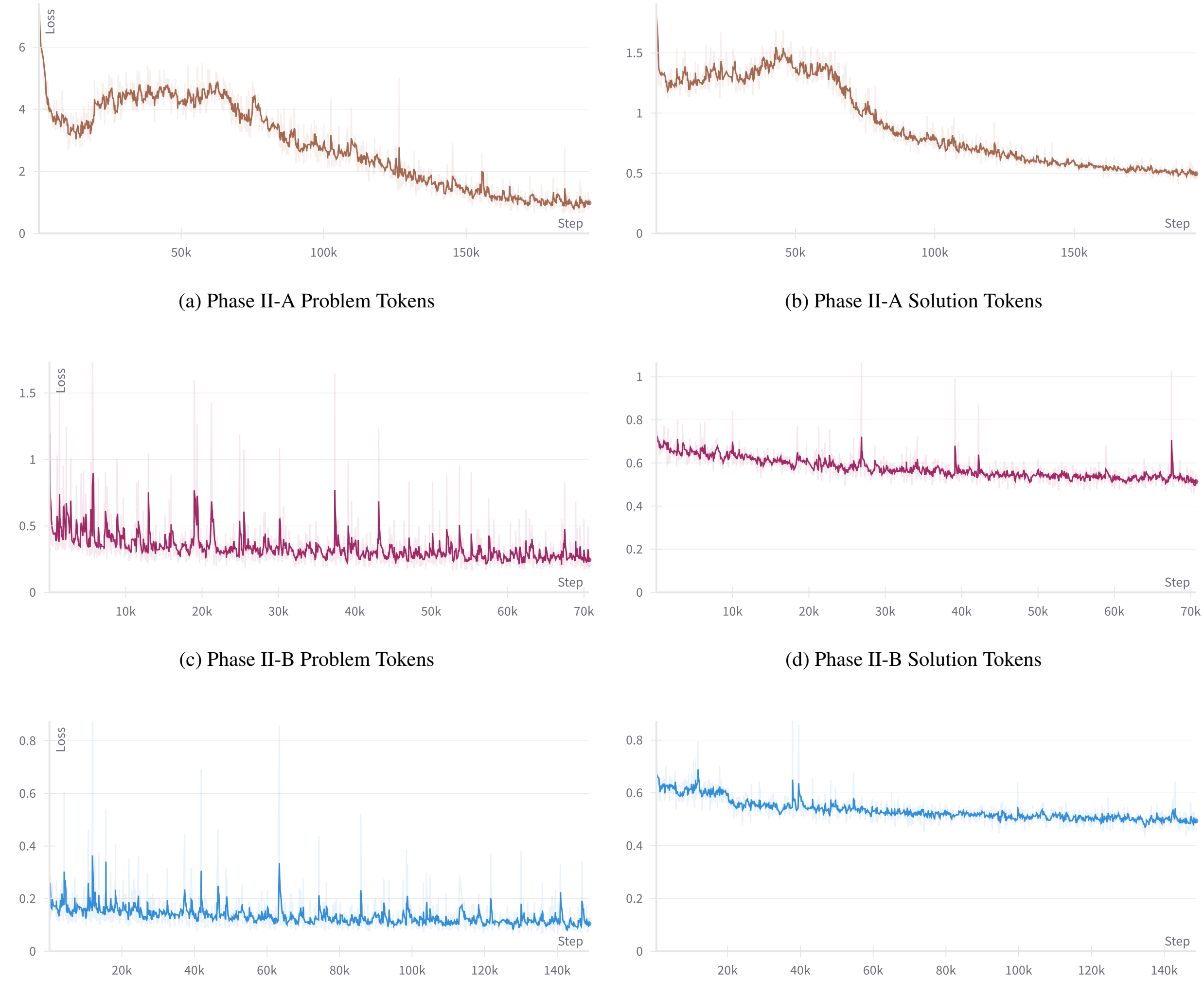
In the following, we present and discuss some of the most relevant results from our computational experiments. We structure our findings loosely according to the main contributions of our work, as stated in Section 1.

In Section 6.1, we discuss the double descent phenomenon that can be observed in our model convergence and its importance in avoiding premature termination of the training process for these types of models. In Section 6.2, we show that FM-MCVRP is able to produce solutions that are of higher quality than the solutions it was trained on. In Section 6.3, we further compare the distance distribution of solutions obtained with FM-MCVRP and HGS to demonstrate that FM-MCVRP is competitive with HGS even under a less restrictive computational budget. In Section 6.4, we show that FM-MCVRP can generalize well to larger problem instances. In Section 6.5, we extend our comparisons to LKH-3 and AM and show that FM-MCVRP is still competitive with recent state-of-the-art methods.

6.1. Convergence and Double Descent

A first key insight from our numerical analysis pertains to the existence of a pronounced double-descent behavior in the convergence of the training loss of FM-MCVRP. Following Raffel et al. (2020) and Touvron et al. (2023), Figure 2 shows the training loss obtained for the various training phases. Specifically, we highlight the loss curves in Phase II-A (Figures 2a and 2b) as we clearly observe the double descent phenomenon discussed by Nakkiran et al. (2021). This is an important insight to bear in mind when training large models as the convergence of these models can sometimes appear to plateau and thus trigger an early termination of model training when it is in fact going through double descent. In the context of our proposed model, for Phase II-A, after 20 hours of training, the loss appears to be plateauing and training might be terminated if one was not aware of the double descent phenomenon.

Observe in our training process that each successive phase in Phase II introduces new problem-solution pairs. For example, Phase II-C contains problem-solution pairs of size 201 to 400 customer nodes for the first time. Therefore, the training loss in Phase II-C (Figures 2e and 2f) is a rough approximation of the performance of the model as more than half of the samples have only been seen once. To interpret the loss



(a) Phase II-A Problem Tokens

(b) Phase II-A Solution Tokens

(c) Phase II-B Problem Tokens

(d) Phase II-B Solution Tokens

(e) Phase II-C Problem Tokens

(f) Phase II-C Solution Tokens

Figure 2 Training loss for problem (left) and solution (right) tokens in Phase II.

values, we remind the reader of the definition of cross-entropy loss we use in Equation (3). For problem tokens, the final loss converges to a value of approximately 0.09, which translates to identifying the correct node ID with a probability of 0.91 on average. For solution tokens, the final loss converges to a value of approximately 0.50, which translates to identifying each token in the HGS solution with a probability of 0.61 on average. This implies that the model is fairly confident of choosing a HGS solution from the training data but also has a relatively large margin of 0.39 on average to explore other nodes. An interesting extension of this work could include training the model with significantly more data and over a longer period of time and analyze the value to which the loss converges to.

6.2. Solution Quality Under a Tight Computational Budget

A second key result from our numerical analysis is that once trained, our proposed model can outperform the solutions it was trained on. As described in Section 5.1, we generate our training data by obtaining a single solution per problem instance with HGS under a time limit of 5 seconds for two reasons. First, it is computationally efficient to obtain HGS solutions under this computational budget and the solutions obtained are generally of good quality but typically sub-optimal. Second, we also purposefully aimed for this good yet sub-optimal solution quality in our training data to emulate the characteristics of many real-world routing datasets available to companies in practice. In Table 3, we compare the solutions obtained by FM-MCVRP with the sub-optimal solutions it was previously trained on. Here, we make three important observations.

First, similar to what Kool et al. (2019) and Kwon et al. (2020) observe, the gap in solution quality between the solutions found by FM-MCVRP and the solutions it was trained on decreases as the instance size gets larger. This implies that under a tight computational budget, the solution quality of a state-of-the-art heuristic such as HGS declines in instance size at a faster rate than the quality of the solutions found with our method.

Second, under an NS decoding strategy with 100 or $1,000$ samples, FM-MCVRP is shown to outperform the solutions it was trained on for large problem instances. Specifically, we show that for an NS decoding strategy with 100 $(1,000)$ samples, our method outperforms the training solutions on average for instance sizes of 200 (100) customers and above. Figure 3 further illustrates this point using an instance size of 400 customers as an example. The figure compares the solutions from FM-MCVRP under an NS decoding strategy with $1,000$ samples to the solutions obtained from running HGS once per problem instance with a 5 second time limit. FM-MCVRP solutions are shifted to the left compared to the HGS solutions with a mean relative difference of -1.05%. This improvement over the HGS solutions is statistically significant as confirmed by a one-sided paired samples t-test (Ross and Willson 2017) (see Appendix D).

Lastly, we note that for large problem instances, FM-MCVRP yields competitive solutions compared to a state-of-the-art heuristic such as HGS, when held to similarly restrictive computational time constraints. For instance, under a GS decoding strategy, FM-MCVRP finds solutions for 400-customer instances after around 6 seconds that are on average within 1.96% of the solutions obtained by HGS with a 5 second time limit.

6.3. Solution Quality Under a Less Restrictive Computational Budget

A third key result from our analyses is that FM-MCVRP trained on sub-optimal solutions produces competitive solutions to large problem instances compared to state-of-the-art heuristics, even when the computational budget is less constrained. Figure 4 illustrates this finding using an instance size of 400 customers as an example. It compares FM-MCVRP solutions under an NS decoding strategy with $1,000$ samples to the best out of $1,000$ HGS runs per problem instance with a 5 second time limit per run. FM-MCVRP solutions are shifted to the right compared to the HGS solutions with a mean relative difference of 0.81%. This deterioration relative to the HGS solutions is also statistically significant (see Appendix D). Recall

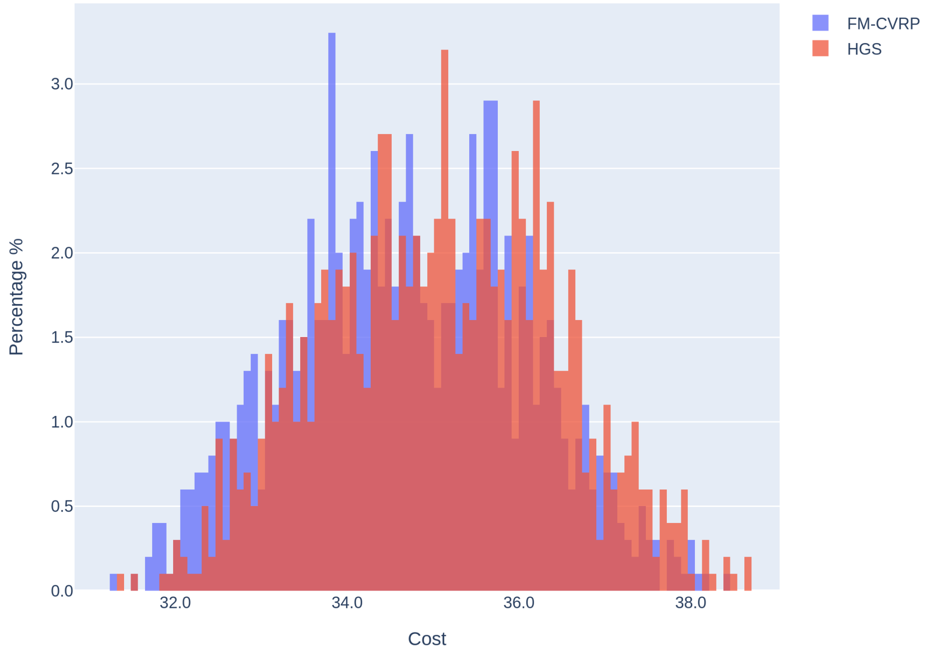


Figure 3 **Distance distributions of FM-MCVRP (NS, 1,000 samples) and HGS (single run) on 1,000 400-customer problem instances.**

		Obj.		Gap (%)		Time
N	Method (Decoder)	Avg.	80% IP Range	Avg.	80% IP Range	Avg.
20	HGS (no decoding, $s = 1$)	5.01	4.42 – 5.62	— *baseline* —		5.00s
	FM-MCVRP (GS)	5.42	4.73 – 6.12	8.29	0.35 – 17.25	0.26s
	FM-MCVRP (NS, $s = 1$)	5.45	4.74 – 6.19	9.05	0.32 – 19.98	0.26s
	FM-MCVRP (NS, $s = 100$)	5.09	4.48 – 5.70	1.62	0.00 – 5.01	2.48s
	FM-MCVRP (NS, $s = 1000$)	5.07	4.48 – 5.68	1.18	0.00 – 4.11	24.77s
50	HGS (no decoding, $s = 1$)	8.35	7.69 – 9.02	— *baseline* —		5.00s
	FM-MCVRP (GS)	8.78	8.08 – 9.55	5.21	0.89 – 9.91	0.55s
	FM-MCVRP (NS, $s = 1$)	8.79	8.07 – 9.58	5.27	0.92 – 10.50	0.55s
	FM-MCVRP (NS, $s = 100$)	8.46	7.83 – 9.13	1.30	-0.68 – 3.69	7.67s
	FM-MCVRP (NS, $s = 1000$)	8.42	7.79 – 9.09	0.87	-0.90 – 3.10	1.28min
100	HGS (no decoding, $s = 1$)	12.65	11.93 – 13.43	— *baseline* —		5.00s
	FM-MCVRP (GS)	13.13	12.31 – 14.02	3.75	0.38 – 7.36	1.03s
	FM-MCVRP (NS, $s = 1$)	13.13	12.29 – 14.01	3.75	0.49 – 7.11	1.03s
	FM-MCVRP (NS, $s = 100$)	12.68	11.96 – 13.46	0.21	-1.51 – 2.10	24.43s
	FM-MCVRP (NS, $s = 1000$)	12.61	11.90 – 13.38	-0.32	-1.99 – 1.25	4.07min
200	HGS (no decoding, $s = 1$)	19.77	18.76 – 20.80	— *baseline* —		5.00s
	FM-MCVRP (GS)	20.27	19.15 – 21.38	2.50	0.06 – 5.07	2.13s
	FM-MCVRP (NS, $s = 1$)	20.27	19.16 – 21.40	2.50	0.18 – 4.84	2.13s
	FM-MCVRP (NS, $s = 100$)	19.65	18.64 – 20.69	-0.61	-1.89 – 0.67	1.53min
	FM-MCVRP (NS, $s = 1000$)	19.54	18.55 – 20.55	-1.16	-2.38 – 0.01	15.30min
400	HGS (no decoding, $s = 1$)	35.08	33.24 – 36.81	— *baseline* —		5.00s
	FM-MCVRP (GS)	35.76	33.82 – 37.67	1.96	0.15 – 4.09	5.97s
	FM-MCVRP (NS, $s = 1$)	35.80	33.81 – 37.77	2.05	0.18 – 4.04	5.97s
	FM-MCVRP (NS, $s = 100$)	34.89	33.08 – 36.65	-0.52	-1.35 – 0.32	6.31min
	FM-MCVRP (NS, $s = 1000$)	34.71	32.87 – 36.45	-1.05	-1.82 – -0.29	63.14min

N: instance size; IP: inter-percentile; GS: greedy sampling; NS: nucleus sampling; s: number of samples
Metrics are reported over $m = 1,000$ instances per instance size
Times are reported based on average time per instance

Table 3 **Performance of FM-MCVRP trained on 38.1M single-sample solutions from HGS under a 5-second time limit compared to its training data for various decoding strategies.**

however, that our model has been trained on a set of sub-optimal solutions obtained from a single solution run of HGS with a time limit of 5 seconds. Therefore, it is not surprising that our model is outperformed by the best solution out of 1,000 HGS runs. However, given that FM-MCVRP was trained on lower-quality solutions, it is noteworthy that the solutions it finds remain highly competitive.

Finally, as an illustration, we show in Figure 5 (see Appendix E) a 400-node problem instance where FM-MCVRP outperformed HGS within the sample size.

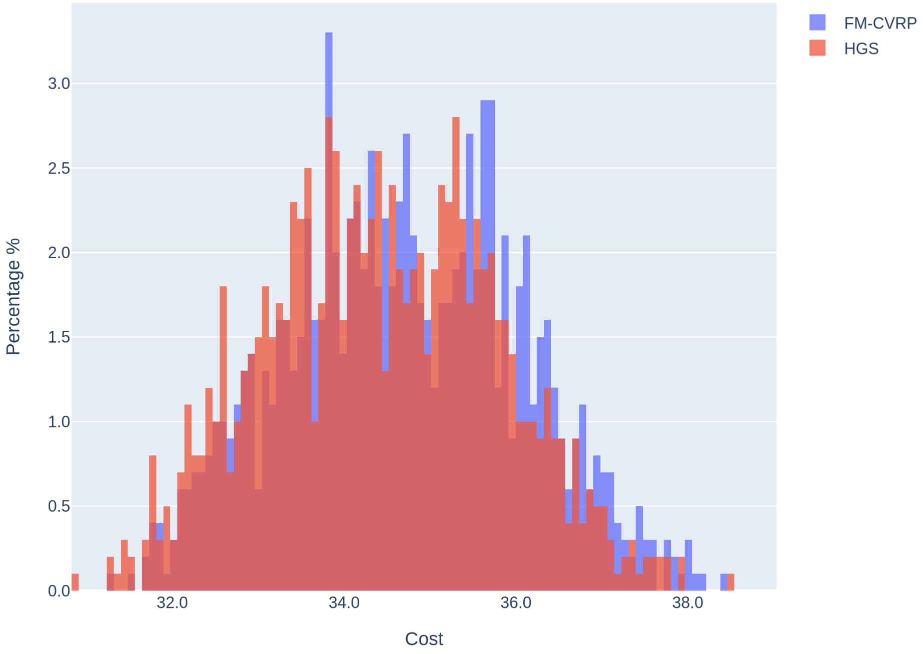


Figure 4 Distance distributions of FM-MCVRP (NS, 1,000 samples) and HGS (best of 1,000 runs) on 1,000 400-customer problem instances.

6.4. Generalizing to Larger Problems

Another key finding from our analyses is that our model generalizes well to problem instance sizes that were not part of the training data and go beyond the instance sizes that the model has seen during training. To illustrate this finding, we extend our performance comparison between FM-MCVRP and HGS to problem instances with 600 and 800 customers. Since these larger problem instances require more time to decode, we only base our performance statistics on the solutions obtained for 100 instances per instance size.

Table 4 shows that under a NS decoding strategy with 100 or $1,000$ samples, FM-MCVRP (trained on solutions for 20 to 400-node problems obtained from running HGS once per problem instance for 5 seconds, as discussed in Section 5.1) continues to outperform single-run solutions from HGS under a 5-second time limit. It yields inferior solutions compared to the single-run HGS benchmark for a decoding strategy with only a single sample, however. Nonetheless, the general capability of our model to generalize to larger problem instances is noteworthy as it illustrates the applicability of our method to real-world scenarios due to its robustness to variations in the problem instance characteristics. Moreover, this also indicates that it is possible to train FM-MCVRP on a larger dataset of smaller instances solved to near-optimality, which is computationally less costly to generate, and then generalize to larger problem instances that one might be interested in solving.

Finally, as an illustration, we show in Figure 6 (see Appendix E) an 800-node problem instance where FM-MCVRP outperformed HGS within the sample size.

N	Method (Decoder)	Wins	Obj.		Gap (%)		Time
			Avg.	80% IP Range	Avg.	80% IP Range	Avg.
600	HGS (no decoding, $s = 1$)	-	44.49	42.70 – 46.30	— *baseline* —		5.00s
	FM-MCVRP (GS, $s = 1$)	7	45.28	43.33 – 47.39	1.78	0.30 – 4.45	11.73s
	FM-MCVRP (NS, $s = 1$)	6	45.26	43.49 – 47.14	1.72	0.24 – 3.45	11.73s
	HGS (no decoding, $s = 100$)	-	43.91	41.95 – 45.68	— *baseline* —		12.75s
	FM-MCVRP (NS, $s = 100$)	5	44.23	42.25 – 46.06	0.72	0.30 – 1.16	14.52min
	HGS (no decoding, $s = 1000$)	-	43.75	41.71 – 45.61	— *baseline* —		2.13min
	FM-MCVRP (NS, $s = 1000$)	3	44.04	42.11 – 45.92	0.66	0.29 – 0.98	145.18min
800	HGS (no decoding, $s = 1$)	-	57.04	54.90 – 59.83	— *baseline* —		5.00s
	FM-MCVRP (GS, $s = 1$)	3	58.35	55.56 – 61.03	2.29	0.64 – 4.73	20.87s
	FM-MCVRP (NS, $s = 1$)	3	58.33	55.93 – 61.61	2.26	0.51 – 4.38	20.87s
	HGS (no decoding, $s = 100$)	-	56.46	54.38 – 59.40	— *baseline* —		12.83s
	FM-MCVRP (NS, $s = 100$)	3	56.91	54.56 – 59.83	0.80	0.31 – 1.32	27.35min
	HGS (no decoding, $s = 1000$)	-	56.29	54.12 – 59.28	— *baseline* —		2.14min
	FM-MCVRP (NS, $s = 1000$)	1	56.69	54.34 – 59.81	0.71	0.36 – 1.06	273.53min

N: instance size; IP: inter-percentile; GS: greedy sampling; NS: nucleus sampling; s: number of samples
Metrics are reported over $m = 1,000$ instances per instance size
Wins represent number problem instances in which the given method outperforms the benchmark method
Times are reported based on average time per instance

Table 4 Comparison of FM-MCVRP with HGS as the baseline for problem instances with 600 and 800 customers. The gap % is computed with respect to HGS within the same sample size.

6.5. Comparison with Other State-of-the-Art Methods

In this section, we extend our performance analysis and compare FM-MCVRP to two other state-of-the-art methods that solve the CVRP. First, we present a comparison with LKH-3, which is the best-performing heuristic for instance sizes of 400 customers or less across our experiments. Second, we present a comparison with AM, which is a widely discussed DRL approach to solving the CVRP.

6.5.1. Comparison with LKH-3. After demonstrating that FM-MCVRP generalizes to better solution qualities, larger problem instances, and less resource constrained heuristic solutions in the previous sections, we now want to show how FM-MCVRP solutions compare to the solutions found by the best-in-class heuristic, which is arguably LKH-3 for problem instances with up to 400 customers. The corresponding results from our numerical experiments are summarized in Table 5, which groups the obtained solutions from LKH-3 (baseline), HGS, and FM-MCVRP by instance size and the number of samples / algorithm runs.

As Table 5 shows, the single-sample solutions from FM-MCVRP (for both GS and NS) are inferior to the single-run solutions obtained from LKH-3 across all instance sizes, with an average relative difference in the solution value of up to 8.59% for 20-customer problems. However, there are two effects worth mentioning in our numerical results, which are affecting the relative performance of our method compared to the benchmark simultaneously. First, similar to what we have seen in Section 6.2 in comparison to HGS, the relative performance gap between LKH-3 (single-run) and FM-MCVRP (single-sample) reduces rapidly as problem instances get larger, which indicates that the solution quality of LKH-3 deteriorates more rapidly than that of our method as instances get larger. This is most clearly visible in Table 5 when comparing the single-run solutions from LKH-3 with their single-sample counterparts from FM-MCVRP. Here the average relative difference in solution value falls from 8.59% for 20-customer problems to 3.02% for 400-customer problems. Second, as we increase the number of samples / runs, FM-MCVRP solutions quickly become more competitive. However, the magnitude of this effect is dampened by the first effect. For 20-customer problem instances, the average relative difference in the solution value between LKH-3 and FM-MCVRP drops to 1.65% and 1.24% for 100 and $1,000$ runs / samples, respectively. This corresponds to a gap reduction by over 80% and over 85%, respectively. For 400-customer problem, the corresponding gap reductions from increasing the number of runs / samples from one to 100 and $1,000$ is only around 28% and 33%, respectively.

6.5.2. Comparison with AM.

After comparing FM-MCVRP with state-of-the-art heuristics, we also want to assess its performance relative to a recent, and widely discussed DRL approach to routing, AM. Table 6 shows the corresponding results of our numerical analyses grouped by instance size and the number of samples considered during decoding. Here, we make a number of important observations.

First, and most notably, AM diverges and thus fails to produce meaningful solutions for problem instances with 400 and more customers.

Second, FM-MCVRP solutions frequently and consistently outperform the solutions obtained from AM. For sample sizes of 100 and $1,000$, the solution values found by FM-MCVRP are up to 2.07% better on average than those found by AM. For large problem instances (here, 200 customers) the 80% inter-percentile range of the relative gap of FM-MCVRP solution over the AM solution spans from -3.23% to -0.88%, indicating that our method almost always outperforms AM. Only for relatively small problem instances (50 customers or less) and a sample size of one, FM-MCVRP on average yields worse solutions than AM. However, even in these cases, the 80% inter-percentile range of the relative gap spans well into the negative range, indicating that FM-MCVRP still regularly beats AM.

It is important to note that these results were obtained for FM-MCVRP being trained on sub-optimal solutions obtained from HGS under a strict time limit (see Section 5), while AM was retrained for this comparison under a much more generous training regime (see Appendix C). Further, we note that the AM is not a unified model. Unlike FM-MCVRP, which as we show above generalizes well to unseen instance sizes and vehicle capacities, we need to train AM for every specific combination of instance size and vehicle capacity we want to apply it to.

			Obj.		**Gap (%)**		**Time**
N	**Method (Decoder)**	**Wins**	Avg.	80% IP Range	Avg.	80% IP Range	Avg.
20	LKH-3 (no decoding, $s = 1$)	-	5.01	4.43 – 5.63	— *baseline* —		0.02s
	HGS (no decoding, $s = 1$)	359	5.01	4.42 – 5.62	-0.04	-0.26 – 0.05	5.00s
	FM-MCVRP (GS, $s = 1$)	38	5.42	4.73 – 6.12	8.18	0.23 – 17.21	0.26s
	FM-MCVRP (NS, $s = 1$)	33	5.44	4.74 – 6.16	8.59	0.26 – 19.30	0.26s
	LKH-3 (no decoding, $s = 100$)	-	5.01	4.43 – 5.61	— *baseline* —		0.20s
	HGS (no decoding, $s = 100$)	73	5.00	4.42 – 5.61	-0.08	0.00 – 0.00	11.16s
	FM-MCVRP (NS, $s = 100$)	36	5.09	4.49 – 5.69	1.65	0.00 – 5.06	2.48s
	LKH-3 (no decoding, $s = 1000$)	-	5.01	4.43 – 5.61	— *baseline* —		2.03s
	HGS (no decoding, $s = 1000$)	72	5.00	4.42 – 5.61	-0.08	0.00 – 0.00	1.85min
	FM-MCVRP (NS, $s = 1000$)	45	5.07	4.48 – 5.68	1.24	0.00 – 4.19	24.77s
50	LKH-3 (no decoding, $s = 1$)	-	8.33	7.70 – 9.02	— *baseline* —		0.08s
	HGS (no decoding, $s = 1$)	369	8.35	7.72 – 8.99	0.18	-1.45 – 1.89	5.00s
	FM-MCVRP (GS, $s = 1$)	36	8.78	8.08 – 9.55	5.42	1.03 – 10.25	0.55s
	FM-MCVRP (NS, $s = 1$)	32	8.80	8.08 – 9.56	5.61	1.14 – 10.83	0.55s
	LKH-3 (no decoding, $s = 100$)	-	8.26	7.65 – 8.89	— *baseline* —		0.43s
	HGS (no decoding, $s = 100$)	117	8.26	7.64 – 8.89	-0.03	0.00 – 0.00	11.52s
	FM-MCVRP (NS, $s = 100$)	9	8.46	7.82 – 9.13	2.40	0.37 – 4.76	7.67s
	LKH-3 (no decoding, $s = 1000$)	-	8.26	7.65 – 8.89	— *baseline* —		4.28s
	HGS (no decoding, $s = 1000$)	73	8.25	7.64 – 8.89	-0.04	0.00 – 0.00	1.92min
	FM-MCVRP (NS, $s = 1000$)	11	8.42	7.79 – 9.09	1.96	0.18 – 4.11	1.28min
100	LKH-3 (no decoding, $s = 1$)	-	12.53	11.78 – 13.32	— *baseline* —		0.47s
	HGS (no decoding, $s = 1$)	229	12.66	11.93 – 13.43	1.02	-1.00 – 3.05	5.00s
	FM-MCVRP (GS, $s = 1$)	34	13.13	12.31 – 14.02	4.77	1.30 – 8.52	1.03s
	FM-MCVRP (NS, $s = 1$)	41	13.14	12.28 – 14.01	4.84	1.34 – 8.60	1.03s
	LKH-3 (no decoding, $s = 100$)	-	12.32	11.63 – 13.08	— *baseline* —		0.99s
	HGS (no decoding, $s = 100$)	88	12.38	11.67 – 13.15	0.47	0.00 – 1.05	11.73s
	FM-MCVRP (NS, $s = 100$)	2	12.68	11.96 – 13.44	2.87	1.20 – 4.62	24.43s
	LKH-3 (no decoding, $s = 1000$)	-	12.30	11.62 – 13.07	— *baseline* —		9.87s
	HGS (no decoding, $s = 1000$)	78	12.33	11.64 – 13.08	0.23	0.00 – 0.62	1.96min
	FM-MCVRP (NS, $s = 1000$)	4	12.61	11.90 – 13.38	2.50	1.06 – 4.09	4.07min
200	LKH-3 (no decoding, $s = 1$)	-	19.48	18.44 – 20.52	— *baseline* —		0.91s
	HGS (no decoding, $s = 1$)	126	19.77	18.73 – 20.81	1.48	-0.29 – 3.20	5.00s
	FM-MCVRP (GS, $s = 1$)	35	20.27	19.15 – 21.38	4.05	1.40 – 6.87	2.13s
	FM-MCVRP (NS, $s = 1$)	25	20.28	19.19 – 21.43	4.12	1.34 – 6.86	2.13s
	LKH-3 (no decoding, $s = 100$)	-	19.13	18.13 – 20.14	— *baseline* —		2.75s
	HGS (no decoding, $s = 100$)	33	19.37	18.37 – 20.39	1.25	0.54 – 1.94	12.13s
	FM-MCVRP (NS, $s = 100$)	4	19.65	18.65 – 20.69	2.76	1.62 – 3.90	1.53min
	LKH-3 (no decoding, $s = 1000$)	-	19.06	18.07 – 20.07	— *baseline* —		27.46s
	HGS (no decoding, $s = 1000$)	32	19.26	18.29 – 20.26	1.06	0.52 – 1.60	2.02min
	FM-MCVRP (NS, $s = 1000$)	3	19.54	18.55 – 20.55	2.56	1.54 – 3.52	15.30min
400	LKH-3 (no decoding, $s = 1$)	-	34.70	32.89 – 36.52	— *baseline* —		3.88s
	HGS (no decoding, $s = 1$)	173	35.08	33.27 – 36.79	1.12	-0.64 – 2.52	5.00s
	FM-MCVRP (GS, $s = 1$)	62	35.76	33.82 – 37.67	3.08	0.73 – 5.46	5.97s
	FM-MCVRP (NS, $s = 1$)	50	35.74	33.78 – 37.61	3.02	0.70 – 5.23	5.97s
	LKH-3 (no decoding, $s = 100$)	-	34.16	32.37 – 35.94	— *baseline* —		10.93s
	HGS (no decoding, $s = 100$)	86	34.58	32.76 – 36.35	1.25	0.15 – 2.05	12.75s
	FM-MCVRP (NS, $s = 100$)	24	34.89	33.04 – 36.68	2.16	0.84 – 3.10	6.31min
	LKH-3 (no decoding, $s = 1000$)	-	34.02	32.22 – 35.80	— *baseline* —		1.82min
	HGS (no decoding, $s = 1000$)	76	34.43	32.61 – 36.19	1.21	0.24 – 1.85	2.12min
	FM-MCVRP (NS, $s = 1000$)	31	34.71	32.87 – 36.45	2.02	0.88 – 2.81	63.14min

N: instance size; IP: inter-percentile; GS: greedy sampling; NS: nucleus sampling; s: number of samples
Metrics are reported over $m = 1,000$ instances per instance size
Wins represent number problem instances in which the given method outperforms the benchmark method
Times are reported based on average time per instance

Table 5 Comparison of FM-MCVRP with LKH-3 as the baseline. The gap % is computed with respect to LKH-3 within the same sample size.

7. Managerial Implications

In the increasingly complex world of delivery logistics, optimizing the CVRP with state-of-the-art methods can yield substantial dividends. Specifically, consider how a 1% improvement in distance travelled for a global logistics company can have significant cost savings. In this paper, we leverage state-of-the-art advancements in LLMs and offer potentially transformative solutions for real-world logistics challenges. We outline four insights next.

			Obj.		Gap (%)		Time
N	**Method (Decoder)**	**Wins**	Avg.	80% IP Range	Avg.	80% IP Range	Avg.
20	AM (VS, $s = 1$)	-	5.17	4.54 – 5.84	— *baseline* —		0.03s
	FM-MCVRP (GS, $s = 1$)	297	5.42	4.73 – 6.12	5.06	-2.69 – 14.84	0.26s
	FM-MCVRP (NS, $s = 1$)	303	5.44	4.74 – 6.16	5.46	-2.61 – 15.80	0.26s
	AM (VS, $s = 100$)	-	5.09	4.47 – 5.73	— *baseline* —		0.03s
	FM-MCVRP (NS, $s = 100$)	539	5.09	4.49 – 5.69	-0.01	-2.85 – 3.31	2.48s
	AM (VS, $s = 1000$)	-	5.08	4.46 – 5.72	— *baseline* —		0.03s
	FM-MCVRP (NS, $s = 1000$)	552	5.07	4.48 – 5.68	-0.15	-2.52 – 2.48	24.77s
50	AM (VS, $s = 1$)	-	8.74	8.04 – 9.47	— *baseline* —		0.06s
	FM-MCVRP (GS, $s = 1$)	510	8.78	8.08 – 9.55	0.54	-4.27 – 5.98	0.55s
	FM-MCVRP (NS, $s = 1$)	473	8.80	8.08 – 9.56	0.72	-4.11 – 6.39	0.55s
	AM (VS, $s = 100$)	-	8.53	7.89 – 9.19	— *baseline* —		0.07s
	FM-MCVRP (NS, $s = 100$)	674	8.46	7.82 – 9.13	-0.78	-3.36 – 1.93	7.67s
	AM (VS, $s = 1000$)	-	8.49	7.85 – 9.16	— *baseline* —		0.08s
	FM-MCVRP (NS, $s = 1000$)	694	8.42	7.79 – 9.09	-0.77	-3.09 – 1.44	1.28min
100	AM (VS, $s = 1$)	-	13.15	12.34 – 13.99	— *baseline* —		0.12s
	FM-MCVRP (GS, $s = 1$)	540	13.13	12.31 – 14.02	-0.15	-4.07 – 3.66	1.03s
	FM-MCVRP (NS, $s = 1$)	533	13.14	12.28 – 14.01	-0.07	-3.90 – 3.74	1.03s
	AM (VS, $s = 100$)	-	12.82	12.08 – 13.64	— *baseline* —		0.13s
	FM-MCVRP (NS, $s = 100$)	751	12.68	11.96 – 13.44	-1.08	-3.08 – 1.08	24.43s
	AM (VS, $s = 1000$)	-	12.75	12.02 – 13.57	— *baseline* —		0.17s
	FM-MCVRP (NS, $s = 1000$)	758	12.61	11.90 – 13.38	-1.06	-2.91 – 0.85	4.07min
200	AM (VS, $s = 1$)	-	20.59	19.55 – 21.62	— *baseline* —		0.29s
	FM-MCVRP (GS, $s = 1$)	782	20.27	19.15 – 21.38	-1.54	-4.17 – 1.21	2.13s
	FM-MCVRP (NS, $s = 1$)	781	20.28	19.19 – 21.43	-1.46	-4.10 – 1.25	2.13s
	AM (VS, $s = 100$)	-	20.07	19.08 – 21.08	— *baseline* —		0.31s
	FM-MCVRP (NS, $s = 100$)	979	19.65	18.65 – 20.69	-2.07	-3.40 – -0.75	1.53min
	AM (VS, $s = 1000$)	-	19.95	18.97 – 20.95	— *baseline* —		0.53s
	FM-MCVRP (NS, $s = 1000$)	981	19.54	18.55 – 20.55	-2.02	-3.23 – -0.88	15.30min
400	AM (VS, $s = 1$)	-	-	-	-	-	
	FM-MCVRP (GS, $s = 1$)	1000	35.76	33.82 – 37.67	-	-	5.97s
	FM-MCVRP (NS, $s = 1$)	1000	35.74	33.78 – 37.61	-	-	5.97s
	AM (VS, $s = 100$)	-	-	-	-	-	
	FM-MCVRP (NS, $s = 100$)	1000	34.89	33.04 – 36.68	-	-	6.31min
	AM (VS, $s = 1000$)	-	-	-	-	-	
	FM-MCVRP (NS, $s = 1000$)	1000	34.71	32.87 – 36.45	-	-	63.14min

N: instance size; IP: inter-percentile; VS: vanilla sampling; GS: greedy sampling; NS: nucleus sampling; s: number of samples
Metrics are reported over $m = 1,000$ instances per instance size
Times are reported based on average time per instance

Table 6 Comparison of FM-MCVRP with AM as the baseline. The gap % is computed with respect to AM within the same sample size.

Supervised Learning on Historical Data. Most, if not all global logistics companies have dedicated OR teams and use sophisticated algorithms that have solved the CVRP over the past few decades. These companies likely have a vast amount of historical problem-solution pairs. In addition, many of these companies may also have records of how routes were executed by the driver in reality, likely taking into account other factors beyond total travel distance, time, or cost. These factors could include safety, convenience, and other factors. We show that our supervised learning method is effective in learning from a state-of-the-art heuristic solutions to the CVRP, and we hypothesize that it can also effectively learn from real-world generated solutions that incorporate more complex objectives and constraints followed and adhered to by actual drivers. With large amounts of historical data readily available, these companies can implement our method to improve their route operations and potentially learn from the tacit knowledge of their most experienced and productive drivers.

Unified Model for Varying Numbers of Customers and Truck Capacities. Our model also provides significantly more convenience from a Machine Learning Operations (MLOps) perspective. As mentioned above, prior work based on DRL require a specific model to be trained for a given number of customers and truck capacity. However, this is not a given in real-world delivery problems. Using these methods would require MLOps teams to deploy and maintain multiple models tailored for specific customer counts and truck capacities. Our approach introduces a unified model that performs well over a wide spectrum of problem sizes and vehicle capacities, simplifying the deployment process and reducing model maintenance effort.

Generalizing to Superior Solution Qualities. Beyond mere adaptability to different customer sizes and truck capacities, our model is able to decode solutions of higher quality than what it was trained on. This property could allow our proposed model to continuously improve the route quality of a company, as it learns from historical or algorithmically generated solutions, and subsequently proposes higher quality solutions that could themselves be used to further (re-)train our model.

Scalability Beyond Normal Operations. Another significant finding of our research is our model's capability to handle larger problem sizes than those it was initially trained on. In particular, we trained the model on 20 to 400-node problem instances, and found that the model could still produce solutions of high quality for 600 and 800-node problem instances. In practical terms, a company can train a model on data for regular delivery scenarios and can confidently apply this model to peak demand periods, even if the model has not previously encountered such high volumes during training.

All in all, our findings chart a path for delivery companies to embrace ML-based routing methods. Especially as the e-commerce landscape rapidly evolves, harnessing ML methods for route planning could be critical for companies to make their delivery operations more flexible, customer-centric, scalable, and adaptable – essential prerequisites for driving down costs and enhancing overall customer satisfaction.

8. Conclusion

In this paper, we propose the FM-MCVRP, a novel Deep Learning (DL) model that solves the so-called Montreal Capacitated Vehicle Routing Problem (MCVRP), a variant of the CVRP that closely mimics real-world delivery problems. To the best of our knowledge, our work is the first to leverage Transformers in an LLM framework to solve the MCVRP, contrary to recent works that use the Pointer Network (PN) framework.

Our proposed unified model and the findings from our numerical study, which demonstrate competitiveness with state-of-the-art heuristics, are of high significance to the academic community as they constitute a first step towards successfully applying LLM frameworks to CO problems. They are also of high significance to industrial practice as many real-world delivery problems operate within fixed and given operational environments (i.e., known road networks, customer addresses). The type of model presented in this paper can help to exploit patterns in these environments, learn from existing operational data, and gradually improve over previously found solutions.

The main limitations of our proposed work are threefold. First, we intentionally trained FM-MCVRP with sub-optimal solutions as we wanted to show that FM-MCVRP is able to generate solutions of a higher quality than the sub-optimal solutions it was trained on. Future research should explore using higher quality solutions by potentially extending the 5s time limit, and exploring the limit in which outperformance is no longer possible.

Second, FM-MCVRP was trained with the T5 schedule, which consists of a fixed learning rate for a certain number of steps followed by an exponential decay of the learning rate. However, Iyer et al. (2023) recently proposed the *Knee* training schedule and showed that this schedule increases the performance of the model. Future work should leverage this work and similar insights into model training to potentially obtain better models.

Third, as FM-MCVRP is a conditional probability model, autoregressive decoding is required to obtain solutions and this operation cannot be parallelized. Future work could involve a non-autoregressive model that can be parallelized, thus speeding up the decoding process significantly.

There are a number of additional areas for future research that appear particularly promising. First, as seen from the 2021 Amazon Last Mile Routing Challenge (Merchán et al. 2022), favorable solutions to real-world routing problems are often not distance optimal. Instead, drivers optimize for more complex objective functions, aiming at balancing safety, convenience, and other factors beyond cost efficiency and our proposed supervised learning method is can potentially capture these preferences.

Second, while Raffel et al. (2020) conclude that the encoder-decoder architecture works best for the type of problem we are solving in this paper, it would be worth exploring GPT-style (decoder-only) architectures. While this would result in significantly higher computational cost, a GPT-style architecture would have access to the features of all layers in the Transformer, which potentially enables learning embeddings with better representations.

Lastly, a particularly intriguing area of future research is the development of a system in which a DL model starts by learning from a state-of-the-art heuristic and then produces increasingly better solutions that can be bootstrapped into training the model itself, leading to even better solutions, and creating a positive feedback loop.

Appendix A: Attention Mechanism

At a high level, the Attention mechanism proposed by Vaswani et al. (2017) takes a set of feature vectors in the form of a matrix and transforms it into another matrix of higher-level features of the same size. Specifically, the equations are given by

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{4}$$

with $\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$, and $Q = XW_Q$, $K = XW_K$, and $V = XW_V$, where X is the input matrix containing the feature vectors; W_Q, W_K, and W_V are learned weight matrices responsible for transforming the input X into Query (Q), Key (K), and Value (V) matrices, respectively; QK^T is the dot product between the query and key matrices, resulting in a matrix whose element at the i-th row and j-th column represents the compatibility of the i-th query with the j-th key; $\sqrt{d_k}$ is a scaling factor, where d_k is the dimensionality of the queries and keys. This scaling ensures that the magnitudes of the dot products do not grow too large, which could potentially lead to gradients that are difficult to manage during the optimization process. In words, this mechanism allows the model to decide which aspects of the input to focus on (via QK^T) and which parts should be summed in the final output (via V).

Appendix B: Choosing Hyperparameters

Modern DL approaches, including Transformer models, are predominantly trained with SGD (Robbins and Monro 1951). In SGD, there are three high-level decisions that have to be made: the type of optimizer to use, setting the learning rate, and setting the batch size.

Optimizers. In SGD, the most basic form of the parameter update rule is $\theta_{t+1} = \theta_t - \alpha g_t$, where θ_t and g_t represent the parameters and the gradient respectively at time step t, and α the learning rate in this specific step. There are a myriad of optimizers in the DL literature and most if not all of them adapt this basic parameter update rule. We focus our discussion on three optimizers that are commonly used in DL research: First, the *Adam* optimizer (Kingma and Ba 2014) is a popular optimizer that was commonly used in early DL research due to its fast convergence when tested empirically. Second, *AdamW* (Loshchilov and Hutter 2019) improves on Adam with a parameter update rule that decouples the weight decay and produces models that generalize better when compared with Adam. Lastly, *AdaFactor* (Shazeer and Stern 2018) is largely equivalent to Adam but is more memory efficient and achieves results comparable to models trained with Adam.

All three of these optimizers generally involve the first and second moments of the gradients. The corresponding parameters, β_1 and β_2, can be tuned along with α. In practice, the default implementation of AdamW in PyTorch sets the values of α, β_1 and β_2 to 0.001, 0.900 and 0.999 respectively.

Learning rates. Depending on the type of DL model, the initial learning rate is another hyperparameter that can be tuned. We omit the discussion of setting the initial learning rate as it is generally a number set to the default of 10^{-3} in state-of-the-art ML libraries like PyTorch (Paszke et al. 2019) and TensorFlow (Martín˜Abadi et al. 2015), or determined based on a grid search. We instead focus our discussion on two broad topics: the warm-up schedule and the annealing schedule. Goyal et al. (2017) first proposed a gradual warm-up, where the learning rate starts from zero and gradually increases to the desired learning rate. Goyal et al. (2017) argue that this enables healthy convergence at the start of training. Their warm-up schedule is commonly used by most recently proposed DL models. For instance, LLaMA-2, a popular LLM, uses the gradual warm-up scheme for the first 2,000 steps of training (Touvron et al. 2023).

In contrast, the T5 model, another relatively popular LLM, does not follow a gradual warm-up, but instead starts with a high constant learning rate (Raffel et al. 2020). Regardless of the warm-up schedule, after reaching the peak learning rate, most models follow an annealing schedule that gradually reduces the learning rate to a constant value that is smaller than the peak learning rate. There are many annealing schedules, and we focus our discussion on the annealing schedules used by LLaMA-2 and T5. LLaMA-2 uses a cosine annealing schedule, first proposed by Loshchilov and Hutter (2016), and T5 uses a learning rate of $\frac{1}{\sqrt{\max(n,k)}}$, where n is the current training iteration and k is the number of warm-up steps, which the authors set to 10^4. In their most recent work, (Iyer et al. 2023) discuss the *wide minima density hypothesis*, which suggests that a high learning rate in the early stages of training increases the probability of the model to explore and arrive at areas with a high density of wide minima. The authors propose a ‘knee-shaped’ *explore-exploit* learning rate schedule. Specifically, they show that training at a high learning rate (*explore* phase) for an extended period of time before linearly decaying to zero (*exploit* phase) yields higher performing models as it increases the probability of the model converging to a wide minima (Iyer et al. 2023), which has been shown to lead to better generalization results (Keskar et al. 2016).

All in all, we see that the literature on determining the warm-up scheme, peak learning rate, and annealing schedule varies widely and is largely based on empirical findings. For the purposes of our research, we chose to adhere closely to established and empirically validated models to limit the extent of experimental variables we modify.

Batch sizes. As DL models are trained with SGD, a decision on the batch size has to be made. As the batch size gets larger, gradients are more accurate and there is less noise in the gradients. In the extreme case of batch gradient descent, where gradients are computed over the entire dataset, convergence to a local minimum is guaranteed but the performance of the model might be poor. Therefore, SGD is commonly used to introduce noise in the optimization process as it allows the escaping of local minima and potentially enables convergence to better minima. The literature on determining suitable batch sizes has evolved over the past few years.

Until recently, it was widely accepted that large batch sizes result in a large generalization gap (or high test error). Keskar et al. (2016) first suggested that large batch sizes (512 and above) tend to converge to sharp minimizers (cf., Figure 1, Keskar et al. 2016), which results in a model with poor generalization and small batch sizes converge to flat minimizers (cf., Figure 1, Keskar et al. 2016), which results in a model with better generalization. However, Hoffer et al. (2017) show that it is not the batch size that affects model generalization, but rather the reduction in the number of SGD updates that results from increasing the batch size while keeping the number of training epochs constant.

Goyal et al. (2017) show that large batch sizes can still result in a small generalization gap. They successfully train ImageNet (Deng et al. 2009), a canonical CV dataset for various CV tasks, within one hour with a simple heuristic that scales the initial or peak learning rate by k, if the batch size increases by k. This finding corroborates with an earlier technical report by Krizhevsky (2014), who suggests that the learning rate should be scaled by $\sqrt{k}$ as it scales the learning rate proportionately to the reduction in the standard deviation of the gradient estimator in a batch. In practice, both k and $\sqrt{k}$ are commonly used. We opted for $\sqrt{k}$ as a more conservative estimate as it is a smaller value than k.

Appendix C: Retraining AM

We follow the default training parameters for AM by training the models for 100 epochs, with each epoch having 1.28M training samples and utilizing the maximum available computational resources as prescribed by their code. Specifically, their code is designed to operate on a single node and utilizes the maximum number of GPUs available on that node. On our infrastructure, this translates to 2 Tesla V100-PCIE-32GB GPUs. Additionally, we increase the batch size to its maximum limit without encountering an out of memory error on the GPUs, maintaining a consistent effective batch size of 512 for problem instances of sizes 20, 50, 100, and 200. However, as the size of the problem instances grow, GPU memory constraints necessitate a reduction in batch size. Consequently, for problem instance sizes of 400, 600, and 800, the batch sizes we employ are 128, 64, and 32, respectively. Finally, the models for problem instances of sizes 20, 50, 100 and 200 successfully converged after 100 epochs, while the models for problem instances of sizes 400, 600 and 800 diverged within the first 10 epochs and thus we omit these results when comparing FM-MCVRP against AM.

Appendix D: Test Statistics

We tested the distance distributions obtained in Section 6.2 and Section 6.3 with a one-sided paired samples t-test (Ross and Willson 2017) as the test instances are the same. In Section 6.2, the null hypothesis H_0 is testing if the solution values of FM-MCVRP (NS, 1,000 samples) is greater than or equal to the solution values of HGS (single run) on 1,000 instances of a 400-customer problem in a paired samples t-test. Given this null hypothesis, the alternative hypothesis H_1, is the direct opposite, where the solution values of FM-MCVRP (NS, 1,000 samples) is less than the solution values of HGS (single run). The one-sided paired samples t-test had a p-value of 0, allowing us to reject the null hypothesis. This implies that FM-MCVRP (NS, 1,000 samples) has a higher performance in terms of solution value when compared with HGS (single run). Table 7 shows the details of the statistics.

	X	Y
H_0	$X \geq Y$	
H_1	$X < Y$	
mean	34.71	35.08
std	1.37	1.37
t-stat	-53.43	
p-value	0.00	
degrees of freedom	999	
95% CI	-0.39 – -0.36	

X: FM-MCVRP (NS, 1,000 samples) ; Y: HGS (single run);
Metrics are reported over 1,000 instances

Table 7 Test statistics of comparing FM-MCVRP (NS, 1,000 samples) with HGS (single run).

In Section 6.3, the null hypothesis H_0 is testing if the solution values of HGS (best of 1,000 runs) is greater than or equal to the solution values of FM-MCVRP (NS, 1,000 samples) on 1,000 instances of a 400-customer problem in a paired samples t-test. Given this null hypothesis, the alternative hypothesis H_1, is the direct opposite, where the solution values of HGS (best of 1,000 runs) is less than the solution values of FM-MCVRP (NS, 1,000 samples). The one-sided paired samples t-test had a p-value of 0, allowing us to reject the null hypothesis. This implies that HGS (best of 1,000 runs) has a higher performance in terms of solution value when compared with FM-MCVRP (NS, 1,000 samples). Table 8 shows the details of the statistics.

	X	Y
H_0	$X \geq Y$	
H_1	$X < Y$	
mean	34.43	34.71
std	1.35	1.37
t-stat	-62.84	
p-value	0.00	
degrees of freedom	999	
95% CI	-0.28 – -0.27	

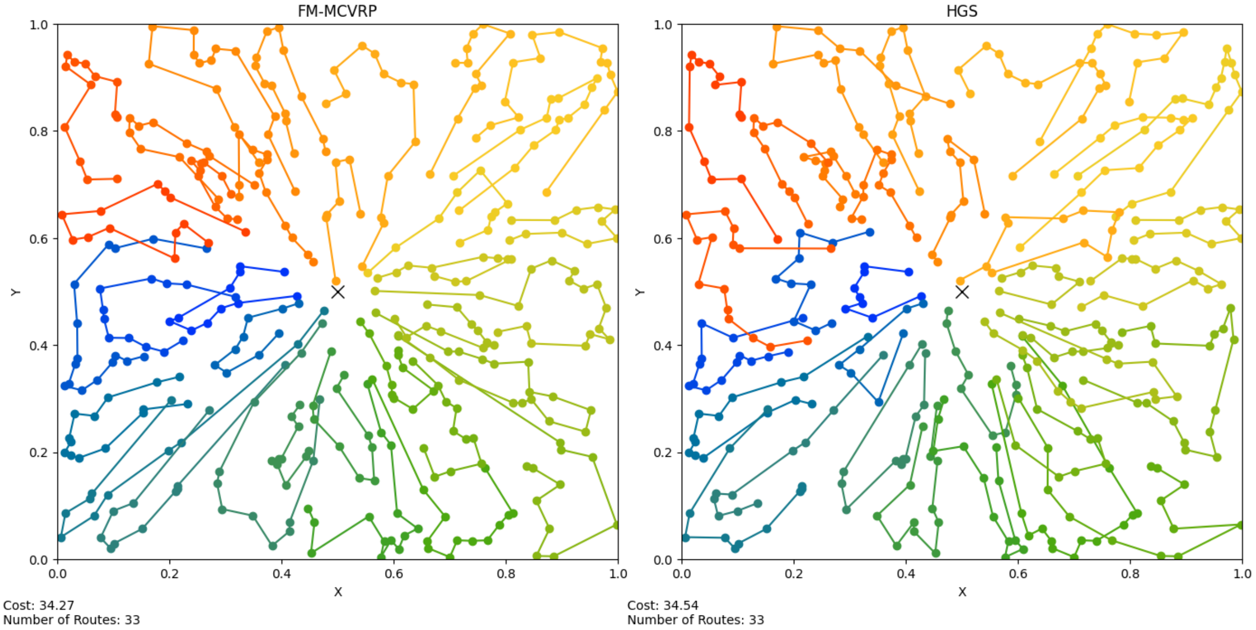
X: HGS (best of 1,000 runs) ; Y: FM-MCVRP (NS, 1,000 samples);

Metrics are reported over 1,000 instances

Table 8 Test statistics of comparing HGS (best of 1,000 runs) with FM-MCVRP (NS, 1,000 samples).

Appendix E: Example Solutions

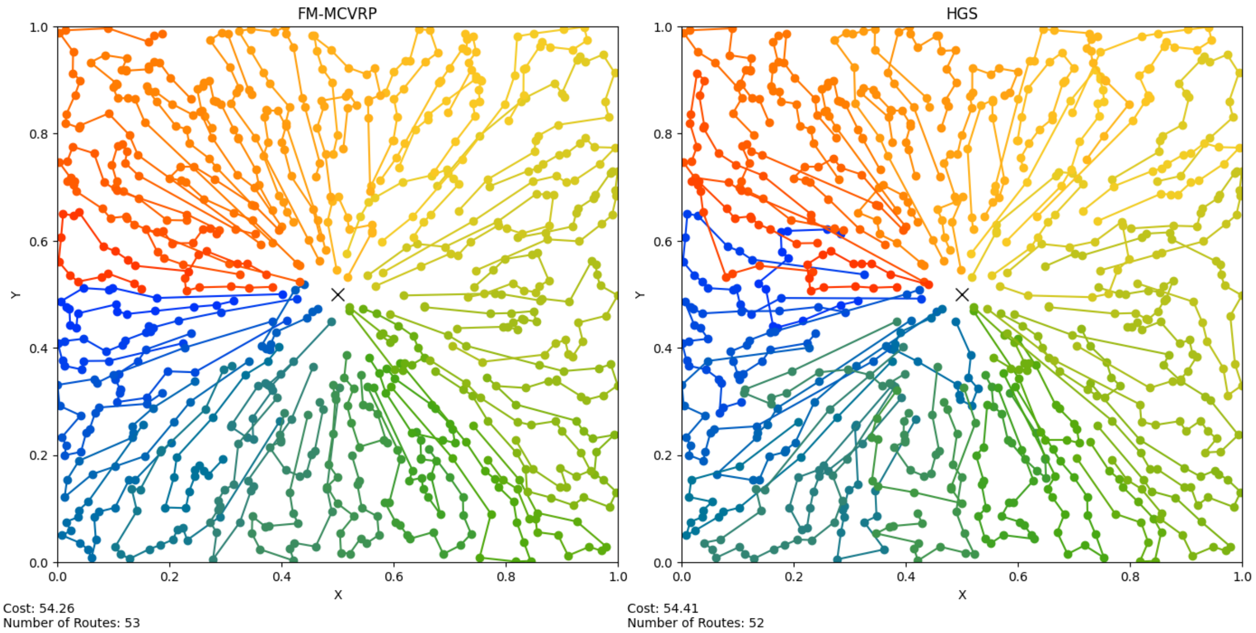
Figure 5 shows an example solution for a 400-node MCVRP problem instance with the best gap when comparing FM-MCVRP (NS, 1,000 samples) with HGS (best of 1,000 runs). Observe how FM-MCVRP finds solutions where the routes are tightly clustered within an angle compared to HGS. Figure 6 shows an example solution for an 800-node MCVRP problem instance with the best gap when comparing FM-MCVRP (NS, 1,000 samples) with HGS (best of 1,000 runs). Again, observe how FM-MCVRP finds solutions where the routes are tightly clustered within an angle compared to HGS. Most notably, in this example, FM-MCVRP found a solution with an additional route but has a better solution value. These visualizations give insight to the solution distribution that FM-MCVRP has learnt.



(a) FM-MCVRP Solution

(b) HGS Solution

Figure 5 400-node problem instance with best gap when comparing FM-MCVRP (NS, 1,000 samples) with HGS (best of 1,000 runs).



(a) FM-MCVRP Solution

(b) HGS Solution

Figure 6 **800-node problem instance with best gap when comparing FM-MCVRP with HGS on 1,000 samples. For this instance, observe how FM-MCVRP found a shorter solution with more routes.**

Appendix F: A Note on Route-Level Permutation Invariance

The Transformer is a Graph Neural Network (GNN) and GNNs have the properties of permutation invariance and permutation equivariance. A function f is permutation invariant if $f(PX) = f(X)$, where P is a permutation matrix and X is the input matrix. In each layer of our architecture, the features are matrices (X) and thus the order of the features can be permuted and the output will be the same. Specifically, consider the routes produced by the solution section of our architecture (note that these are route IDs and not node IDs) $[0, 1, 2, 3, 4]$ and $[0, 1, 3, 2, 4]$. The embedding computed for the first node of route 4 is exactly the same and the order of the routes before that do not matter as the Attention mechanism transforms the input with a weighted sum and the sum operator is permutation invariant. We hypothesized that this better represented the true space of solutions, and thus augmented the solutions during training by permuting our routes in this manner, giving the model multiple valid solutions for a given problem. We experimented with multiple models trained in this manner and found that these models had extremely poor performance. In particular, the routes were decoded in no particular order and was prone to miss out nodes in certain areas, which resulted in a giant loop being executed as the final route. We found that ordering the solutions in a manner where routes that have similar angles with respect to the depot are grouped together alleviated the problem of having a giant loop. Specifically, with reference to Figure 5 or 6, an example of a solution that follows this order would have the blue routes decoded first and then sweeps counter clockwise to decode the green, yellow and eventually red routes. While this specific ordering enabled us to achieve outperformance on HGS, it remains an open question as to whether or not a permutation invariant operator can result in a higher performing model.

Acknowledgments

The authors acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC resources that have contributed to the research results reported within this paper.

References

Archetti C, Speranza M (2014) A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization* 2(4):223–246, ISSN 21924406, URL `http://dx.doi.org/10.1007/s13675-014-0030-7`.

Arnold F, Gendreau M, Sörensen K (2019) Efficiently solving very large-scale routing problems. *Computers & Operations Research* 107:32–42, ISSN 03050548, URL `http://dx.doi.org/10.1016/j.cor.2019.03.006`.

Arnold F, Sörensen K (2019) Knowledge-guided local search for the vehicle routing problem. *Computers & Operations Research* 105:32–46, ISSN 03050548, URL `http://dx.doi.org/10.1016/j.cor.2019.01.002`.

Bahdanau D, Cho K, Bengio Y (2014) Neural Machine Translation by Jointly Learning to Align and Translate .

Bengio Y, Lodi A, Prouvost A (2021) Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research* 290(2):405–421, ISSN 03772217, URL `http://dx.doi.org/10.1016/j.ejor.2020.07.063`.

Bengio Y, Louradour J, Collobert R, Weston J (2009) Curriculum learning. *Proceedings of the 26th Annual International Conference on Machine Learning*, 41–48 (New York, NY, USA: ACM), ISBN 9781605585161, URL `http://dx.doi.org/10.1145/1553374.1553380`.

Bertsimas D, Tsitsiklis J (1993) Simulated annealing. *Statistical science* 8(1):10–15.

Choo J, Kwon YD, Kim J, Jae J, Hottung A, Tierney K, Gwon Y (2022) Simulation-guided Beam Search for Neural Combinatorial Optimization. Oh AH, Agarwal A, Belgrave D, Cho K, eds., *Advances in Neural Information Processing Systems*, URL `https://openreview.net/forum?id=tYAS1Rpys5`.

Christiaens J, Berghe GV (2020) Slack induction by string removals for vehicle routing problems. *Transportation Science* 54(2):417–433, ISSN 15265447, URL `http://dx.doi.org/10.1287/trsc.2019.0914`.

Cook WJ, Applegate DL, Bixby RE, Chvátal V (2011) *The Traveling Salesman Problem* (Princeton University Press), ISBN 9781400841103, URL `http://dx.doi.org/10.1515/9781400841103`.

Costa L, Contardo C, Desaulniers G (2019) Exact Branch-Price-and-Cut Algorithms for Vehicle Routing. *Transportation Science* 53(4):946–985, ISSN 0041-1655, URL `http://dx.doi.org/10.1287/trsc.2018.0878`.

Croes GA (1958) A method for solving traveling-salesman problems. *Operations research* 6(6):791–812.

Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, 886–893.

Deng J, Dong W, Socher R, Li LJ, Kai Li, Li Fei-Fei (2009) ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255 (IEEE), ISBN 978-1-4244-3992-8, URL `http://dx.doi.org/10.1109/CVPR.2009.5206848`.

Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Burstein J, Doran C, Solorio T, eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186 (Minneapolis, Minnesota: Association for Computational Linguistics), URL `http://dx.doi.org/10.18653/v1/N19-1423`.

Fan A, Lewis M, Dauphin Y (2018) Hierarchical Neural Story Generation. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 889–898 (Melbourne, Australia: Association for Computational Linguistics), URL `http://dx.doi.org/10.18653/v1/P18-1082`.

Goyal P, Dollár P, Girshick R, Noordhuis P, Wesolowski L, Kyrola A, Tulloch A, Jia Y, He K (2017) Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour .

Helsgaun K (2000) An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research* 126(1):106–130, ISSN 03772217, URL `http://dx.doi.org/10.1016/S0377-2217(99)00284-2`.

Helsgaun K (2017) An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University* 24–50.

Hoffer E, Hubara I, Soudry D (2017) Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems* 30.

Holland JH (1992) *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence* (MIT press).

Holtzman A, Buys J, Du L, Forbes M, Choi Y (2019) The Curious Case of Neural Text Degeneration .

Holtzman A, Buys J, Du L, Forbes M, Choi Y (2020) The Curious Case of Neural Text Degeneration. *International Conference on Learning Representations*, URL `https://openreview.net/forum?id=rygGQyrFvH`.

Holtzman A, Buys J, Forbes M, Bosselut A, Golub D, Choi Y (2018) Learning to Write with Cooperative Discriminators. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1638–1649 (Melbourne, Australia: Association for Computational Linguistics), URL `http://dx.doi.org/10.18653/v1/P18-1152`.

Hopfield JJ, Tank DW (1985) “Neural” computation of decisions in optimization problems. *Biological Cybernetics* 52(3):141–152, ISSN 0340-1200, URL `http://dx.doi.org/10.1007/BF00339943`.

Iyer N, Thejas V, Kwatra N, Ramjee R, Sivathanu M (2023) Wide-minima density hypothesis and the explore-exploit learning rate schedule. *Journal of Machine Learning Research* 24(65):1–37.

Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, Tunyasuvunakool K, Bates R, Žídek A, Potapenko A, Bridgland A, Meyer C, Kohl SAA, Ballard AJ, Cowie A, Romera-Paredes B, Nikolov S, Jain R, Adler J, Back

T, Petersen S, Reiman D, Clancy E, Zielinski M, Steinegger M, Pacholska M, Berghammer T, Bodenstein S, Silver D, Vinyals O, Senior AW, Kavukcuoglu K, Kohli P, Hassabis D (2021) Highly accurate protein structure prediction with AlphaFold. *Nature* 596(7873):583–589, ISSN 0028-0836, URL `http://dx.doi.org/10.1038/s41586-021-03819-2`.

Keskar NS, Mudigere D, Nocedal J, Smelyanskiy M, Tang PTP (2016) On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima .

Kingma DP, Ba J (2014) Adam: A Method for Stochastic Optimization .

Kool W, van Hoof H, Welling M (2019) Attention, Learn to Solve Routing Problems! *International Conference on Learning Representations*, URL `https://openreview.net/forum?id=ByxBFsRqYm`.

Krizhevsky A (2014) One weird trick for parallelizing convolutional neural networks .

Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, volume 25 (Curran Associates, Inc.), URL `https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html`.

Kwon YD, Choo J, Kim B, Yoon I, Gwon Y, Min S (2020) Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems* 33:21188–21198.

Leblond R, Alayrac JB, Sifre L, Pislar M, Jean-Baptiste L, Antonoglou I, Simonyan K, Vinyals O (2021) Machine Translation Decoding beyond Beam Search. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 8410–8434 (Online and Punta Cana, Dominican Republic: Association for Computational Linguistics), URL `http://dx.doi.org/10.18653/v1/2021.emnlp-main.662`.

Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. *Operations research* 21(2):498–516.

Loshchilov I, Hutter F (2016) SGDR: Stochastic Gradient Descent with Warm Restarts .

Loshchilov I, Hutter F (2019) Decoupled Weight Decay Regularization. *International Conference on Learning Representations*, URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60:91–110.

Lowerre BT (1976) The HARPY speech recognition system .

Lu H, Zhang X, Yang S (2019) A learning-based iterative method for solving vehicle routing problems. *International conference on learning representations*.

Lu X, Welleck S, West P, Jiang L, Kasai J, Khashabi D, Le Bras R, Qin L, Yu Y, Zellers R, Smith NA, Choi Y (2022) NeuroLogic A*esque Decoding: Constrained Text Generation with Lookahead Heuristics. *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 780–799 (Seattle, United States: Association for Computational Linguistics), URL `http://dx.doi.org/10.18653/v1/2022.naacl-main.57`.

Martín~Abadi, Ashish~Agarwal, Paul~Barham, Eugene~Brevdo, Zhifeng~Chen, Craig~Citro, Greg~S~Corrado, Andy~Davis, Jeffrey~Dean, Matthieu~Devin, Sanjay~Ghemawat, Ian~Goodfellow, Andrew~Harp, Geoffrey~Irving, Michael~Isard, Jia Y, Rafal~Jozefowicz, Lukasz~Kaiser, Manjunath~Kudlur, Josh~Levenberg, Dandelion~Mané, Rajat~Monga, Sherry~Moore, Derek~Murray, Chris~Olah, Mike~Schuster, Jonathon~Shlens, Benoit~Steiner, Ilya~Sutskever, Kunal~Talwar, Paul~Tucker, Vincent~Vanhoucke, Vijay~Vasudevan, Fernanda~Viégas, Oriol~Vinyals, Pete~Warden, Martin~Wattenberg, Martin~Wicke, Yuan~Yu, Xiaoqiang~Zheng (2015) TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. URL `https://www.tensorflow.org/`.

Merchán D, Arora J, Pachon J, Konduri K, Winkenbach M, Parks S, Noszek J (2022) 2021 Amazon Last Mile Routing Research Challenge: Data Set. *Transportation Science* ISSN 0041-1655, URL `http://dx.doi.org/10.1287/trsc.2022.1173`.

Nakkiran P, Kaplun G, Bansal Y, Yang T, Barak B, Sutskever I (2021) Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment* 2021(12):124003.

Nazari M, Oroojlooy A, Snyder L, Takác M (2018) Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems* 31.

Oliver IM, Smith D, Holland JRC (1987) A study of permutation crossover operators on the traveling salesman problem. *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, 224–230.

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, others (2019) Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32.

Pecin D, Pessoa A, Poggi M, Uchoa E (2017) Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation* 9(1):61–100, ISSN 18672957, URL `http://dx.doi.org/10.1007/s12532-016-0108-8`.

Pessoa A, Sadykov R, Uchoa E, Vanderbeck F (2020) A generic exact solver for vehicle routing and related problems. *Mathematical Programming* 183(1-2):483–523, ISSN 0025-5610, URL `http://dx.doi.org/10.1007/s10107-020-01523-z`.

Queiroga E, Sadykov R, Uchoa E (2021) A POPMUSIC matheuristic for the capacitated vehicle routing problem. *Computers & Operations Research* 136:105475, ISSN 03050548, URL `http://dx.doi.org/10.1016/j.cor.2021.105475`.

Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I, others (2019) Language models are unsupervised multitask learners. *OpenAI blog* 1(8):9.

Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu PJ (2020) Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21(1):5485–5551.

Reuther A, Kepner J, Byun C, Samsi S, Arcand W, Bestor D, Bergeron B, Gadepally V, Houle M, Hubbell M, Jones M, Klein A, Milechin L, Mullen J, Prout A, Rosa A, Yee C, Michaleas P (2018) Interactive Supercomputing on 40,000 Cores for Machine Learning and Data Analysis URL `http://dx.doi.org/10.1109/HPEC.2018.8547629`.

Robbins H, Monro S (1951) A Stochastic Approximation Method. *The Annals of Mathematical Statistics* 22(3):400–407, ISSN 0003-4851, URL `http://dx.doi.org/10.1214/aoms/1177729586`.

Ross A, Willson VL (2017) Paired Samples T-Test. *Basic and Advanced Statistical Tests*, 17–19 (Rotterdam: SensePublishers), URL `http://dx.doi.org/10.1007/978-94-6351-086-8{_}4`.

Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. *International conference on principles and practice of constraint programming*, 417–431.

Shazeer N, Stern M (2018) Adafactor: Adaptive learning rates with sublinear memory cost. *International Conference on Machine Learning*, 4596–4604.

Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, Lillicrap T, Simonyan K, Hassabis D (2018) A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362(6419):1140–1144, ISSN 0036-8075, URL `http://dx.doi.org/10.1126/science.aar6404`.

Skålnes J, Vadseth ST, Andersson H, Stålhane M (2023) A branch-and-cut embedded matheuristic for the inventory routing problem. *Computers & Operations Research* 159:106353.

Toth P, Vigo D (2002) *The vehicle routing problem* (SIAM).

Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, Bashlykov N, Batra S, Bhargava P, Bhosale S, Bikel D, Blecher L, Ferrer CC, Chen M, Cucurull G, Esiobu D, Fernandes J, Fu J, Fu W, Fuller B, Gao C, Goswami V, Goyal N, Hartshorn A, Hosseini S, Hou R, Inan H, Kardas M, Kerkez V, Khabsa M, Kloumann I, Korenev A, Koura PS, Lachaux MA, Lavril T, Lee J, Liskovich D, Lu Y, Mao Y, Martinet X, Mihaylov T, Mishra P, Molybog I, Nie Y, Poulton A, Reizenstein J, Rungta R, Saladi K, Schelten A, Silva R, Smith EM, Subramanian R, Tan XE, Tang B, Taylor R, Williams A, Kuan JX, Xu P, Yan Z, Zarov I, Zhang Y, Fan A, Kambadur M, Narang S, Rodriguez A, Stojnic R, Edunov S, Scialom T (2023) Llama 2: Open Foundation and Fine-Tuned Chat Models .

Uchoa E, Pecin D, Pessoa A, Poggi M, Vidal T, Subramanian A (2017) New benchmark instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research* 257(3):845–858, ISSN 03772217, URL `http://dx.doi.org/10.1016/j.ejor.2016.08.012`.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. *Advances in neural information processing systems* 30.

Vidal T (2022) Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research* 140:105643, ISSN 03050548, URL `http://dx.doi.org/10.1016/j.cor.2021.105643`.

Vidal T, Laporte G, Matl P (2020) A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research* 286(2):401–416, ISSN 03772217, URL http://dx.doi.org/10.1016/j.ejor.2019.10.010.

Vinyals O, Fortunato M, Jaitly N (2015) Pointer Networks. *Advances in Neural Information Processing Systems*, volume 28 (Curran Associates, Inc.), URL https://papers.nips.cc/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html.

Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3):229–256.