Universiteit Leiden

POMO on the VRPTW: Comparing Constraint-Handling Approaches

Paul Tielens (s3612031)

Supervisors:
Furong Ye & Niki van Stein

Bachelor Thesis– Informatica

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

July 23, 2026

Abstract

The Vehicle Routing Problem with Time Windows (VRPTW) is a type of Combinatorial Optimization Problem (COP), a fleet of vehicles capacity-constrained vehicles must visit a set of customers, within a specified time window. This is a computationally challenging problem where the solution space grows exponentially with the problem size. The Policy Optimization with Multiple Optima (POMO) algorithm has been proved to be effective on the Capacitated Vehicle Routing Problem (CVRP), but its adaptation to the VRPTW remains underexplored. In this Bachelor Project Thesis we test two main methods: hard/soft constraint, with three soft constraint submethods: set/decaying penalty value and Lagrangian relaxation, totalling four POMO variants. Each variant is trained and tested on multiple different types of problem instances for their base performance and their performance on unseen data. The results prove the hard constraint yields the strongest and most robust performance without the need for hyperparameter optimization. Soft constraint methods offer flexibility by trading feasibility for solution quality, which may make them more suitable for real world applications where strict feasibility is not required. Overall, there remains room for further research into the mechanisms and behaviour of the Soft Constraint Methods for Neural Combinatorial Optimization (NCO).

Contents

1 Introduction

Combinatorial Optimization Problems (COP) cover a diverse set or real-world and theoretical problems and can be found in logistics, distribution and planning. Finding optimal solutions remains computationally challenging due to the NP-hard nature of many such problems, where the solution space grows exponentially with the problem size. A wide range of solutions methods have been explored, from classical exact solvers and hand-crafted heuristics, to more recent learning-based approaches. Neural Combinatorial Optimization (NCO) [2] has gained traction, using deep learning to construct high-quality solutions with low inference time. A notable example is the Policy Optimization with Multiple Optima (POMO) algorithm [9], which combines reinforcement learning with a Transformer-based attention architecture and has demonstrated strong performance on standard benchmarks such as the Capacitated Vehicle Routing Problem (CVRP). However, many evaluations of POMO and its improvements has been based only on the CVRP, which doesn't match modern day's complexity and real-world routing scenarios. In practice, deliveries are often subject to time window schedules, where vehicles are required to arrive within a certain time interval. So, our goal is to adapt POMO to suit the Vehicle Routing Problem with Time Windows (VRPTW) using multiple mechanisms and charting their performance and behaviour. This gives the following research question that we would like to answer:

How to adapt POMO to deal with the time windows constraints in the Vehicle Routing Problem with Time Windows?

1.1 Thesis overview

Section 2 provides the necessary background for understanding this research; Section 3 introduces some additional related work; in Section 4 we discuss what experiments we run to collect the data and to answer the research question; the experiment setup, in which we provide further details on the experiment settings, and the experiment results can be found in Section 5; our conclusions of the test results are provided in Section 6; in Section 7 we discuss the project further and finally we provide ideas on further research in Section 8.

This bachelor project thesis for the bachelor Computer Science was conducted and written under the supervision of Furong Ye and Niki van Stein, and is part of the Leiden Institute of Advanced Computer Science (LIACS).

2 Background

2.1 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is an NP-hard problem that can be categorized as a Combinatorial Optimization Problem (COP) where the goal is to find an optimal solution from a finite but usually astronomically large set of solution candidates [13]. In VRP a number of identical vehicles will need to serve a set of customers from a single depot. Let $G = (V, A)$ be a complete graph, where $V = \{0, 1, ..., n\}$ is the set of nodes and $A = \{(i, j) : i, j \in V, i \neq j\}$ is

the set of arcs, each with an associated travel cost c_{ij} which is the Euclidean distance between nodes i and j. Node 0 represents the depot with coordinates $x_0 \in \mathbb{R}^2$ and the remaining nodes $i \in \{1, ..., n\}$ represent the set of n customers with coordinates $x_i \in \mathbb{R}^2$. The travel cost is thus computed by $c_{i,j} = \|x_i - x_j\|_2$. A solution or trajectory τ is defined as a sequence of node visits: $\tau = (\pi_0, \pi_1, \pi_2, ..., \pi_M)$, where $\pi_t \in V$ denotes the node visited at step t, with $\pi_0 = \pi_M = 0$ (the trajectory starts and ends at the depot). Every customer node must be visited exactly once across the trajectory $(\pi_t)_{t:\pi_t \neq 0}$ is a permutation of $(1, 2, .., n)$ While the depot node 0 may appear multiple times within τ, marking the end of one vehicle's route and the start of the next. The total travel cost of a trajectory is given by:

$$d(\tau) = \sum_{t=0}^{M-1} c_{\pi_t \pi_{t+1}} \tag{1}$$

The objective is to find the trajectory that minimizes this cost:

$$\min_{\tau} d(\tau) \tag{2}$$

Some constraints can be added to increase the difficulty of the task and/or to match reality.

2.1.1 Capacitated Vehicle Routing Problem

The first constraint is the capacity constraint. In the Capacitated Vehicle Routing Problem (CVRP) each customer node $i \in \{1, ..., n\}$ has been assigned an additional demand δ_i, with $0 < \delta_i \leq D$ where $D > 0$ is the fixed capacity that all the vehicles can carry [8]. The depot has no demand, so $\delta_0 = 0$. A vehicle route within the trajectory τ is defined as a maximal subsequence between two consecutive depot visits. Formally, if $\pi_{t_1} = \pi_{t_2} = 0$ are two consecutive occurrences of the depot in τ, the route τ^k they enclose is $\tau^k = (\pi_{t_1}, \pi_{t_1+1}, ..., \pi_{t_2})$ and $\tau^1, ..., \tau^K$ denotes the full set of routes in τ, where K is the total number of routes. The total demand served by each route must not exceed the vehicle capacity:

$$\sum_{t=t_1}^{t_2} \delta_{\pi_t} \leq D \qquad \forall k \in \{1, ..., K\} \tag{3}$$

k is the number of routes in a trajectory. This constraint restricts the set of feasible trajectories: a trajectory τ is only valid if every rout it encodes satisfies Equation 3. The objective remains the same as before described in Equation 2.

2.1.2 Vehicle Routing Problem with Time Windows

The Vehicle Routing Problem with Time Windows (VRPTW) extends the CVRP and incorporates the time window constraint [11]. Each node $i \in V$ has a time window $[e_i, l_i]$, where e_i is the earliest allowed arrival time and l_i is the latest allowed arrival time, as well as a service time $s_i \geq 0$ representing the duration required to serve node i once a vehicle arrives. The depot's time window $[e_0, l_0]$ represents the overall time window in which all routes must be completed. Let a_t denote the arrival time of the vehicle at node π_t. Arrival times are defined recursively along the trajectory: the vehicle departs node π_t after completing service, and travels to π_{t+1}, if a vehicle arrives earlier than a node's available time e_i then the vehicle can wait until the node is available. Whenever the vehicle departs from the depot ($\pi_t = 0$), the arrival-time clocks resets to e_0, since each route is

served by a fresh vehicle beginning at the start of the overall time window, this is described using the following formula:

$$a_0 = e_0, \qquad a_{t+1} = \begin{cases} e_0 & \text{if } \pi_{t+1} = 0 \\ \max(a_t + s_{\pi_t} + c_{\pi_i \pi_{t+1}}, e_{\pi_{t+1}}) & \text{otherwise} \end{cases} \tag{4}$$

A trajectory τ is time-feasible only if every node is reached no later than its latest allowed arrival time:

$$a_t \leq l_{\pi_t} \qquad \forall t = 0, ..., M \tag{5}$$

Together with the capacity constraint from Equation 3, this defines the feasible set for VRPTW: a trajectory τ is valid only if it satisfies both the capacity constraint at every route segment and the time window constraint at every node visit. The objective remains the same as in Equation 2, minimizing total travel cost $d(\tau)$.

The addition of the time windows constraint significantly increases the complexity of the VRP. Without the time windows the solution space consists of all the possible permutations of the customer visits and the optimization is solely focussed on optimizing the distance travelled. However, the time windows add a temporal dimension which drastically reduces the feasible solution space. Additionally, customers must also be visited in their allotted time windows, meaning that the order in which nodes are visited is also crucial.

2.2 Policy Optimization with Multiple Optima.

POMO was introduced in 2021 [9], it runs multiple rollouts and generates N sequences of nodes with N different starting nodes. The model will calculate N solution trajectories $\{\tau^1, \tau^2, ..., \tau^N\}$ in parallel. VRP and its variants are very dependent on the starting positions, as the first node may exclude a subset of possible paths which may contain the optimal solution(s). So POMO is used to explore multiple starting positions and learn to literally tackle the problem from multiple angles. An Attention Model (AM) [8] is used for the policy network and choosing the next node in the rollout, the AM consists of an encoder and a decoder. The encoder encodes each node and its relations with other nodes into embeddings. The decoder generates a solution sequence step by step using these embeddings. The nodes that have already been visited and are unavailable due to constraints, will be masked and their embeddings will not be used in the decoder. The rewards are calculated for each sequence. These rewards are used for further improving the policy network and updating the policy gradient based on the REINFORCE method [15]. The reward or total return $R(\tau^i)$ is simply the negative total travel cost of a trajectory in Equation 1:

$$R(\tau) = -d(\tau) \tag{6}$$

We use $R(\tau)$ to increase our expected reward J, we calculate this using Gradient ascend with an approximation:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} (R(\tau^i) - b^i(s)) \nabla_\theta \log p_\theta(\tau^i | s) \tag{7}$$

where $p_\theta(\tau^i|s) \equiv \prod_{t=2}^{M} p_\theta(a_t^i|s, a_{(1:t-1)}^i)$, b^i is in POMO the shared baseline where the average is taken of all the rewards combined:

$$b^i(s) = b_{shared}(s) = \frac{1}{N}\sum_{j=1}^{N} R(\tau^j) \quad \text{for all } i. \tag{8}$$

This makes POMO an effective construction type Reinforcement Learning method for solving COP's.

We will test the effectiveness of POMO on VRPTW in this bachelor thesis, for this we need to extend the POMO algorithm to actually work on VRPTW. First, the features of the nodes are expanded with the information of the time windows, so the input for the decoder is expanded accordingly. Second, POMO will be further adjusted and constrained, so that it will actually adhere to the time windows constraint. For this, we have chosen two types of constraint.

2.3 Constraint handling

2.3.1 Hard Constraint

The first constraint for adapting POMO to the VRPTW is the hard constraint. Here, the time windows constraint may not be violated. The time windows are enforced by not only masking the already visited nodes, but additionally masking the nodes that are unavailable because their time windows have already passed. The capacity constraint is similarly implemented, the nodes will be masked if the demand of a customer is more than a vehicle has in its current capacity. If all the nodes are masked, then the vehicle will return to the depot and start a new fresh route, which will ensure feasibility. The reward of the hard constraint is simply the negative total travel cost of a trajectory which is already described in Equation 6.

2.3.2 Soft Constraint

We have chosen soft constraint as the second mechanism. Here, the time windows constraint does not have to be adhered, and may be violated in exchange for a penalty per violation. For calculating the reward, we use $R(\tau)$ from Equation 6 added with the penalty value times the number of violations:

$$R_{\text{soft}}(\tau) = -d(\tau) - \lambda \sum_{t=0}^{M} 1[a_{\pi_t} > l_{\pi_t}] \tag{9}$$

λ is the added penalty per violation which can be set, a_{π_t} is the arrival time at node π_t, l_{π_t} represents the end time of the time window at node π_t.

Linear, convex and concave λ decay The value of λ in Equation 9 can be set and changed over time and is quite flexible. So, we could set an initial λ value and decrease it over time, this way we hope to encourage finding feasible solutions in the beginning and to find better solutions over time. We have chosen linear, concave and convex decrease of the λ value over time for variation and varying behaviour. For calculating the λ values we use the following:

$$u = \frac{k}{K-1} \tag{10}$$

Where k is the current epoch and K the total number of epochs. Linear decay:

$$\lambda_k = \lambda_{\text{initial}} - (\lambda_{\text{initial}} - \lambda_{\text{min}}) \cdot u \tag{11}$$

Convex or exponential decay:

$$\lambda_k = \lambda_{\text{initial}} \cdot \left(\frac{\lambda_{\text{min}}}{\lambda_{\text{initial}}} \right)^u \tag{12}$$

Concave or quadratic decay:

$$\lambda_k = \lambda_{\text{initial}} - (\lambda_{\text{initial}} - \lambda_{\text{min}}) \cdot u^2 \tag{13}$$

The values of λ are visualized in Figure 1.

Lagrangian Relaxation For our final mechanism we will use Lagrangian Relaxation [12]. The idea with Lagrangian Relaxation is that the penalty value will increase and eventually converge by increasing the penalty value at every violation. So, the penalty value will start low and the model will have a count of violations of the time window constraint in the early stages. However, as the penalty values progressively increases, in turn will the number of violations decrease, until the penalty values will converge to a certain value as the number of violations approach zero. We re-use Equation 9 but λ is replaced by the epoch-dependent λ_k, and will be computed as follows:

$$R_{\mathcal{L}}(\tau) = -d(\tau) - \lambda_k \sum_{t=0}^{M} 1[a_{\pi_t} > l_{\pi_t}] \tag{14}$$

For calculating λ at epoch k, we will use:

$$\lambda_{k+1} = \max(0, \lambda_k + \alpha \cdot \bar{v}_k) \tag{15}$$

k represents the epoch, λ_k is the penalty value at epoch k, α is the learning rate and $\bar{v}_k$ represents the average number of violations of a trajectory at epoch k.

3 Related Work

Many modifications have already been researched for further improvements on POMO, by selecting better starting positions [5], optimizing the policy [10][14], exploiting rotational and reflection invariance of the nodes in COPs [6].

The VRPTW has also been addressed using neural methods. The Attention Model, which is already integrated in POMO, has directly been applied to the VRPTW [1]. Another framework was proposed MVMoE, where a unified solver based on a mixture-of-experts architecture was trained across multiple VRPT variants simultaneously, including the VRPTW [16]. Another novel idea, the PIP framework, combines a Lagrangian multiplier with preventative infeasibility masking to steer solution construction away from infeasible regions [4].

4 Approach

Ten models will be trained based on the described mechanisms in Sections 2.3.1 and 2.3.2 to answer our research question. We log important metrics during training, such as the reward, mean violations, tour length and the λ value for the soft constraint models that use Lagrangian relaxation. The ten models that are trained are distributed as follows: one model is based on the hard constraint, three on the soft constraint models with a set penalty value $\lambda = 10, 50$ and 75, three soft constraint models with decreasing penalty values: linear, concave and convex, two soft constraint models based on Lagrangian relaxation with a start penalty value $\lambda_0 = 0$ and varying learning rate values $\alpha = 0.1, 0.5$, and a final Lagrangian relaxation model with a starting penalty value $\lambda_0 = 40$ and learning rate $\alpha = 0.5$. We have set the penalty values to 10, 50, and 75, as this range provides insight into the effect of penalty magnitude on model performance. For the soft models with decaying penalty value, the initial penalty value $\lambda_0 = 200$ and decreases to $\lambda_{100} = 20$ over 100 training epochs, the trajectory can be found in Figure 1. We have chosen learning rates α 0.1 and 0.5 to provide insight into how the learning rate affects the convergence speed of the penalty value and the resulting number of violations. Finally, an additional Lagrangian relaxation model with $\lambda_0 = 40$ was computed to examine the effect of giving the model a 'head start' on the penalty value.

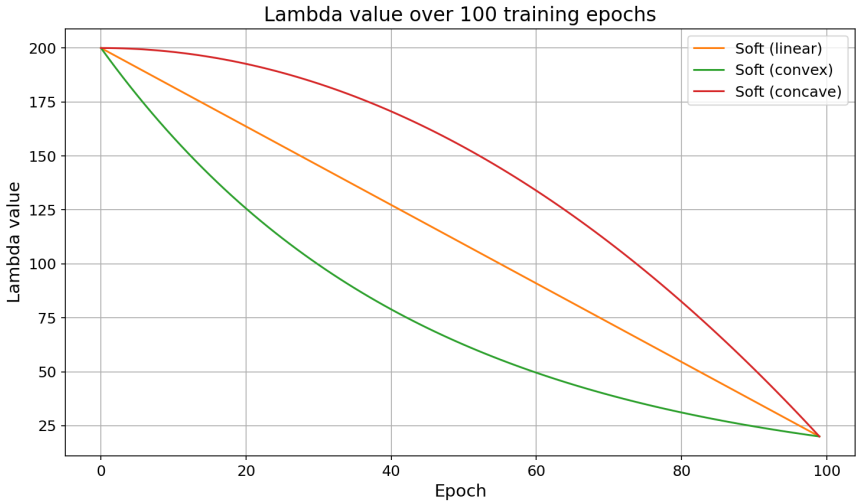
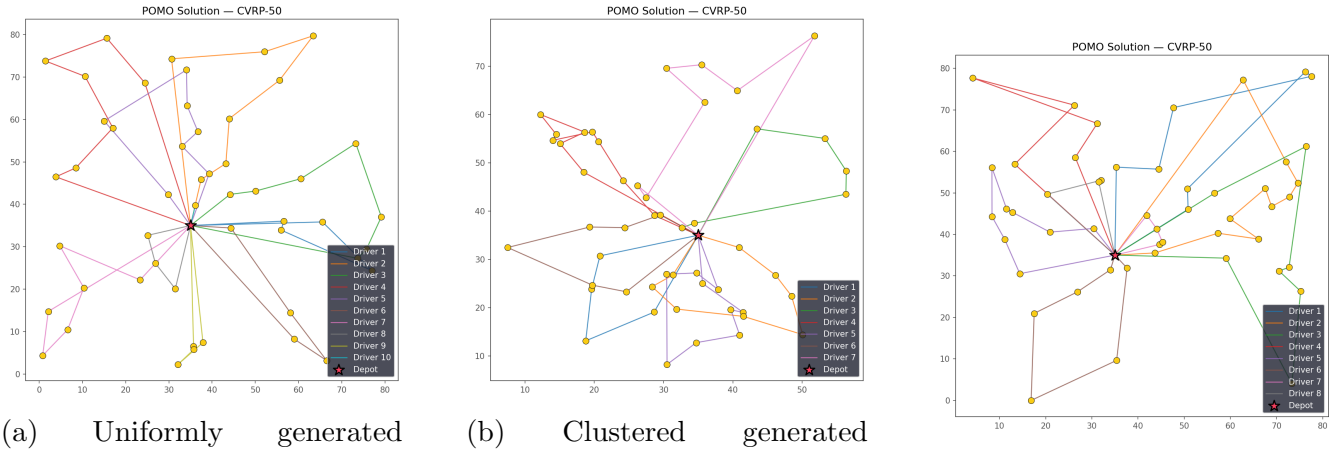


Figure 1: Linear, concave and convex decrease of the λ value over 100 training epochs. Starting from 200 decreasing to 20.

We test our models on three types of problem instances, distinguished by how customer node locations are generated: uniformly distributed, clustered, and a mix of both. These problem instances are drawn from the Solomon benchmark [11]. The process of generating the uniform locations of the nodes is identical during testing as during training. The clustered locations are generated by first sampling k cluster centres uniformly within the coordinate space, each node is randomly assigned to one of the k centres, the coordinates are sampled from a Gaussian distribution centred on the assigned centre with a standard deviation of σ. For the mixed generation one half consists of uniformly generated locations and the other half are clustered generated locations. Examples of the three types of problem instances can be found in Figure 2. We will run the same

100 problem instances for every model on the VRPTW50 or VRPTW100 respectively and calculate the average tour length, number of violations and number of routes per type of problem instance, which can be found in the tables of Section 5.3.1, 5.3.2 and 5.3.3. Additionally, we will use our models that are trained on the VRPTW50 instances, to test them on the VRPTW100 instances, for generalization, which can be found in Section 5.3.4. We will only test on uniformly generated locations of the nodes as we are only interested in the generalization ability of the models on higher number of nodes and not additionally with another type of problem instance. For this test, we will generate the same 100 problem instances that are also used for the uniformly generated problem instances, for a more fair and reliable comparison.



(a) Uniformly generated VRTPW50 solution using 10 routes

(b) Clustered generated VRPTW50 solution using 7 routes

(c) Mixed generated VRPTW50 solution using 8 routes

Figure 2: Solutions computed by the hard constraint model on uniformly, clustered and mixed VRPTW50 generated problem instances.

We can draw conclusions from the observed metrics during training, the performance results from testing and compare the results on the VRPTW100 from the VRPTW50 and VRPTW100 models, to examine the generalization ability of the models.

5 Experiments

5.1 Setup

Hardware The computer that we use contain an Intel Core i7-14700 as CPU with 20 cores divided into 8 HyperThreaded performance-cores and 12 efficient-cores, a 32 GB RAM, NVidia GeForce GTX4060 with 8 GB VRAM and finally a 1 TB NVMe.

Software We have directly used the POMO and VRPTW implementations of RL4CO version 0.6.0 [3] for the Hard Constraint model, and a modified version of the RL4CO implementation for the Soft Constraint and Lagrangian Relaxation models.

Training We have used Adam as our optimizer [7] during training with a learning rate of 0.0001. Training was conducted with a batch size of 64, the models will train for 100 epochs, training on 1 epoch for VRPTW100 takes about 5 to 6 minutes and 2.5 to 3 minutes for VRPTW50. The training instances are uniformly generated where location coordinates can range from 0.0 to 80.0, e_0 (the earliest allowed time) is set to 0 and l_0 (the latest allowed time, i.e. the end of the overall time window) is set to 230, the minimal demand of a node is 1 and the maximum is 41, the capacity D is 200 for every vehicle, and the depot is always located at $(35, 35)$. We have based these settings on the Solomon Benchmark for VRPTW [11]. The number of starting positions (N) is equal to the number of nodes. So N would be 50 and 100 for VRPTW50 and VRPTW100 respectively. The resulting figures describing the reward, mean violations, tour length and λ value progression can be found in Sections 5.2.1, 5.2.2, 5.2.3 and 5.2.4 respectively. These figures were generated from a single training run.

Testing The settings for generating the uniform locations of the nodes is identical during testing as during training, which is already provided in the previous paragraph. We use $k = 5$ for the number of clusters and standard deviation $\sigma = 10.0$ for generating the clustered locations of the nodes. The mixed generation will be a combination of both uniform and clustered generation, specifically half of the locations will be uniformly generated and the other half will be generated using clustered generation. The same problem instance settings used for training are applied here for testing.

5.2 Training

Plotting the metrics of all 10 models in a single graph makes both the graph and the underlying data difficult to read, so the models have been split into three groups instead: soft constraint with a set penalty value, soft constraint models with a decaying penalty value and the models using Lagrangian relaxation. The hard constraint model is included in all three groups, as it serves as a reference point. Furthermore, four of the VRPTW100 models could not reach the 100th epoch in the figures due to insufficient memory. These four models are the hard constraint model, the Lagrangian relaxation model with $\alpha = 0.1$ and the soft constraint models with penalty values λ of 10 and 50, these models reached at least epoch 85.

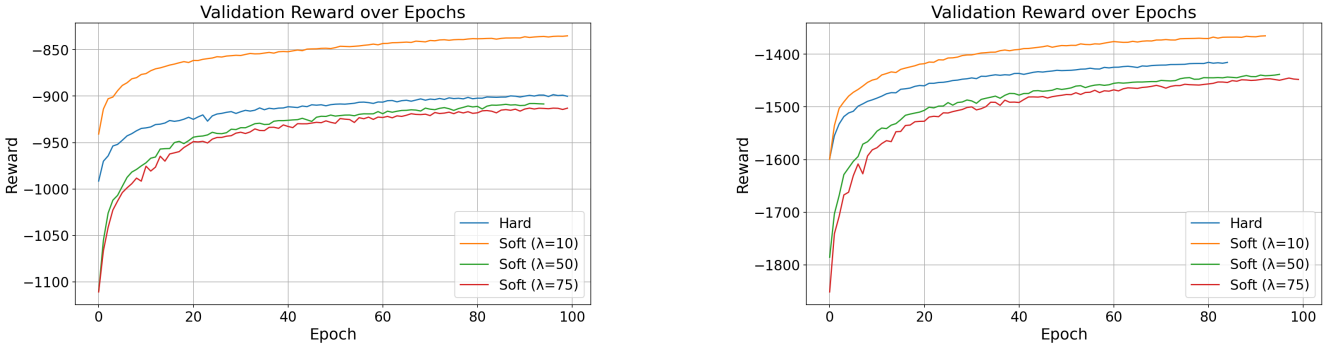
5.2.1 Reward

In Figures 3, 4 and 5 we can find the calculated validation reward during training. Note that the reward is negative and we seek to maximize it. These figures do not account for violations and only for their calculated rewards.

Interestingly, we find that the decaying soft models approximately end at the same final reward value, though first diverging during training, most notably in Figure 4b. This suggests that the final penalty value has a greater influence on the resulting model than the specific decay schedule used to reach the final model, indicating that the learned model parameters do not differ substantially across schedules.

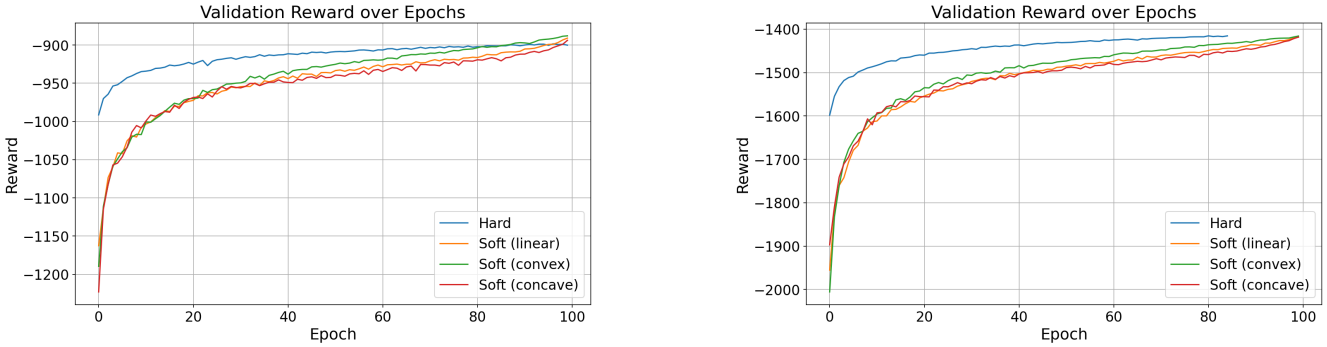
In Figures 3, 4 we see that the models improve their reward over the epochs, except for the Lagrangian Relaxation models with $\lambda_0 = 0$ in Figure 5. This is due to their penalty value starting at 0 and iteratively increasing, meaning that while violating the Time Windows constraint often in the early stages, the penalty is still small. Seemingly, their reward will then worsen over time as the penalty value increases.

If we only consider Figures 3, 4 and 5 and look for the strongest rewards without further context, then the soft model with a penalty value of 10 would be the best performing, but these figures do not account for the number of time windows violations.



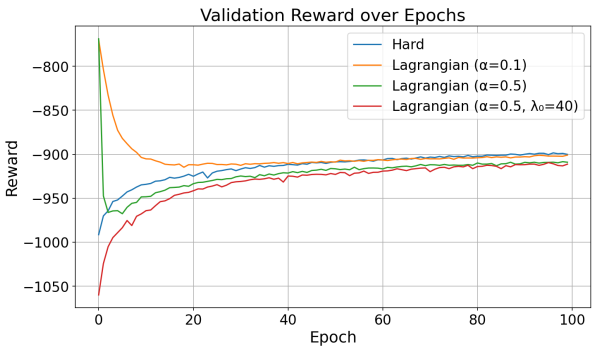
(a) VRPTW50 models (b) VRPTW100

Figure 3: Reward value over 100 epochs trained on VRPTW50 or VRPTW100 of hard constraint and soft constraint models with set λ (10, 50 and 75).

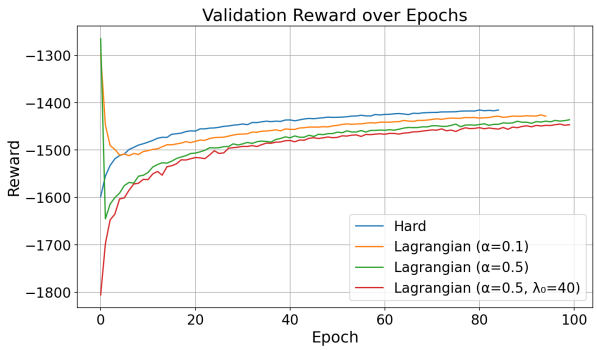


(a) VRPTW50 models (b) VRPTW100 models

Figure 4: Reward value over 100 epochs trained on VRPTW50 or VRPTW100 of hard constraint and soft constraint models with decaying penalty values (linear, convex and concave).



(a) VRPTW50 models



(b) VRPTW100 models

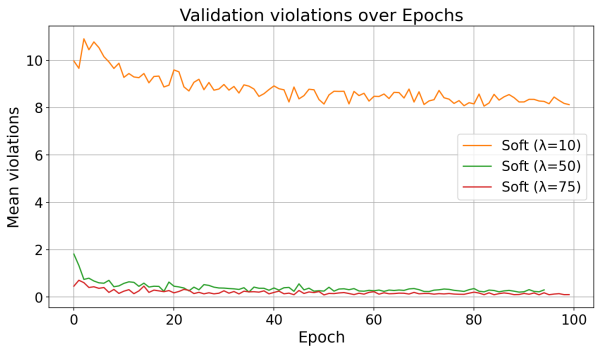
Figure 5: Reward value over 100 epochs trained on VRPTW50 or VRPTW100 of hard constraint and soft constraint models using Lagrangian relaxation with varying λ learning rates (0.1, 0.5) and initial λ values (0, 40)

5.2.2 Mean violations

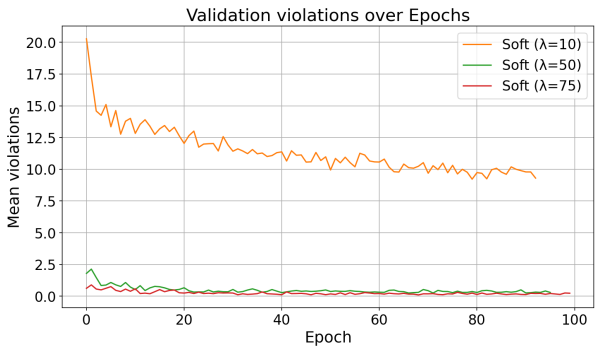
In Figures 6, 7 and 8 we can find the average violations of the models during training over 100 epochs, the hard model is not included as it always has a mean violation of 0. The y-axis of Figure 8 is shown on a logarithmic scale. In Figure 6 we can see that the penalty value of 10 has a high number of violations. When we compare the soft models $\lambda = 10$ in Figures 6a and 6a, we can see that its trajectory starts with a high violations count in the VRPTW100 training. This is probably due to the inherent greater number of nodes that the VRPTW100 has compared to the VRPTW50, this means that the number of violations can be greater. The soft models with λ values of 50 and 75 approach but never actually reach 0, even though the soft model with a penalty value of 75 has a much higher penalty value.

In Figure 7 we can find that the soft decay models start with a low mean violations, this is due to their high initial penalty value, the mean violations never reaches 0 however. As the epochs progresses, the mean violations also increase and actually hit a peak at the latests epochs. This is probably due to their decreasing penalty value and in such 'allowing' them to make more violations, as the penalty would be less for every violation. In both Figure 7a and Figure 7b we can see that the convex decaying soft model has the most number of violations, most notably at the higher epochs, and the concave decaying soft model the least number of violations. This could tell us that the concave approach is simply more effective or that the concave approach is more effective because the penalty value is always greater then the other two decaying methods.

In Figure 8 we can see that all three models approach 0, but that the learning rate $\alpha = 0.5$ models approach it more effectively. The Lagrangian model with a starting $\lambda_0 = 40$ has a 'head start' on the other Lagrangian models in terms of mean violations, however it does not seem to actually capitalize on this head start as the other Lagrangian model with $\lambda_0 = 0$ quickly gains the same results, most notably in Figure 8b, again because of the higher number of nodes this means that the λ_k value can increase more quickly as there is a higher number of violations.

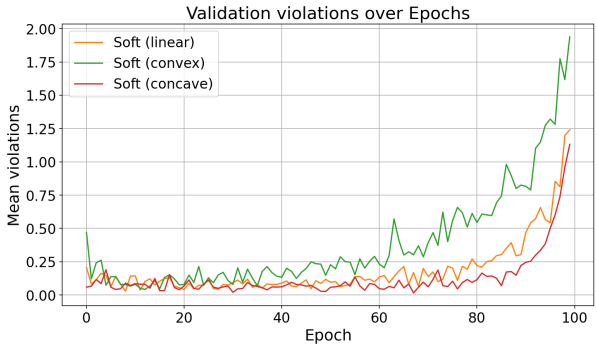


(a) VRPTW50 models

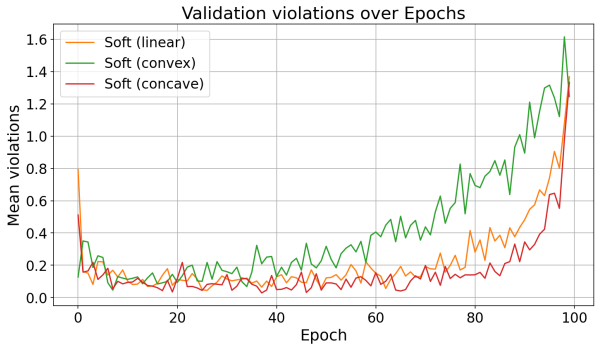


(b) VRPTW100 models

Figure 6: Mean violations over 100 epochs trained on VRPTW50 or VRPTW100 of soft constraint models with set λ (10, 50 and 75).

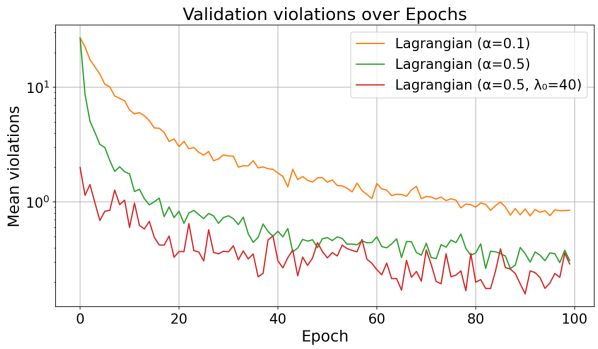


(a) VRPTW50 models

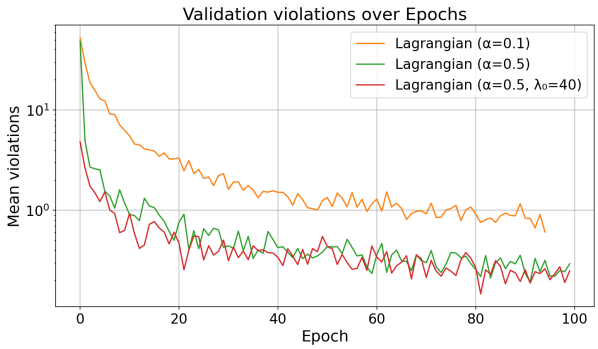


(b) VRPTW100 models

Figure 7: Mean violations over 100 epochs trained on VRPTW50 or VRPTW100 of soft constraint models with decaying penalty values (linear, convex and concave).



(a) VRPTW50 models



(b) VRPTW100 models

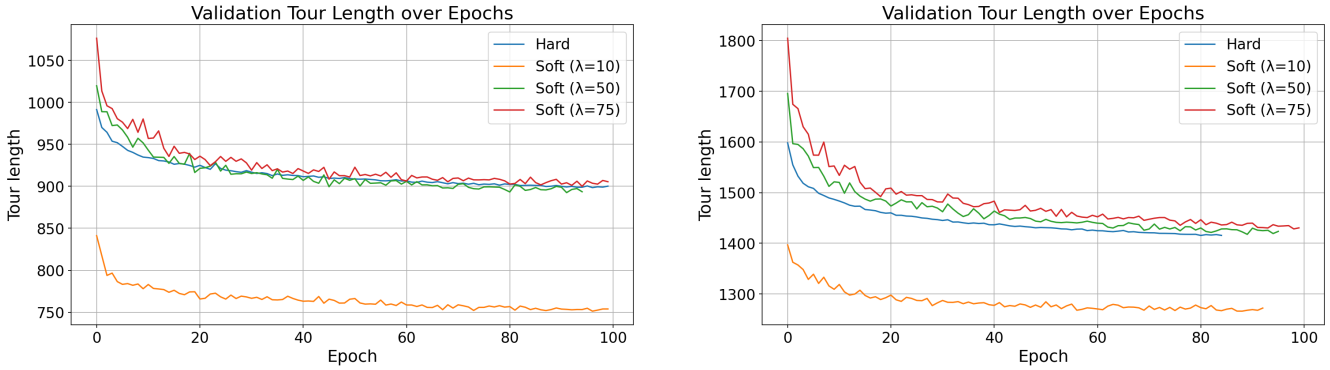
Figure 8: Mean violations over 100 epochs trained on VRPTW50 or VRPTW100 of soft constraint models using Lagrangian relaxation with varying λ learning rates α (0.1, 0.5) and initial λ values λ_0 (0, 40). The y-axis is shown on a logarithmic scale.

5.2.3 Tour length

In Figures 9, 10 and 11 we can find the achieved tour length of the models during training. The hard constraint model is used as a reference and comparison between the graphs.
In Figure 9 we can find that the soft model with $\lambda = 10$ performs the best according to the tour length, however this is misleading as the model has a high number of violations as we saw in Figure 6. We can see that the soft constraint models with a set penalty λ of 50 and 75 perform very similar to the hard constraint model, most notably on the VRPTW50 in Figure 9a.

In Figure 10 we can see that the soft models with a decaying penalty value, have a final tour length shorter than the hard model's tour length. However this is because the penalty value is very low for these models in the latest stages and have an increased number of violations which we saw in Figure 7. We see that the decaying soft models perform very similar to each other, but a certain ranking can be discerned, this is probably due to their differing penalty values at every epoch.

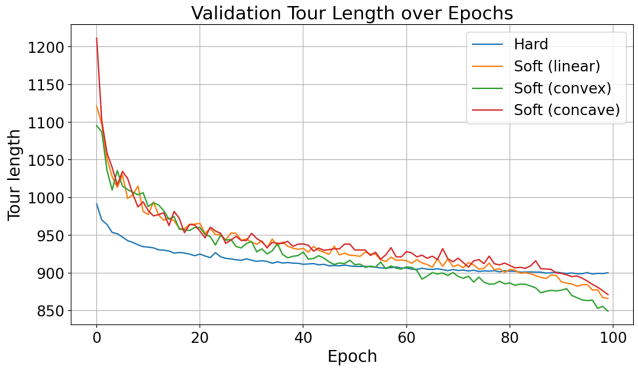
We can see that the Lagrangian models with a learning rate of 0.5 match the hard model very closely in Figure 11, most notably in Figure 11a the Lagrangian models are performing even better then the hard model in terms of the tour length. A starting value for λ of 40 did not ultimately help the Lagrangian model for the number of violations as discussed in Section 5.2.2, and additionally it seems to perform worse than setting the λ value to 0 when we compare the tour length in Figure 11.



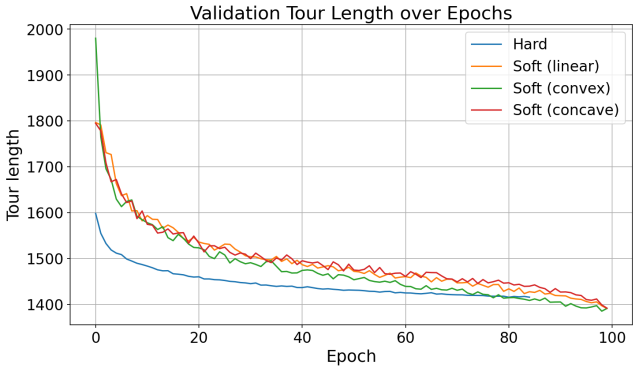
(a) VRPTW50 models

(b) VRPTW100 models

Figure 9: Tour length over 100 epochs trained on VRPTW50 or VRPTW100 of soft constraint models with set λ (10, 50 and 75).

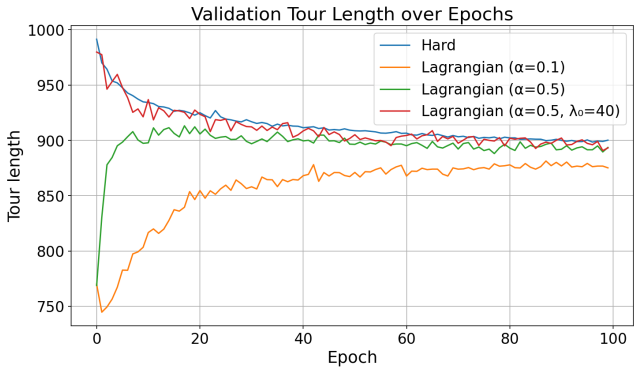


(a) VRPTW50 models

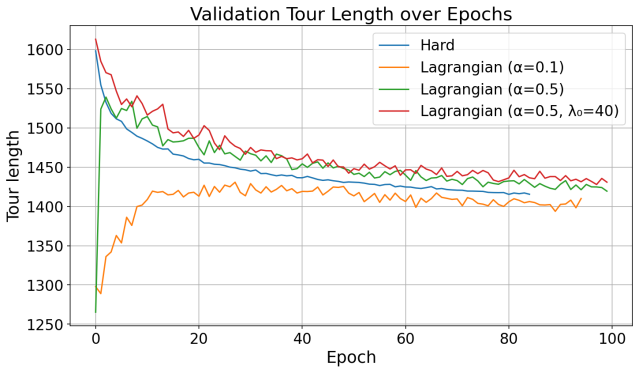


(b) VRPTW100 models

Figure 10: Tour length over 100 epochs trained on VRPTW50 or VRPTW100 of soft constraint models with decaying penalty values (linear, convex and concave).



(a) VRPTW50 models

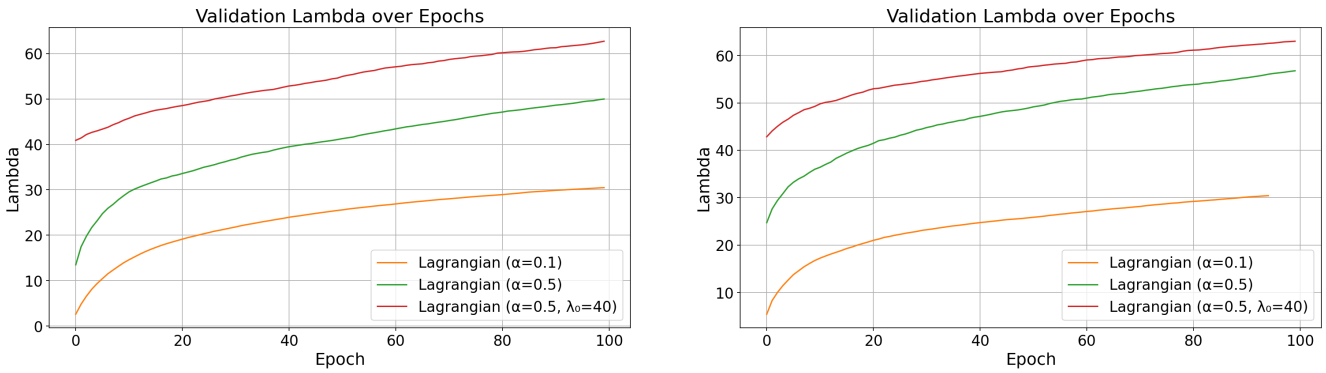


(b) VRPTW100 models

Figure 11: Tour length over 100 epochs trained on VRPTW50 or VRPTW100 of soft constraint models using Lagrangian relaxation with varying λ learning rates α (0.1, 0.5) and initial λ values λ_0 (0, 40).

5.2.4 λ value progression

Figure 12 gives the value of λ_k over 100 training epochs of the soft models that uses Lagrangian relaxation. Remarkably, all the Lagrangian models are steadily increasing, none of the models have actually converged to a final λ_k value, even the Lagrangian model with $\lambda_0 = 40$. Even though λ_k keeps increasing, the tour length is decreasing and performing better in Figure 11, but the mean violations seem to have plateaued in Figure 8. Interestingly, the magnitude of the learning rate α can be discerned from the angle of the slopes.



(a) VRPTW50 models

(b) VRPTW100 models

Figure 12: λ value over 100 epochs, trained on VRPTW50 or VRPTW100, of soft constraint models using Lagrangian relaxation with varying λ learning rates α (0.1, 0.5) and initial λ values λ_0 (0, 40).

5.3 Testing

5.3.1 Uniform generated VRPTW

Table 1 presents the performance of all the trained models on VRPTW50 and VRPTW100 respectively, where the locations of the nodes are generated uniformly, note that this is also how the locations were generated during training. The soft constraint models that use Lagrangian relaxation are denoted as $\mathcal{L}$ with their distinguishing features. Interestingly, the soft model with a set penalty of 75 performs very similar to the soft model with a set penalty value of 50 on the VRPTW50, the difference of the penalty values is apparently less visible on fewer number of locations as the difference increases on the VRPTW100. Finally, for the soft constraint models, tour length, number of routes, and number of violations exhibit a clear trade-off: improving feasibility consistently comes at the cost of solution quality, and vice versa.

5.3.2 Clustered generated VRPTW

Table 2 shows the performance of all the trained models on VRPTW50 and VRPTW100 respectively, where the locations of the nodes are generated in clusters, furher details on how the locations are generated can be found in Section 5. We can see the same trade-off which we saw in previous Section 5.3.1. Simply increasing the penalty value does not simply improve the number of violations, as we can see with the soft constraint models with the penalty values set to 50 and 75 on the VRPTW50, and even if it does improve the number of violations it could come at the expense of other metrics, such as the tour length and number of routes on the VRPTW100. When we compare the results of Table 1 and Table 2, we can see that the average tour length is much lower, this is also true for the average number of violations and routes but in some cases less notable. This is expected, as the locations of the nodes are more grouped together and the distance between these nodes is less compared to the uniformly generated locations.

	Uniform VRPTW50			Uniform VRPTW100		
	Length	Viol.	Routes	Length	Viol.	Routes
Hard	843.73	0	8.70	1346.46	0	14.23
$\text{Soft}_{\lambda=10}$	735.91	7.06	7.38	1235.78	7.72	12.99
$\text{Soft}_{\lambda=50}$	845.43	0.08	8.40	1372.31	0.05	14.66
$\text{Soft}_{\lambda=75}$	854.66	0.01	8.76	1680.17	0.23	18.27
$\text{Soft}_{\text{Linear}}$	819.13	0.97	8.20	1337.14	0.93	14.07
$\text{Soft}_{\text{Convex}}$	804.59	1.42	8.15	1339.09	0.67	14.03
$\text{Soft}_{\text{Concave}}$	824.90	0.74	8.34	1340.74	0.72	14.05
$\mathcal{L}_{\alpha=0.1}$	831.51	0.43	8.41	1353.59	0.24	14.31
$\mathcal{L}_{\alpha=0.5}$	847.28	0.08	8.66	1363.36	0.05	14.43
$\mathcal{L}_{\alpha=0.5,\lambda_0=40}$	844.76	0.05	8.67	1372.92	0.02	14.41

Table 1: Test results of the trained models on 100 problem instances of the VRPTW50 and VRPTW100 respectively with the locations of the nodes generated uniformly, containing the mean tour length, mean number of violations and mean number of routes needed.

	Clustered VRPTW50			Clustered VRPTW100		
	Length	Viol.	Routes	Length	Viol.	Routes
Hard	682.42	0	8.06	1131.06	0	14.49
$\text{Soft}_{\lambda=10}$	623.32	5.19	7.44	1062.13	5.99	13.50
$\text{Soft}_{\lambda=50}$	689.25	0.02	8.11	1129.99	0.11	14.16
$\text{Soft}_{\lambda=75}$	693.06	0.04	8.46	1443.77	0.09	19.36
$\text{Soft}_{\text{Linear}}$	676.58	0.39	8.01	1130.91	0.60	14.55
$\text{Soft}_{\text{Convex}}$	677.27	0.65	8.00	1343.68	0.83	14.98
$\text{Soft}_{\text{Concave}}$	677.32	0.46	8.19	1112.32	0.50	13.71
$\mathcal{L}_{\alpha=0.1}$	681.86	0.22	8.30	1144.54	0.30	14.90
$\mathcal{L}_{\alpha=0.5}$	692.35	0.02	8.33	1136.65	0.32	14.27
$\mathcal{L}_{\alpha=0.5,\lambda_0=40}$	690.31	0.00	8.27	1151.77	0.31	14.56

Table 2: Test results of the trained models on 100 problem instances of the VRPTW50 and VRPTW100 respectively with the locations of the nodes generated in 5 clusters, containing the mean tour length, mean number of violations and mean number of routes needed.

5.3.3 Mixed generated VRPTW

	Mixed VRPTW50			Mixed VRPTW100		
	Length	Viol.	Routes	Length	Viol.	Routes
Hard	834.54	0	8.52	1341.85	0	14.13
$\text{Soft}_{\lambda=10}$	733.54	6.73	7.35	1232.24	7.19	12.92
$\text{Soft}_{\lambda=50}$	837.92	0.06	8.48	1362.43	0.05	14.60
$\text{Soft}_{\lambda=75}$	844.09	0.02	8.73	1684.50	0.15	18.45
$\text{Soft}_{\text{Linear}}$	815.85	0.70	8.24	1334.81	0.76	14.12
$\text{Soft}_{\text{Convex}}$	798.20	1.39	8.01	1330.98	0.81	14.02
$\text{Soft}_{\text{Concave}}$	814.51	0.79	8.30	1332.07	0.77	13.96
$\mathcal{L}_{\alpha=0.1}$	825.50	0.30	8.47	1346.18	0.25	14.29
$\mathcal{L}_{\alpha=0.5}$	836.25	0.04	8.57	1359.26	0.05	14.42
$\mathcal{L}_{\alpha=0.5,\lambda_0=40}$	836.49	0.03	8.61	1366.21	0.04	14.55

Table 3: Test results of the trained models on 100 problem instances of the VRPTW50 and VRPTW100 respectively with the locations of the nodes combining clustered and uniform generation, containing the mean tour length, mean number of violations and mean number of routes needed.

The results of the final type of problem instance can be found in Table 3, the table contains the performance of all the trained models on VRPTW50 and VRPTW100 respectively, the locations of the nodes were generated with a combination of clustered and uniform generation, further details can be found in Section 5. The results in Tale 3 are very similar to the results in Table 2, with the results in Table 3 slightly lower, the probable cause is that half of the locations is generated in clusters, this decreases the distance between these nodes and lowers the final metrics.

5.3.4 Generalization

The results of the performance of the modes that were trained on the VRPTW50 and tested on the VRPTW100 with uniformly generated locations, can be found in Table 4. Note that the same problem instances were used for this test as were used in Table 5.3.1 and can be used for comparison. We can observe that all of the models struggle more than their VRPTW100 counterparts, except for the soft constraint model with a penalty value of 75 which seems to be performing better on a higher number of nodes than it was trained on. Furthermore, the hard constraint model seem to be quite robust and performs similar to its VRPTW100 version.

	VRPTW100		
	Length	Viol.	Routes
Hard	1405.10	0	14.76
$\text{Soft}_{\lambda=10}$	1261.84	13.23	12.66
$\text{Soft}_{\lambda=50}$	1441.97	0.29	16.53
$\text{Soft}_{\lambda=75}$	1463.64	0.03	16.52
$\text{Soft}_{\text{Linear}}$	1382.29	2.52	13.70
$\text{Soft}_{\text{Convex}}$	1369.29	3.17	13.57
$\text{Soft}_{\text{Concave}}$	1383.68	2.36	13.94
$\mathcal{L}_{\alpha=0.1}$	1390.88	1.76	14.30
$\mathcal{L}_{\alpha=0.5}$	1419.66	0.34	14.45
$\mathcal{L}_{\alpha=0.5,\lambda_0=40}$	1454.25	0.26	15.96

Table 4: Test results of the trained VRPTW50 models on 100 problem instances of the VRPTW100, the locations of the nodes are uniformly generated, containing the mean tour length, mean number of violations and mean number of routes needed.

6 Conclusions

From this bachelor thesis we conclude that the soft constraint method cannot guarantee feasible solutions and requires careful tuning to perform well, for our current methods. Future work should therefore explore more advanced constraint-handling techniques. We found that the hard constraint method is an effective mechanism for adapting POMO to the VRPTW. It inherently gives feasible solutions and does not suffer from the feasibility and solution quality trade-off as the soft constraint models have shown, it generalizes well on higher number of nodes and additionally is the easiest to implement as it needs no tuning or setting hyperparameters. If not entirely feasible solutions are desired or limitations such as limited number of available vehicles, as can happen in real world applications, then POMO with soft constraint incorporated may be better, as it has the ability to be tuned and to trade feasibility for better solution quality. However, the settings still have to be tuned, which can prove difficult and time-consuming. Setting the penalty value for the soft constraint can be quite tedious, as it is hard to find the optimal penalty value, and can have unexpected and undesirable behaviour on problem instances on which it has not been trained. When using the soft constraint with a decaying penalty value, the concave profile performed the best, so the aim is to keep the penalty value as high as possible for as long as possible before decreasing the penalty value to the desired value, this has shown to yield the best test results in our research. Lagrangian relaxation would be the best option if one would want to implement the soft constraint and not want to tune and test the model as much, the model just has to be trained long enough for it to settle. A low initial λ value would be best, as the training metrics of a high λ_0 value were not better and even slightly worse. Training the model for longer would allow the model to reach the same ultimate λ value. Ultimately, the choice of the constraint mechanism should be guided by the requirements. If feasibility is most important, then the hard constraint would be the best option, but if some form of flexibility is needed, soft constraint and Lagrangian relaxation

could provide a good alternative, even if it would be more demanding.

7 Discussion

The success and robustness of the hard constraint is probably due to its simplicity. It was initially thought that the soft constraint may have an advantage, due to its freedom of choice during its training and may provide better solutions, but this proved to be untrue. Altering the hyperparameters proved to be more complex than initially thought and produced several unexpected outcomes. The settings for generating the nodes during training and testing were inspired by the Solomon benchmarks [11], but may need to be re-evaluated as the coordinates ranged from 0.0 to 80.0 but the depot location was always at (35, 35), which is not the absolute centre coordinate space. This may have induced some variance and bias which may have influenced the data and results. Training the models proved difficult sometimes due to hardware limitations. Four models that were training on the VRPTW100 couldn't reach the 100th epoch, as a `bus error` was thrown due to insufficient memory, this happened around epoch 85-95. Fortunately, the weights of the model were saved at every epoch, so no progress was lost. We do not expect the last epochs to have a major impact, but as we never actually reached those stages we do not know for certain. We can estimate the trajectory that it would probably have taken looking at the earlier results. In the original POMO paper [9] it was said that 200 epochs for training would suffice, which we were not able to execute due to our computational limitations.

8 Further Research

When calculating the reward of the soft and Lagrangian models, Equations 9 and 14, we only multiply the penalty value against the number of violations, this does not emphasize how 'late' a vehicle arrives at a node, this could especially be important on bigger node counts. For further work, $1[a_{\pi_t} > l_{\pi_t}]$ could be replaced with $|a_{\pi_t} - l_{\pi_t}|$ in Equation 9 and 14 so that the lateness of a vehicle is also accounted for. The number of epochs could be extended to 200 with increased computational power, as the original POMO paper suggested [9], this may help the Lagrangian models converge to a final λ value. Further research could explore more pronounced convex and concave decay profiles, in particular the concave shape, as it has yielded the strongest results in this work. The resulting penalty value for the decaying penalty models was set to 20, but we do not actually have a soft constraint model that was trained with a set penalty value of 20. So, we cannot determine if decaying the penalty value is more effective than setting it to the final penalty value. Overall, there remains room for further research into the mechanisms and behaviour of the soft constraint methods for neural combinatorial optimization.

References

[1] Abdelhakim Abdellaoui, Issmail El Hallaoui, and Loubna Benabbou. Neural combinatorial optimization with reinforcement learning : Solving thevehicle routing problem with time windows, 2022.

[2] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[3] Federico Berto, Chuanbo Hua, Junyoung Park, Laurin Luttmann, Yining Ma, Fanchen Bu, Jiarui Wang, Haoran Ye, Minsu Kim, Sanghyeok Choi, Nayeli Gast Zepeda, André Hottung, Jianan Zhou, Jieyi Bi, Yu Hu, Fei Liu, Hyeonah Kim, Jiwoo Son, Haeyeon Kim, Davide Angioni, Wouter Kool, Zhiguang Cao, Jie Zhang, Kijung Shin, Cathy Wu, Sungsoo Ahn, Guojie Song, Changhyun Kwon, Lin Xie, and Jinkyoo Park. RL4CO: an Extensive Reinforcement Learning for Combinatorial Optimization Benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

[4] Jieyi Bi, Yining Ma, Jianan Zhou, Wen Song, Zhiguang Cao, Yaoxin Wu, and Jie Zhang. Learning to handle complex constraints for vehicle routing problems. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 93479–93509. Curran Associates, Inc., 2024.

[5] Szymon Jakubicz, Karol Kuźniak, Jan Wawszczak, and Paweł Gora. Pomo+: Leveraging starting nodes in pomo for solving capacitated vehicle routing problem, 2025.

[6] Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 1936–1949. Curran Associates, Inc., 2022.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[8] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems!, 2019.

[9] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning, 2021.

[10] Mingjun Pan, Guanquan Lin, You-Wei Luo, Bin Zhu, Zhien Dai, Lijun Sun, and Chun Yuan. Preference optimization for combinatorial optimization problems, 2025.

[11] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(1):254–265, 1987.

[12] Qiaoyue Tang, Yangzhe Kong, Lemeng Pan, and Choonmeng Lee. Learning to solve soft-constrained vehicle routing problems with lagrangian relaxation, 2022.

[13] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[14] Chaoyang Wang, Pengzhi Cheng, Jingze Li, and Weiwei Sun. Leader reward for pomo-based neural combinatorial optimization, 2024.

[15] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.

[16] Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Chi Xu. Mvmoe: Multi-task vehicle routing solver with mixture-of-experts, 2024.