FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of master's thesis

Title:	Vehicle Routing Problem with Time Windows solved via Machine Learning and Optimization Heuristics
Student:	Bc. Adam Zvada
Supervisor:	doc. Ing. Pavel Kordík, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

The thesis will follow these steps:

- Research state of the art methods for solving VRPTW using optimization heuristics
- Research a various solution which leverages machine learning techniques for solving VRPTW
- Implement selected method for VRPTW using optimization heuristics
- Implement selected method for VRPTW using machine learning techniques
- Benchmark implemented methods on a public dataset and discuss their results in detail.

Electronically approved by Ing. Karel Klouda, Ph.D. on 11 February 2021 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Vehicle Routing Problem with Time Windows solved via Machine Learning and Optimization Heuristics

Bc. Adam Zvada

Department of Applied Mathematics
Supervisor: doc. Ing. Pavel Kordík, Ph.D.

April 21, 2021

TODO

Acknowledgements

TODO

Declaration

In Prague on April 21, 2021 . . . .

Citation of this thesis

Zvada, Adam. *Vehicle Routing Problem with Time Windows solved via Machine Learning and Optimization Heuristics.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

TODO

Klíčová slova TODO

Abstract

TODO

Keywords TODO

Contents

List of Figures

Introduction

The VRP is one of the most extensively studied combinatorial problems. It is easy to define but very difficult to solve[7]. The reason VRP is attracting many researchers is the fact that finding a near-optimal solution in a reasonable time would have a great impact on many industries, especially in the domain of transportation and logistics.

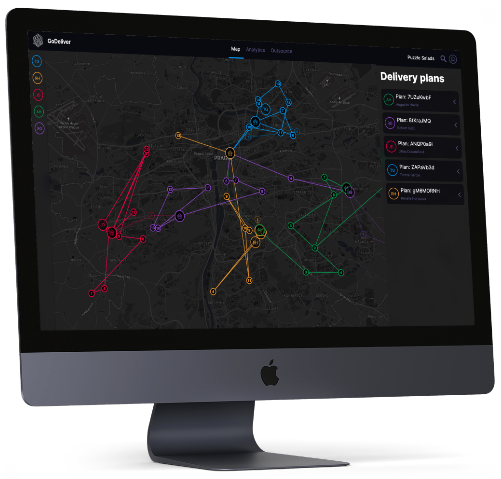

Figure 0.1: GoDeliver dashboard visualizing solution for an instance of vehicle routing problem with time windows.

Motivation

The paradigm shift in logistics business models towards instant gratification of customers are pushing the planning systems to be flexible and dynamic. The environment is constantly changing and planning systems have to update or entirely replan the instance in a short amount of time but maintaining the best delivery efficiency. Having a powerful planning system results in a dramatic reduction of delivery expanses.

Challenges

In the real world, the general VRP problem is not enough to solve the business-related problems. VRP has multiple variants adding various constraints such as capacity, demand or time windows for given set of customers. The vehicle routing problem with time windows (VRPTW) is main focus of this thesis and we will be looking at some novel approaches how to solve it with artificial intelligence (AI).

Assumptions

We expect that leveraging AI or machine learning (ML) techniques to solve VRP would lead in a drastic reduction of computational time for solving given instance of VRP. Moreover, the time complexity would not be exponentially increasing with the problem size. The trade-off lays in the required time to allow AI to train and learn how to solve the problem of vehicle routing.

Thesis structure

The rest of this thesis is organized as follows:

- Chapter 1 presents an formal introduction to Vehicle Routing Problem.
- Chapter 2 provides an advanced theoretical background.
- Chapter 3 describes state-of-the-art solutions in optimization heuristics for solving VRPTW
- Chapter 4 describes state-of-the-art solutions in optimization heuristics for solving VRPTW
- Chapter 5 descImplementation details are explained in
- Chapter 6 summarizes our work, and presents possible future extensions.

Chapter 1

Introduction to Vehicle Routing Problem

The problem objective of VRP is simply finding the shortest route for multiple vehicles to serve all the given set of customers. The shortest route can be differently interpreted based on your minimalization criteria, e.g, traveled distance, time, or a combination of both. It was first proposed by Dantzig and Ramser [8] in 1959, and since then researchers are coming up with different approaches how to solve the problem.

1.1 Vehicle Routing Problem Definition

The general VRP can be defined as a problem in a complete graph $G = (V, E)$ of finding the optimal permutation $\pi_l = (\pi_0, \cdots, \pi_m)$ of nodes V all starting from a node v_0 for given number of paths k which results in minimal traversal cost where $\forall v \in (V \setminus v_0)$ are visited only once. VRP is generalization of traveling salesman problem (TSP) which only has one path.

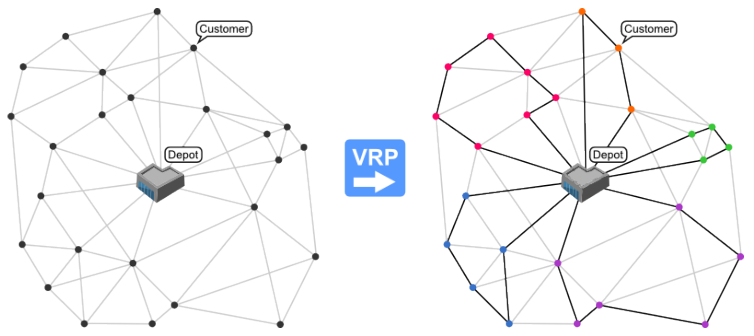

Figure 1.1: Intuitive view of VRP instance on left and proposed solution for 5 vehicles on the right [1]

VRPs are classified as NP-Hard problems which was proved by Lenstra and Kan [7]. It means that in the worst case, adding new nodes, i.e., customers results in an exponential increase of computational complexity.

1.1.1 VRP Notation

Let's introduce our used notation and its real-world interpretation.

- $G = (V, E)$ is a complete undirected graph
  - Network of routes
- v_0 is the initial node
  - A depot
- $V' = (v_1, \cdots, v_n)$ nodes expect the initial node
  - Geographically scattered location of customers
- $E = \{(v_i, v_j) | v_i, v_j \in V, i \neq j\}$ with associated weight as a cost $c : E \rightarrow \mathbb{N}^+$
  - A single route between two locations with associated cost, e.g., distance.
- C is a matrix of edge weights indexed by nodes. c_{ij} where $i, j \in V$
  - Matrix of costs between customers
- $R_i \subset V$ is a path that starts and ends at v_0. $(r_0 = v_0 \wedge r_{|R_i|} = v_0)$
  - Route visit a subset of customers starting and ending at the depot, it can be referred to it as a delivery plan.
- k number of paths
  - Number of vehicles
- $R = R_1, \cdots, R_k$ is a set of paths
  - All routes (delivery plans) for a given instance of VRP.
- $\pi = (\pi_1, \cdots, \pi_k)$ solution for a given instance of VRP.
  - Customer locations in visiting order for multiple vehicles.

Fesability of VRP solution for VRP of routes R is feasible only if each node V_1 is visited exactly once.

The cost of route R_i which we aim to minimize is the sum of its weights (costs). If we operate in Euclidean space, then it is L2 norm of route locations.

$$C(R_i) := \sum_{k=0}^{|R_i|} c_{r_k r_k+1} \tag{1.1}$$

The cost of VRP solution is the sum of route costs.

$$C(R) := \sum_{i=1}^{|R|} C(R_i) \tag{1.2}$$

1.2 Vehicle Routing Flavors

Our modern world heavily relies on complex logistics networks. It requires to synchronize multimodal planning to ship your goods from one side of the world to your doorstep. In order to achieve this, multiple variants and flavours of VRP had to be studied and implemented in the real world use cases. It goes from ordinary variants like measuring the capacity of cars to a more niche problem like eVRP where vehicles are required to make stops to recharge.

All the flavours of VRP can be mutually combined, which is usually the main area of research.

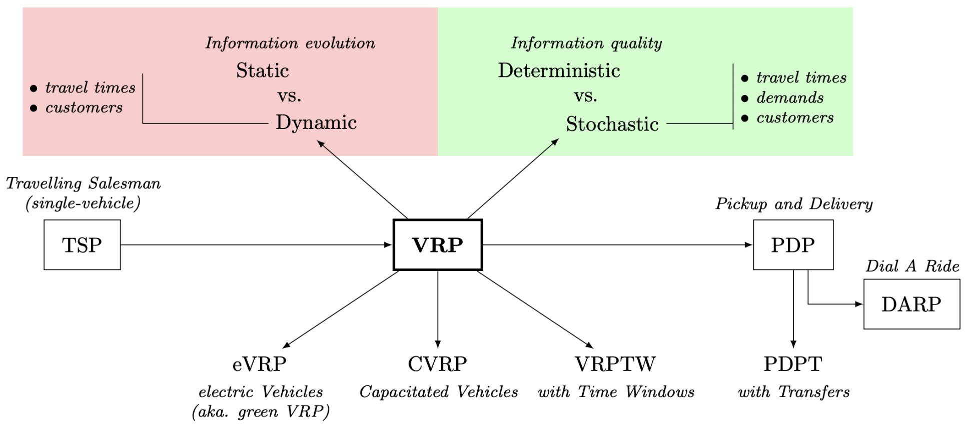

Figure 1.2: Taxonomy of VRPs [2]

The sections below are describing each flavour shown in 1.2.

1.2.1 Capacitated Vehicle Routing Problem

The capacitated vehicle routing problem (CVRP) extends the regular CVRP in introducing a capacity element for each customer. In the literature, it is sometimes referred to as a demand. The customer's demand is $d \in \mathbb{N}^+$ which may represent capacity in the form of weight, size but also in some abstract concepts such as a basket of apples. Additionally, each vehicle has a predefined capacity $Q > 0$.

The CVRP extends the solution feasibility formula by the following capacity constrain.

$$q(R) := \sum_{i \in R} d_i \leq Q \tag{1.3}$$

If the vehicle capacity of the fleet stays the same, we are dealing with CVRP with homogeneous fleet. A fleet with varying capacity for each vehicle is a heterogeneous fleet.

1.2.2 Vehicle Routing Problem with Time Windows

The VRPTW [9] extends the regular VRP by time constraint for each customer. Customers have assigned time window interval $[e_i, l_i]$ where $e_l i < l_i$. The time interval is the request within a vehicle is supposed to visit the node.

The time window can be either implemented as a hard constraint or a soft constraint. Hard constraint forces the vehicle to visit the node, i.e., the customer either in the given time interval or the solution is not feasible. Soft constrains are not strictly enforcing the vehicle to visit the customer, but they introduce a penalty for a violated interval barring a penalty cost. The penalty becomes a part of the cost function which VRP aims to minimize.

In this thesis, we will be focusing on soft constraints for time windows since it is a better reflection of real world use cases. Most businesses allow couriers to arrive late or early, but these types of arrivals are supposed to be minimized.

1.2.3 Pick and Deliver

The Pick and Deliver (PDP) extends the regular VRP by pairing pick and drop with precedence relationships, in which a pickup point must precede the paired delivery point. This flavour of VRP is one of the most complex and even challenging for conventional methods like optimization heuristics algorithms.

The feasibility of a PDP solution is checking whether all delivery points have preceded pickup point.

1.2.4 Static vs. Dynamic

When solving the vehicle routing model, usually we assume that all the input data are static and known with certainty. However, this is not the case in real-life applications where data such as customer demand or travel time are often incomplete or not precise during the planning phase, they are only gradually revealed and specified.

Static VRP does not assume that the input data could be subject to change. The **dynamic VRP** is aware about the information evolution[10] and its goal is to obtain a robust routing planner that will be able to solve already seen instance with subject to small changes without need of recalculating the whole instance again. This is called a priori optimization, after solving a given instance of a combinatorial optimization problem, it becomes necessary to repeatedly solve many other instances with a small variation from the original instance but without reconsideration of the entire problem [11].

In this thesis, the VRP based on AI could be a great candidate for dynamic VRP even though, the entire instance is being recalculated. The reason is that the problem solution is calculated in a seconds instead of minutes and the AI technically already seen the instance in some variation during the training phase.

Dynamic VRP can be achieved with enough robust architecture around the core planner and periodically recalculating the instance with newly revealed information. The planner needs to take its previous solution as an input so the part of the problem does not need to be recalculated. This approach tends to be more exploitative since it is finding a solution in a predefined search space. It would benefit from introducing an explorative element which would diversify the search and could find better cost in a different local optimum.

1.2.5 Deterministics vs. Stochastic

Psaraftis [10] stated that there are two important dimensions of input data, information evolution which is used in dynamic VRP and quality of information for stochastic VRP. The majority of studied VRP models are under the assumption that all the information necessary to formulate the problems is known and readily available. This is true but only for the deterministic settings [12].

A **VRP is stochastic** [13] when some of its data behave as random variables, and the routes must be defined before the values of these random variables become known. Based on the probability distribution of the random variables, we may extract some hidden information and use it to our advantage in the planning process. The newly created plans will have the incorporated stochastic information and the routing decisions may lead to different decision because of the stochastic information being part of the cost function.

A specific real-life example of stochastic VRP would be if we consider electric fleet of shared mobility vehicles and treating locations of Blinkee electric scooters as a random variables. Based on the probability distributions of Blinkee scooter we may predict the time and location where courier will transfer to new fully charged Blinkee scooter. This action will be incorporated into the planned routes.

In contrast, **deterministic VRP** has no random information which could be leveraged before the execution of routes and all the given information are known with certainty. In this thesis, we are focusing on deterministic VRP.

1.2.6 Other Flavours

Dial-a-Ride (DARP) proposed by Wilson et al. [14] in 1971 is a special case of dynamic VRP with pick and deliver. Passengers request a ride at a specified origin and drop location with an optional time window.

Split Delivery VRP [15] is a variant where customers are allowed to be visited more than once. This can be convenient for deliveries of large capacity or stocking fulfillment centers.

Multi Depot VRP is a simplification of the vehicle routing problem with pick and deliver, where pick can happen only on predefined depot locations. This simplification of pickup location is making the problems less complex then vehicle routing problem with pick and deliver (VRPPD).

1.3 VRP in a Real-world

Consumer habits have been shifting towards online and the pandemic situation only accelerated this process. Delivery option is nowadays taken for granted and consumers are demanding a perfect delivery experience. In 2020, there has been shipped over 5.5 billion packages around the world [**?**].

Solving various flavors of efficiently in a reasonable time plays a crucial role for multiple businesses. For example, urban logistics is an essential part of the delivery process, not only it is the last part of the delivery chain, but frequently the courier interacts with the customer and delivery on time with proper ETA prediction is a must. It is also the most expansive part which makes up about 53% of shipment's total cost[**?**]. Urban logistics by large benefits from a better and more optimized which increase the delivery efficiency and reduces the delivery cost. Delivery efficiency has been stated as the biggest challenge in last-mile deliveries by 25% of surveyed companies in Logistics White Paper by ETA[**?**].

At GoDeliver, we are building an autonomous last-mile delivery system and at the core is a planning system which is solving various VRP. The flavour which we are focusing on is Dynamic Capacitated Vehicle Routing Problem with Time Windows and Pick and Delivery (CVRPDPTW). GoDeliver typical use case is on-demand food delivery with multiple depots (pick and deliver), this means that the system has to be dynamic and flexible because a new customers are ordering stochastically for a chosen time window. Another our common use case shown in 1.3 is grocery delivery with multiple depots, time windows, and capacity for customers.

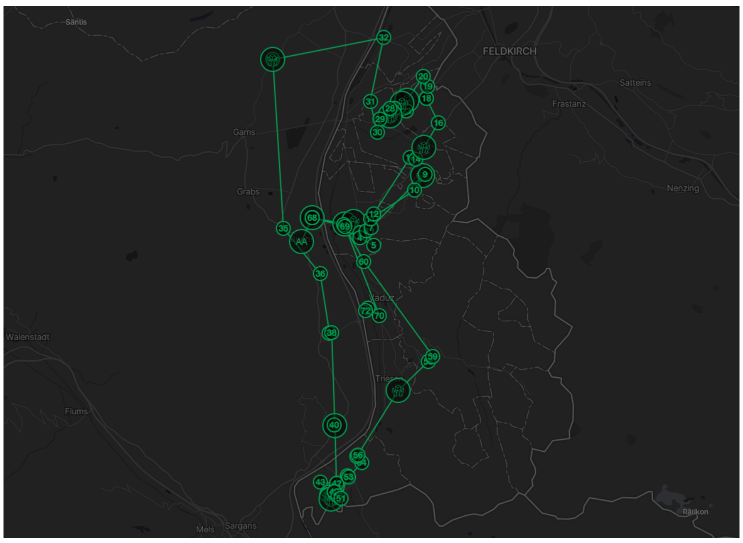

Figure 1.3: Grocery delivery planning with multiple depots from the GoDeliver system

Chapter **2**

Theoretical Background

In this chapter, we will be covering the advanced theoretical background to fully understand the solved task of VRPTW using AI.

2.1 Reinforcement Learning

ML can be divided with a little simplification into three categories; supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is the most common where the model is learned from the provided labeled data. Unsupervised learning, on the other hand, is about finding a hidden patterns in a collection of data with no labels. Finally, reinforcement learning has no labeled data but learns by interacting with the environment and getting feedback in the form of rewards as shown in Figure 2.1.

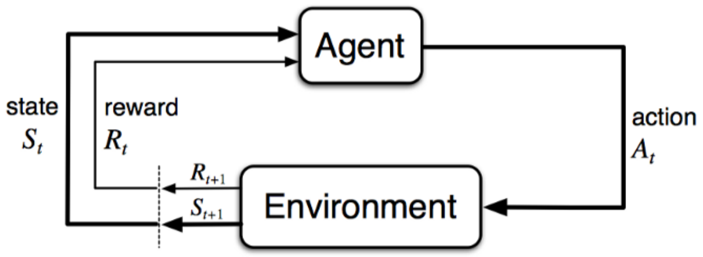

Figure 2.1: Agent feedback loop[3]

The Reinforcement Learning mimics the learning process of humans beings. By experiencing the world and accumulating knowledge, we are learning how to handle novel situations. reinforcement learning (RL) system consists of agent in observed state s_t, the agent interacts with the environment via its actions a_t at discrete time steps t and receives a reward r_{t+1} for given action. The action moves the agent into a new state s_{t+1}. The goal of the agent is to learn a policy π which chooses the action that maximizes the agent's rewards based on the environment [3].

2.1.1 State and Action Value Functions

Transition to a new state gives us a reward and to maximize it, we need a way to quantify how good a state is. A state-value function $V_\pi(s)$ predicts a future reward for a given state when following the policy π [3].

$$V_\pi(s) = \mathbb{E}[G_t | S_t = s] \tag{2.1}$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.2}$$

The equation 2.2 calculates G_t, all future rewards, sometimes called as *return* [3]. The $\gamma \in [0, 1]$is a discount factor and penalizes the rewards in the future, incorporating the possible uncertainty and variance of the future rewards.

We will also define action-value $Q_\pi(s, a)$ which is for a similar purpose as state-value function but predicts the reward for action and state following the policy π.

$$Q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a] \tag{2.3}$$

The decomposition of state-value and action-value function replays on Bellman equations [16]. The decomposition of state-value function is

$$V_\pi(s) = \mathbb{E}[G_t | S_t = s] \tag{2.4}$$

$$V_\pi(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s] \tag{2.5}$$

$$V_\pi(s) = \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \cdots) | S_t = s] \tag{2.6}$$

$$V_\pi(s) = \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \tag{2.7}$$

$$V_\pi(s) = \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s] \tag{2.8}$$

Similarly, this method is applicable to action-value function,

$$Q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s, A_t = a] \tag{2.9}$$

$$Q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma \underset{a \sim \pi}{\mathbb{E}} Q(S_{t+1}, a) | S_t = s, A_t = a] \tag{2.10}$$

2.1.2 Policy Gradients

Policy Gradient [17] is a method for solving the reinforcement learning problem and learning the policy that maximizes the rewards. We define a set of parameters θ that directly models the policy, $\pi_\theta(a|s)$.

To optimize θ for the best reward, we define an objective function [17] as

$$J(\theta) = \sum_{s \in S} d_{\pi_\theta}(s) V_{\pi_\theta}(s) \tag{2.11}$$

where $d_{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ, the probability of ending in a given state [18].

$$d_{\pi_\theta} = \lim_{t \longrightarrow \infty} P(S_t = s | s_0, \pi_\theta) \tag{2.12}$$

The objective function $J(\theta)$ optimizes the θ parameters via gradient ascent [19].

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \tag{2.13}$$

However, computing $\nabla J(\theta)$ is tricky because it depends on the action selection and the stationary distribution of states [20]. Policy gradient can be simplified using Policy Gradient Theorem by Sutton et al. [17].

The proof of policy gradient theorem is quite long and complicated, but you may go through it in this article [20] which is inspired by Sutton and Barto [3]. Policy gradient is simplified to the form as

$$\nabla J(\theta) = \mathbb{E}[\nabla \ln \pi(a|s, \theta) Q_\theta(s, a)] \tag{2.14}$$

2.1.3 REINFORCE

REINFORCE algorithm, proposed by Williams [21] in 1992, is a policy gradient method to update the policy parameter θ.

Let us define the additional terms required by the REINFORCE algorithm. We define a trajectory τ which is a sequence of states, actions, and rewards. Episode is a trajectory which ends at the terminal state S_t.

$$\tau = (S_0, A_0, R_0, S_1, A_1, R_1, \cdots) \tag{2.15}$$

REINFORCE algorithm computes the policy gradient as follows

$$\nabla J(\theta) = \mathbb{E}[G_t \nabla \ln \pi(A_t | S_t, \theta))] \tag{2.16}$$

It is a simplification of a regular policy gradient because $Q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$ and in REINFORCE algorithm we rely on a full trajectory where we can estimate G_t based on Monte-Carlo method which is describe in this article [3].

Algorithm 1: REINFORCE algorithm

```
Result: Updated $\theta$ that maximises reward
Initialize theta at random;
Generate one episode $S_0, A_0, R_0, \cdots, S_T$;
for $t = 1, 2, \cdots, T$ do
    Estimate the the return $G_t$ since the time step t;
    $\theta \leftarrow \theta + \alpha\gamma^t G_t \nabla \ln \pi(A_t|S_t, \theta)$
end
```

In the REINFORCE algorithm, the estimated gradient is highly effected by variance. A technique called baseline $b(S_t)$ is common to be used which subtracts a baseline from the estimated G_t to reduce the variance [22]. The baseline function can be in many forms, but the most common one is to calculate the advantage function $A(s,a) = Q(s,a) - V(s)$ and use it to be subtracted from the gradient.

2.2 Attention

Attention mechanism was first proposed by Bahdanau et al.[23] in 2014 with a problem to help memorize long source sentences in neural machine translation. Not long after that, this concept gave birth to Transformers which dramatically improved many domains, especially Natural Language Processing and brought state-of-the-art results [24].

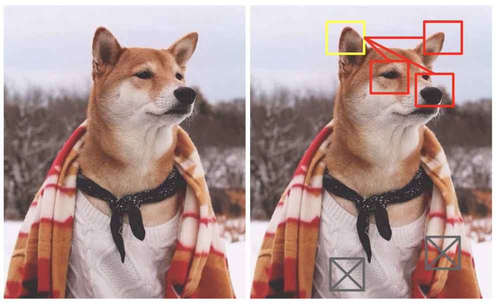

Figure 2.2: A Shiba Inu and what is caughting the network's attention [4], photo credit by @mensweardog.

For humans, visual attention is allowing us to focus on certain regions as visualized on Figure 2.2. Attention mechanism decides on which part of the given source should pay attention to.

2.2.1 Transformer

The Transformer neural network architecture proposed by Vaswani et. al [5] was one of the major breakthroughs in the field of Natural Language processes (NLP). Transformer is getting rid of recurrence [25] in favor of the attention mechanism which allows global dependencies between input and output.

The Transformer architecture is based on encoder-decoder structure as shown on Figure 2.3. Encoder maps the input sequence x to continues representation z which is taken by decoder and decodes it to output sequence y one element at a time. The model is auto-regressive, it takes into account the previously generated output as additional input.

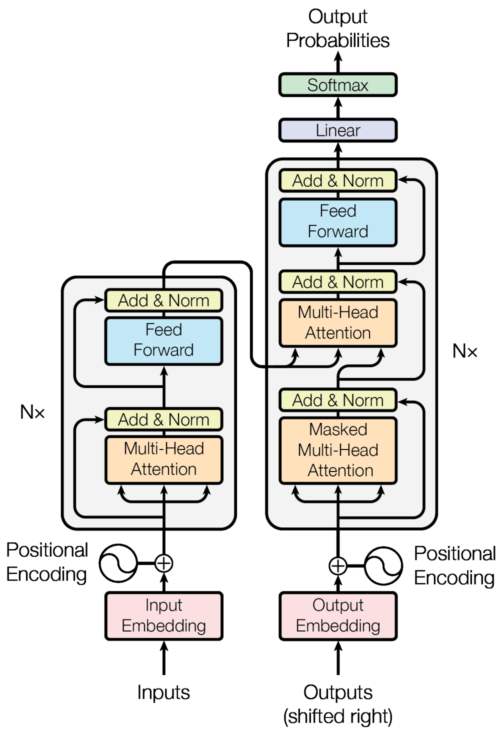

Figure 2.3: The Transformer architecture, encoder on the left and decoder on the right [5].

2.2.1.1 Encoder

The encoder on Figure 2.3 has N identical layers (Vaswani et al. set $N = 6$ [5]). Each layer has two sublayers, the first is multi-head attention 2.2.1.3, and the second is a simple fully connected feed-forward network. Each of the sublayers has a residual connection [26] that sums the input and output of the sublayer and normalize it [27].

$$OutputSubLayer = Norm(x + SubLayer(x)) \tag{2.17}$$

The residual connections are similar to the skip connection which propagates input of the sublayer to the output which helps to avoid exploding or vanishing gradient.

2.2.1.2 Decoder

The decoder on Figure 2.3 is also built from N identical layers and with similar sublayers as the encoder. The first sublayer is masked multi-head attention 2.2.1.3 also with residual connection. The masking mechanism is only there to hide the future information because the Transformers are processing the input one by one. The next two sublayers of the decoder are same as the encoder with the only difference that multi-head attention 2.2.1.3 receives part of the input from the encoder.

2.2.1.3 Multi-Head Attention

The major component of Transformers is Multi-Head Attention (MHA). It runs the input through an attention mechanism h times in parallel. The independent attention outputs are then concatenated and transformed into the expected dimension as shown in Figure 2.4 [4].

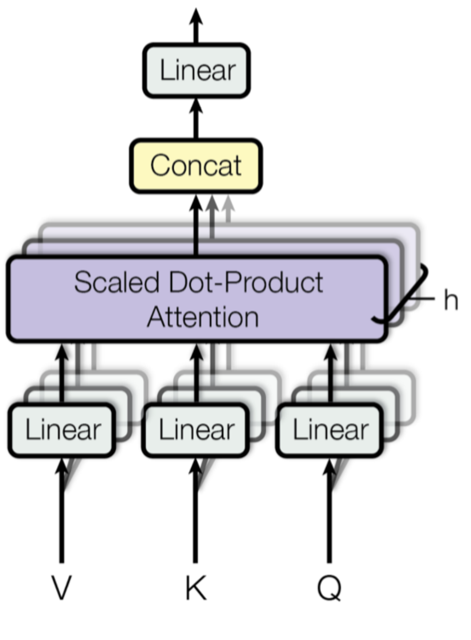

Figure 2.4: Multi-Head Attention component [4]

The Multi-Head Attention takes the embedded input and split the input copy to Key K, Value V and Query Q. They all have the same dimension of input sequence length and embedded vector size d_{model} (Vaswani et al. set $d_{\text{model}} = 512$ [5]).

The matrices K, V, and Q get multiplied for each head $i \in (1, \cdots, \text{h})$ by trainable parameters W_i^K, W_i^V, and W_i^Q, respectively. Then it performs scaled dot-product attention as shown in equations 2.18 and 2.19

$$\text{head}_i = \text{ScaledDotAttention}(QW_i^Q, KW_i^K, VW_i^Q) \tag{2.18}$$

$$\text{ScaledDotAttention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{n}})V \tag{2.19}$$

According to the paper[5], the scaled dot product attention allows the model to jointly attend to information from different representation subspaces at different positions.

Finally, we concatenate all head outputs $head_i$ for $i \in (1, \cdots, \text{h})$ from scaled dot product attention and perform linear transformation by trainable parameter W_0 of Multi-head Attetntion (MHA).

$$\text{MHA}(Q, K, V) = \text{Concat}(\text{head}_1, \cdots, \text{head}_h)W_0 \tag{2.20}$$

2.2.2 Graph Attention Network

To understand the model used for solving VRPTW we need a basic understanding of Graph Attention Network (GAT), first proposed by Veličković et al. [6] in 2017.

It is a neural network architecture capable of operating on graph-structured data. GAT extend the work on Graph Convolution Networks [28] which has a problem with generalizability because they are structure-dependent [29]. GAT address the shortcoming of Graph Convolution Networks (GCN) by leveraging masked self-attentional layers 2.2.

GAT is a single layer (possibly can be stacked) which takes a set of nodes features $h = \{h_0, h_1, \cdots, h_N\}, h_i \in \mathbb{R}^F$ where N is the number of nodes and F is the number of features for a node. The GAT layer assigning different importance to each node through the attention coefficients.

$$e_{ij}^l = a^l(W^l h_i^l, W^l h_j^l) \tag{2.21}$$

The equation 2.21 is a linear transformation of node features and weight matrix W^l which is used for calculating the attention coefficients via learnable parameter a^l. It is gathering information about the importance of node j's features to node i.

$$\alpha_{ij}^l = \text{softmax}_j(e_{ij}^l) = \frac{exp(e_{ij}^l)}{\sum_{k \in N_i} exp(e_{ik}^l} \tag{2.22}$$

The graph structural data are represented in attention via equation 2.22 which is computed only for all neighbour nodes N_i of node i. The softmax function is used to make the attention scores comparable across different nodes.

$$h_i^{l+1} = \sigma(\sum_{j \in N_i} \alpha_{ij}^l W^l h_i^l) \tag{2.23}$$

The equation 2.23 is aggregating together all attention embeddings of all neighbour nodes. If we perform multi-head attention on the final layer of GAT then computation of the final high level feature embedding is as follows

$$h_i^{l+1} = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N_i} \alpha_{ij}^l W^k h_i^l) \tag{2.24}$$

The aggregation process of a multi-head graph attentional layer for a given node is illustrated on Figure 2.5.

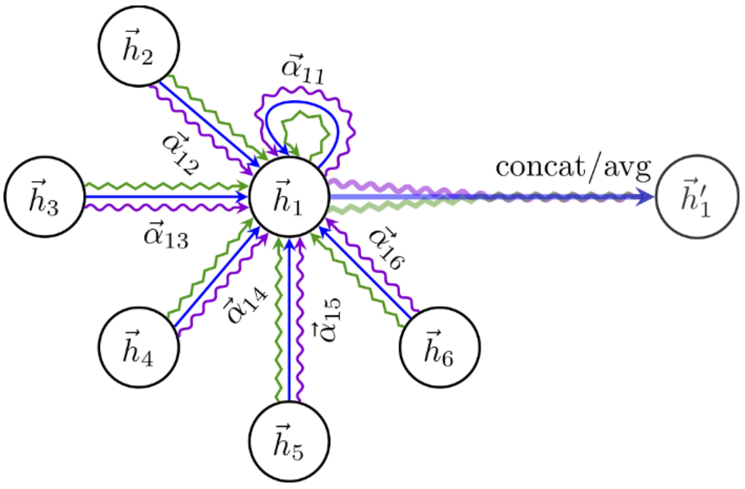

Figure 2.5: Multi-head attention with $K = 3$ heads [6].

Chapter 3

VRPTW via Optimization

3.1 Insertion Heuristics

TODO

3.2 Genetic Approach

TODO

3.3 Google OR-Tools

TODO

3.4 Hybrid Adaptive Large Neighbourhood Search

TODO

Chapter 4

VRPTW via AI

In this chapter, we will review the current end-to-end AI methods for solving VRP and propose our extension for solving VRPTW.

Machine learning and artificial intelligence have been replacing many hand-engineered algorithms and providing state-of-the-art results. In recent years, reinforcement learning 2.1 and advances in attention models 2.2 has shown great promise to disrupt the field of heuristics algorithms [30, 31, 32]. Heuristics algorithms [33] are incomplete methods that can compute solutions efficiently, but are not able to prove the optimality of a solution. Most of the business challenges do not require the most optimal exact solution [34] but focus on approximation of the optimal solution in a reasonable time.

4.1 Related Work

Researchers have been mainly active in solving general VRP... TODO

4.2 Solution

The end-to-end AI method pro solving VRPTW is extension of the work done by Kool et al. [31].

Let us describe the high-level concept behind the method. Consider we have a model as blackbox which takes VRPTW instance as an input and outputs probabilities for all the VRPTW nodes. The probability represents which node should be visited next and by following to the most probable node we get a partial solution which will be considered by the blackbox. We iterate this process until all nodes have been visited and we acquire a feasible plan as shown on Figure 4.1.

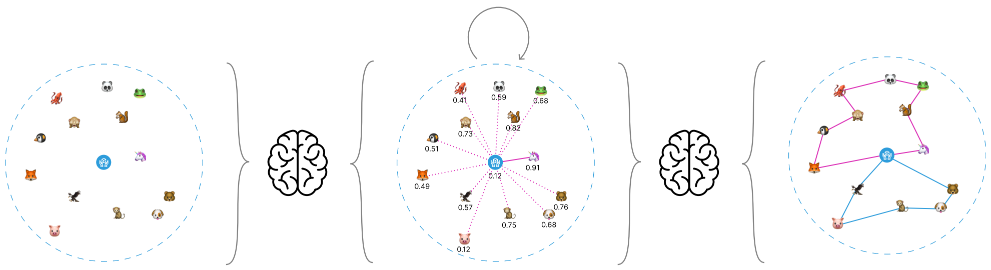

Figure 4.1: High-level concept behind the used method.

4.2.1 Model Architecture

The model architecture by [31] extended by time windows is leveraging recent advancements in attention mechanism. The model is built upon transformers 2.2.1, graph attention network 2.2.2, and reinforcement learning 2.1. The network structure is encoder-decoder which is great for solving sequential decision problems. The structural input instance is extracted by the encoder **??** and then the solution is incrementally constructed by the decoder **??**.

The VRPTW input instance is consited from

- $X = \{x_1, \cdots, x_n\}$ where x_i is two-dimensional coordinates in the euclidean space.
- x_0 is the location of depot.
- $D = \{d_1, \cdots, d_n\})$ is the demand capacity for each of the locations.
- $T = \{(e_1, l_1), \cdots, (e_n, l_n)\})$ is time windows for each of the location where e_i is the beginning and l_i is the end of the considered time window.

The output is the solution of VRPTW instance and is represented as a permutation π of locations $X \cup x_0$.

- $\pi = \{\pi_1, \cdots, \pi_T\} \in \{x_0, \cdots, x_n\})$

4.2.1.1 Encoder

The encoder uses graph attention network 2.2.2 to embed the node features to graph embedding. Then the decoder architecture is the same as the decoder of transformer 2.2.1. Typicaly, the decoder of transformer uses positional encoding [35] to embed the input, but in this case it has been replace with GAT since we deal with graph-based structure.

4.2.1.2 Decoder

4.2.2 Reinforcement Learning

4.2.3 Integrating Distance Matrix

Multidimensional scaling

Chapter 5

Implementation

5.1 VRPTW via Optimization

5.2 VRPTW via AI

Chapter 6

Evaluation

Choosing TW penalties Model genealization Training process - loss + used parametes I1 vs Or-tools vs RL

Conclusion

TODO

Bibliography

[1] Available from: `https://neo.lcc.uma.es/vrp/vehicle-routing-problem/`

[2] Bono, G. Deep multi-agent reinforcement learning for dynamic and stochastic vehicle routing problems. , no. 2020LYSEI096, Oct. 2020. Available from: `https://tel.archives-ouvertes.fr/tel-03098433`

[3] Sutton, R. S.; Barto, A. G. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018, ISBN 0262039249.

[4] Weng, L. Attention? Attention! *lilianweng.github.io/lil-log*, 2018. Available from: `http://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html`

[5] Vaswani, A.; Shazeer, N.; et al. Attention Is All You Need. *CoRR*, volume abs/1706.03762, 2017, `1706.03762`. Available from: `http://arxiv.org/abs/1706.03762`

[6] Veličković, P.; Cucurull, G.; et al. Graph Attention Networks. 2018, `1710.10903`.

[7] Lenstra, J. K.; Kan, A. H. G. R. Complexity of vehicle routing and scheduling problems. *Networks*, volume 11, no. 2, 1981: pp. 221–227, doi:https://doi.org/10.1002/net.3230110211, `https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230110211`. Available from: `https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230110211`

[8] Dantzig, G. B.; Ramser, J. H. The Truck Dispatching Problem. *Management Science*, volume 6, no. 1, 1959: pp. 80–91. Available from: `https://EconPapers.repec.org/RePEc:inm:ormnsc:v:6:y:1959:i:1:p:80-91`

[9] Solomon, M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Oper. Res.*, volume 35, 1987: pp. 254–265.

[10] Psaraftis, H. A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem. *Transportation Science*, volume 14, 05 1980: pp. 130–154, doi:10.1287/trsc.14.2.130.

[11] Bertsimas, D.; Jaillet, P.; et al. A Priori Optimization. *Operations Research*, volume 38, 02 1989, doi:10.1287/opre.38.6.1019.

[12] Toth, P.; Vigo, D.; et al. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. USA: Society for Industrial and Applied Mathematics, 2014, ISBN 1611973589.

[13] Gendreau, M.; Laporte, G.; et al. Stochastic vehicle routing. *European Journal of Operational Research*, volume 88, no. 1, 1996: pp. 3–12, ISSN 0377-2217, doi:https://doi.org/10.1016/0377-2217(95)00050-X. Available from: `https://www.sciencedirect.com/science/article/pii/037722179500050X`

[14] STEIN, D. M. Scheduling Dial-a-Ride Transportation Systems. *Transportation Science*, volume 12, no. 3, 1978: pp. 232–249, ISSN 00411655, 15265447. Available from: `http://www.jstor.org/stable/25767916`

[15] DROR, M.; TRUDEAU, P. Savings by Split Delivery Routing. *Transportation Science*, volume 23, no. 2, 1989: pp. 141–145, ISSN 00411655, 15265447. Available from: `http://www.jstor.org/stable/25768367`

[16] Bellman, R. E. *The Theory of Dynamic Programming*. Santa Monica, CA: RAND Corporation, 1954.

[17] Thomas, P. S.; Brunskill, E. Policy Gradient Methods for Reinforcement Learning with Function Approximation and Action-Dependent Baselines. 2017, `1706.06643`.

[18] j2kun. Markov Chain Monte Carlo Without all the Bullshit. Sep 2016. Available from: `https://jeremykun.com/2015/04/06/markov-chain-monte-carlo-without-all-the-bullshit/`

[19] Ng, A. CS229 Lecture notes - Supervised learning, 2012.

[20] Weng, L. Policy Gradient Algorithms. *lilianweng.github.io/lil-log*, 2018. Available from: `https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html`

[21] Williams, R. J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, volume 8, 1992: pp. 229–256.

[22] Available from: `https://danieltakeshi.github.io/2017/03/28/going-deeper-into-reinforcement-learning-fundamentals-of-policy-gradients/`

[23] Bahdanau, D.; Cho, K.; et al. Neural Machine Translation by Jointly Learning to Align and Translate. 2016, `1409.0473`.

[24] Radford, A.; Wu, J.; et al. Language Models are Unsupervised Multitask Learners. 2019.

[25] Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural computation*, volume 9, 12 1997: pp. 1735–80, doi:10.1162/neco.1997.9.8.1735.

[26] He, K.; Zhang, X.; et al. Deep Residual Learning for Image Recognition. 2015, `1512.03385`.

[27] Ba, J. L.; Kiros, J. R.; et al. Layer Normalization. 2016, `1607.06450`.

[28] Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. 2017, `1609.02907`.

[29]

[30] Cappart, Q.; Moisan, T.; et al. Combining Reinforcement Learning and Constraint Programming for Combinatorial Optimization. 2020, `2006.01610`.

[31] Kool, W.; van Hoof, H.; et al. Attention, Learn to Solve Routing Problems! 2019, `1803.08475`.

[32] Kool, W.; van Hoof, H.; et al. Deep Policy Dynamic Programming for Vehicle Routing Problems. 2021, `2102.11756`.

[33] *Local Search in Combinatorial Optimization*. Princeton University Press, 2003. Available from: `http://www.jstor.org/stable/j.ctv346t9c`

[34] Lawler, E. L.; Wood, D. E. Branch-And-Bound Methods: A Survey. *Operations Research*, volume 14, no. 4, 1966: pp. 699–719, ISSN 0030364X, 15265463. Available from: `http://www.jstor.org/stable/168733`

[35] Transformer Architecture: The Positional Encoding. Available from: `https://kazemnejad.com/blog/transformer_architecture_positional_encoding/`

Appendix **A**

Acronyms

AI artificial intelligence

CVRP capacitated vehicle routing problem

GAT Graph Attention Network

GCN Graph Convolution Networks

MHA Multi-head Attetntion

ML machine learning

PDP Pick and Deliver

RL reinforcement learning

TSP traveling salesman problem

VRP vehicle routing problem

VRPPD vehicle routing problem with pick and deliver

VRPTW vehicle routing problem with time windows

Appendix **B**

Media contents

```
readme.txt ........................ the file with CD contents description
data ........................................ the data files directory
  example_sequence . the directory with example sequence from dataset
    *.jpg ................................... the example images
  example_sequence_graphs ..... the directory of graphs of experiments
    *.png ............................... the motion output graphs
  tracking_example.mp4 ...................... the example video file
src .................................... the directory of source codes
  models ...................... the directory of deep learning modules
  utils ............................... the directory of helper modules
  *.py ...................................... the Python source files
text ........................................ the thesis text directory
  thesis ............. the directory of LaTeX source codes of the thesis
  thesis.pdf ..................... the Diploma thesis in PDF format
```