APPENDIX

A ADDITIONAL DETAILS ABOUT METHODOLOGY

A.1 LRP CONFIGURATION

Assumptions: Following the established assumptions Belenguer et al. (2011): (1) Each customer's demand must be served by a delivery from exactly one depot and load transfers at intermediate locations are not allowed; (2) Each customer must be served exactly once by one vehicle, i.e., splitting order is not allowed; (3) No limits on the number of vehicles utilized, but the vehicle cost should be minimized as part of the objective.

Constraints: The constraints in LRP includes three aspects. (1) Customer Demand: The vehicle's remaining capacity must suffice to cover its next target customer's demand during service; (2) Vehicle Capacity: The cumulative demands delivered in a single vehicle route cannot surpass the vehicle's maximum capacity; (3) Depot Supply: The aggregate demands dispatched from a specific depot is expected not to exceed its desired maximum supply.

Remark 1: *The first two items are hard constraints determining solution feasibility, whereas the last item is a soft constraint manifesting as a penalty in the objective function.*

A.2 MDP FORMULATION

Here, we propose the formulation of feasible LRP solution routes in form of MDP, which is an entire permutation of the vertices in the graph. As depicted in Fig. 3, the routes corresponding to the same depot have the identical start and end point, facilitating their aggregation into an entire permutation by jointing their identical depot. Consequently, by linking together these permutations from all depots, a feasible solution can be finally formulated as an MDP.

Remark 2: *The MDP is a necessary mathematical formulation used to construct the feasible solution routes when engaging DRL method. Once the solution is derived in MDP form, it will be reverted to a set of routes for simultaneous execution by multiple vehicles.*

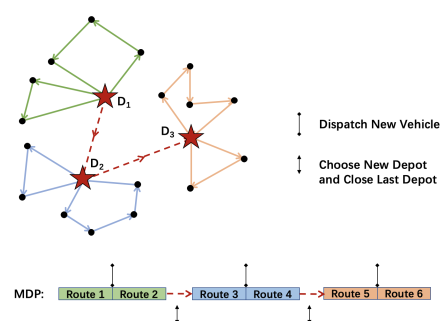

Figure 3: The feasible LRP solution in this example consists of 6 single routes, which are simultaneously carried out by multiple vehicles. The routes in same color belongs to a same depot. By linking them together, the feasible solution is formulated in points permutation, as an MDP.

We define this MDP with a tuple $(\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{R}, \gamma)$, where, in each decision step t, the current iteration is represented by a tuple $(s_t, a_t, p_t, r_t, \gamma_t)$.

(a) $\mathbf{S}$: is a set of states, wherein each state corresponds to a tuple $(G, D_t, \mathbf{v}_t, Q_t)$, where G denotes entire static graph information; D_t indicates the depot which current route belongs to; $\mathbf{v}_t$ signifies current customer in decision step t; Q_t records remaining capacity on current vehicle; This tuple is updated at each decision step within MDP.

(b) $\mathbf{A}$: is a set of actions, wherein each action a_t is the next point that current vehicle plans to serve. In this problem configuration, to ensure that the MDP represents a feasible solution, actions should be selected from feasible points whose demands can be satisfied by current vehicle's remaining capacity. Upon selecting the a_t, the state tuple should be updated accordingly:

$$Q_{t+1} = \begin{cases} Q_t - q_e & \text{if } a_t \in \{\mathbf{v}_{S_e} | e = 1, 2, \ldots, n\}, \\ Q & \text{if } a_t \in \{\mathbf{v}_{D_k} | k = 1, 2, \ldots, m\}, \end{cases} \tag{9}$$

$a_t \in \{\mathbf{v}_{S_e} | e = 1, 2, \ldots, n\}$ indicates that current vehicle is scheduled to visit an unserved customer. Then, the remaining capacity Q_t should be updated according to Eq. (9), wherein q_e represents the demand associated with the customer selected by action a_t. Meanwhile, $a_t \in \{\mathbf{v}_{D_k} | k =$

$1, 2, \ldots, m\}$ indicates that current vehicle chooses to return to its departure depot, or start planning for a new depot. Then, a new vehicle's route will commence from this depot, thereby the capacity Q_t is refreshed to full state.

(c) $\mathbf{P}$: is a set of probabilities, wherein each element p_t represents the probability transiting from state s_t to s_{t+1} by taking action a_t, and p_t can be expressed as: $p_t = p(s_{t+1}|s_t, a_t)$

(d) $\mathbf{R}$: is a set of costs, wherein each element r_t denotes the cost incurred by taking action a_t in step t. The r_t can be expressed as follows, where d_{ij} denotes the length between $\mathbf{v}_i$ in step t and $\mathbf{v}_j$ in step $t+1$:

$$r_t = \begin{cases} 0 & \text{if } \mathbf{v}_i, \mathbf{v}_j \in \{\mathbf{v}_{D_k} | k = 1, 2, \ldots, m\}, \\ d_{ij} & \text{otherwise}, \end{cases} \tag{10}$$

As is shown in Eq. (1), apart from this step-wisely accumulated transit distance, other costs used to depict the overall performance of the solution routes, which are not accumulated step-wisely, are added into the total cost after an entire MDP is generated. These additional overall costs include: *(i) the opening cost for used depots; (ii) the setup cost for dispatched vehicles; (iii) penalty of exceeding depot desired maximum supply*.

(e) $\gamma \in [0, 1]$: the discount factor for cost in each step. Here, we presume no discount applies to the costs, i.e., $\gamma = 1$

A.3 Multi-depot mask mechanism

In each decoding step, guided by the context embedding $\mathbf{h}_c^t$, the decoder produce the corresponding probabilities for all the feasible points within the selection domain. This selection domain should exclude all the points that current vehicle cannot visit in next step, which is subject to vehicle capacity and current state in MDP. Because the model processes problem instances in batches, simultaneous updates to their respective selection domains at each decoding iteration is necessary.

We identify four key scenarios to categorize the selection domain of each instance at any given step, based on the vehicle's location (depot or customer) and the completion status of delivery tasks. Specifically, these four potential patterns are summarized as follows:

- (i) When current vehicle is at a depot and all the customers' delivery tasks are finished: it can only stay at current depot.
- (ii) When current vehicle is at a depot but not all the customers' delivery tasks are finished: it can choose from the vertices set including all the unserved customers and unplanned depots but excluding current depot.
- (iii) When current vehicle is at a customer and all the customers' delivery tasks are finished: this represents the current customer is the last delivery task, implying that the only selection is the vehicle's departure depot.
- (iv) When current vehicle is at a customer but not all the customers' delivery tasks are finished: it can choose from the vertices set including all the unserved customers and its departure depot.

Based on these four patterns, the selection domain is updated before each decoding iteration.

As discussed, the model operates in batch-wise manner, necessitating simultaneous updating each instance's selection domain at each decoding iteration. **The challenge is**, in each decoding step, the selection domain of each problem instance within one batch can be very different. Thus, an efficient boolean mask matrix specific to the LRP scenario is devised for batch-wise manipulation on selection domain, avoiding repeated operation on individual problem instance.

The Algorithm 1 specifies our mask mechanism specifically tailored for LRP scenario. which includes manipulations on the selection domain of customers and depots. Firstly, by masking the customers which have been served or cannot be satisfied with remaining capacity, the selection domain of customers can be simply derived. Crucially, for the depot selection domain, we notice that among the four patterns above: three patterns (i, iii, and iv) include only the departure depot, whereas one pattern (ii) excludes the departure depot. Thus, at each decoding step for a batch of instances, we initially mask all the depots unanimously and only reveal their departure depot of current routes. Then, we identify the problem instances belonging to pattern-ii in this batch, mask the departure depots and reveal the unplanned depots. All the manipulations operate in batches to avoid repeated operation on individual problem instance.

Algorithm 1 Mask Mechanism for batch-wise manipulation on selection domain for a batch of problem instances

Input: A batch of problem instances with Batch Size B

```
Init Record = [σ_ij] ∈ ℝ^{B×(m+n)} where σ_ij ∈ {0,1} representing, in problem instance i,
    whether the vertex j is visited (σ_ij = 0) or unvisited (σ_ij = 1)
Init ID ∈ ℝ^B current situated vertices for all instances
Init DP ∈ ℝ^B current departure depots for all instances
for each decoding step t = 1, 2, ... do
    {φ_i} ← Batch No. for the problem instances where not all the tasks are finished
    {φ_j} ← Batch No. for the problem instances where all the tasks are finished
    σ_ij ← 0 according to the ID_t
    (Mask_0)_ij ← True if σ_ij = 0, (Mask_0)_ij ← False if σ_ij = 1
    (Mask_1)_ij ← True if (Q_t)_i < (q_e)_j, (Mask_0)_ij ← False if (Q_t)_i > (q_e)_j
    Mask ← Mask_0 + Mask_1
    (Mask)_ij ← True for all j ∈ {0, 1, ..., m − 1}
    (Mask)_ij ← False according to the DP_t
    {φ_k} ← Batch No. for the problem instances where current vertex is one of the depots
    {φ_e} ← {φ_i} ∩ {φ_k} Batch No. for the problem instances where current vertex is one of the
        depots and not all tasks are finished
    (Mask)_ij ← False where i ∈ {φ_e} and j ∈ {0, 1, ..., m − 1}
    (Mask)_ij ← True where i ∈ {φ_e} and DP_{φ_e} ∈ {0, 1, ..., m − 1}
    (Mask)_ij ← True where j ∈ {0, 1, ..., m − 1} and σ_ij = 0
end for
Return Mask
```

A.4 MDLRAM's pre-training & DGM's dual-mode training

Algorithm 2 Pre-training for MDLRAM

Input: M batches of problem instances with Batch Size B

```
for each epoch ep = 1, 2, ..., 100 do
    for each batch bt = 1, 2, ..., M do
        {G_b | b = 1, 2, ..., B} ← A Batch of Cases
        {A_b^{θ_I} | b = 1, 2, ..., B} ← MDLRAM_{θ_I}({G_b})
        {A_b^{θ_I^*} | b = 1, 2, ..., B} ← MDLRAM_{θ_I^*}({G_b})
        ∇L(θ_I) ← (1/B) Σ_{b=1}^{B} [(L(A_b^{θ_I}) − L(A_b^{θ_I^*})) ∇ log p_{θ_I}(A_b^{θ_I})]
        if One Side Paired T-test (A_b^{θ_I}, A_b^{θ_I^*}) < 0.05 then
            θ_I^* ← θ_I
        end if
    end for
end for
```

The baseline $\bar{\mathcal{B}}$ in Algorithm 2 is established through a parallel network mirroring the structure of MDLRAM, persistently preserving the best parameters attained and remaining fixed. Parameters' update solely occurs if a superior evaluation outcome is derived by MDLRAM, enabling baseline network's adoption of these improved parameters from MDLRAM. The actions in MDPs produced by MDLRAM is selected with probabilistic sampling in each decoding step, whereas that of baseline network is greedily selected based on the maximum possibility.

Algorithm 3 Dual-mode training for DGM, coupled with pretrained MDLRAM functioning as a critic model

Input: Batches of problem instances with Batch Size B_{main}

if in Multivariate Gaussian Distribution mode **then**
  for each epoch $ep = 1, 2, ..., 100$ **do**
    for each batch $bt = 1, 2, ..., M$ **do**
      $\{G_b | b = 1, 2, ..., B_{\text{main}}\} \leftarrow$ A Main-Batch of graphs with customers Info
      $\{\mathcal{N}_b^{\theta_{\text{II}}} | b = 1, 2, ..., B_{\text{main}}\} \leftarrow \text{DGM}_{\theta_{\text{II}}}(\{G_b\})$
      for each graph $b = 1, 2, ..., B_{\text{main}}$ **do**
        $\{\mathcal{D}_{\text{multiG}}^{(b')} | b' = 1, 2, ..., B_{\text{sub}}\} \leftarrow$ A Sub-Batch of sampled depot sets
        $\nabla L_{DGM}(\mathcal{N}_b) \leftarrow \mathbb{E}^{(b)}_{p_{\boldsymbol{\theta}_{\text{II}}}(\mathcal{D}_{\text{multiG}})}[\text{MDLRAM}(\mathcal{D}_{\text{multiG}}^{(b')}, G_b)$
        $\cdot \nabla \log p_{\boldsymbol{\theta}_{\text{II}}}(\mathcal{D}_{\text{multiG}}^{(b')})]$
      end for
      $\nabla \mathcal{L}(\boldsymbol{\theta}_{\text{II}}) \leftarrow \frac{1}{B_{\text{main}}} \sum_{b=1}^{B_{\text{main}}} \nabla L_{DGM}(\mathcal{N}_b)$
    end for
  end for
else if in Exact Position mode **then**
  for each epoch $ep = 1, 2, ..., 100$ **do**
    for each batch $bt = 1, 2, ..., M$ **do**
      $\{G_b | b = 1, 2, ..., B_{\text{main}}\} \leftarrow$ A Main-Batch of graphs with customers Info
      $\{\mathcal{D}_{\text{exactP}}^{(b)} | b = 1, 2, ..., B_{\text{main}}\} \leftarrow \text{DGM}_{\theta_{\text{II}}}(\{G_j\})$
      $\nabla \mathcal{L}(\boldsymbol{\theta}_{\text{II}}) \leftarrow \frac{1}{B_{\text{main}}} \sum_{b=1}^{B_{\text{main}}} \nabla \text{MDLRAM}((\mathcal{D}_{\text{exactP}}^{(b)})_{\boldsymbol{\theta}_{\text{II}}}, G_b)$
    end for
  end for
end if

B Extended details about Experimental Results

B.1 Hyperparameters Details

For MDLRAM, we train it for 100 epochs with training problem instances generated on the fly, which can be split into 2500 batches with batchsize of 512 (256 for scale 100 due to device memory limitation). Within each epoch, by going through the training dataset, MDLRAM will be updated 2500 iterations. After every 100 iterations, the MDLRAM will be assessed on an evaluation dataset to check whether improved performance is attained. The evaluation dataset consists of 20 batches of problem instances, with the same batch size of 512(256).

For DGM, we also train it for 100 epochs. In each epoch, 2500 main-batches of problem instances are iteratively fed into DGM. **In multivariate Gaussian distribution mode**, the main-batch size B_{main} is set as 32 (16 for scale 100), and the sub-batch size B_{sub} for sampling in each distribution is selected as 128, 64, 32 for scale $20, 50, 100$ respectively. During training, after every 100 iterations' updating, the DGM will be evaluated on an evaluation dataset to check if a better performance is derived. The evaluation dataset is set as 20 main-batches of problem instances, maintaining the same batch size B_{main} and B_{sub}. **In exact position mode**, where no sampling is performed, we set main-batch size as 512 (256 for scale 100). Likewise, after every 100 iterations' updating, an evaluation process is conduct on 20 main-batches of problem instances with corresponding batch size of 512 (128) to check if DGM achieves a better performance.

As for the hyperparameters in model architecture across the entire framework, the encoding process employs $N = 3$ attention modules with 8-head MHA sublayers, featuring an embedding size of 128. All the training sessions are finished on one single A40 GPU.

Parameters for heuristic methods in Table 1: **(a)** Adaptive Large Neighborhood Search (ALNS): *Destroy (random percentage* $0.1 \sim 0.4$*, worst nodes* $5 \sim 10$*); Repair (random, greedy, regret with 5 nodes); Rewards (*$r_1 = 30, r_2 = 20, r_3 = 10, r_4 = -10$*); Operators weight decay rate:* 0.4*; Threshold decay rate:* 0.9; **(b)** Genetic Algorithm (GA): *Population size: 100; Mutation probability:*

0.2; Crossover probability: 0.6; **(c)** Tabu Search (TS): *Action Strategy (1-node swap, 2-node swap, Reverse 4 nodes); Tabu step:* 30;

B.2 Visualize Depots Distribution:

DGM's distribution mode is trained to understand correlations between coordinates of various depots, manifested as their learnable covariances. To visualize the distribution generated in the Gaussian mode of DGM and observe how this multivariate Gaussian distribution is represented in a 2-D graph, we depict the generated multivariate Gaussian distribution for problem instances from all three scales. A notable pattern is revealed as below:

In the problem scale of $m = 3, n = 20$, the 6-D normal distribution tends to present as three separate 2-D normal distributions, as depicted in Fig. 4. However, as the problem scales increase, such as the 12-D ($m = 6, n = 50$) or 18-D ($m = 9, n = 100$) normal distributions, they do not tend to present as several discrete 2-D normal distributions.

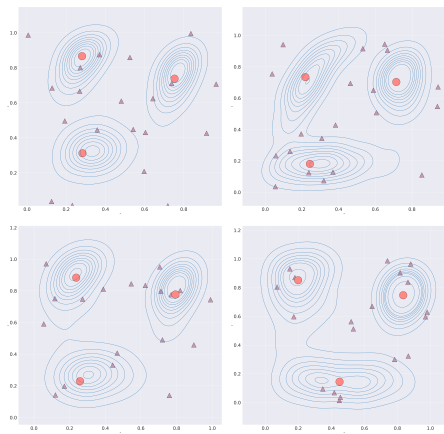

Figure 4: Visualization of Multivariate Gaussian Distribution outputted by DGM based on customer requests (Gray): Predicted Depot Distribution (Blue), and Optimal Depots Identified (Red).

This trend indicates that, in large-scale scenario, the covariance between coordinates from different depots exhibit a more complex relationship, which further implies that simply relying on randomly sampling depots in pursuit of covering optimal depots would require an expansive search and substantial computational effort.

B.3 MDLRAM's ability on balancing route length among depots

With MDLRAM's structure, fine-tuning the model to align with diverse additional requirements associated to the multiple depots in LRP scenario is flexible through designing specialized cost functions. Here, we examine the route balancing challenge among various depots.

If the objective is to maintain the route length $l_k(\mathbf{A})$ associated with each depot D_k ($k \in \{1, 2, \ldots, m\}$) in a specific proportional relationship, namely $l_1(\mathbf{A}) : l_2(\mathbf{A}) : \ldots : l_m(\mathbf{A}) = \rho_1 : \rho_2 : \ldots : \rho_m$, while simultaneously minimizing the overall cost $L_{\text{Sel}}(\mathbf{A})$, it can be achieved by augmenting the cost function $L_{\text{Sel}}(\mathbf{A})$ in Eq. (1) with a balance penalty as follows:

$$\tilde{L}_{\text{Sel}}(\mathbf{A}) = L_{\text{Sel}}(\mathbf{A}) + \sum_{k=1}^{m} \sum_{k'=k}^{m} |l_k(\mathbf{A}) - \frac{\rho_k}{\rho_{k'}} l_{k'}(\mathbf{A})| \tag{11}$$

To evaluate the adaptability of MDLRAM in addressing LRP with additional requirements on adjusting inter-depot cost distribution, we fine-tune the MDLRAM, which has been pre-trained with original objective $L_{\text{Sel}}(\mathbf{A})$ in Eq. (1), with this new balance-oriented objective $\tilde{L}_{\text{Sel}}(\mathbf{A})$ in Eq. (11) on the same training dataset. In this context, our specific goal is to ensure that the lengths belonging to each depot are approximately equal (i.e., $\rho_k = 1$). Notably, ρ_k can be adjusted based on specific proportion requirements.

To illustrate the effectiveness of balance-oriented fine-tuning, we select random cases from each scale for direct comparison of route length belonging to each depot, generated by MDLRAM under different objectives. In Table 4, it can be observed that, for each case, the balance penalty of solution routes found by MDLRAM under balance-oriented objective Eq. (11) is conspicuously smaller than that of original objective Eq. (1), only incurring a slight wave on the total length as an acceptable trade-off for incorporating the additional item in the balance-oriented objective function. This can also be directly reflected by the balanced route length distribution across depots in 5th column of Table 4.

Table 4: Comparison of Each Depot's Route Length, respectively planned by Original MDLRAM and the Fine-tuned Version. ("Obj.": Objective Function; "Ori.Obj.": Original Objective Function in Eq. (1); "Bln.Obj.": Balance-oriented Objective Function in Eq. (11); "Bln.Pen.": penalty for measuring the balancing performance of route length among depots; "Dpt.Nb.": opened depot number out of total available depots).

	Case	Obj.	**Bln. Pen.**	(Dpt Nb.)	**Saperate Depot Len.**	Total Len.
scale 20	case1	Ori obj.	0.758	2/3	3.487-2.729	6.216
		Bln obj.	**0.008**	2/3	2.781-2.772	5.554
	case2	Ori obj.	0.929	2/3	3.439-2.511	5.951
		Bln obj.	**0.007**	2/3	3.022-3.016	6.038
	case3	Ori obj.	0.926	2/3	3.608-2.682	6.290
		Bln obj.	**0.022**	2/3	3.123-3.102	6.225
	case4	Ori obj.	0.693	2/3	2.853-2.159	5.012
		Bln obj.	**0.0002**	2/3	2.518-2.518	5.036
scale 50	case1	Ori obj.	3.131	4/6	2.158-2.536-2.155-3.073	9.922
		Bln obj.	**0.129**	4/6	2.492-2.530-2.521-2.507	10.052
	case2	Ori obj.	3.738	4/6	2.150-3.154-2.947-2.220	10.471
		Bln obj.	**0.283**	4/6	2.449-2.434-2.473-2.383	9.739
	case3	Ori obj.	2.016	3/6	2.981-2.579-3.586	9.146
		Bln obj.	**0.067**	3/6	3.085-3.091-3.058	9.234
	case4	Ori obj.	2.416	4/6	1.808-2.596-1.918-1.969	8.292
		Bln obj.	**0.176**	4/6	2.190-2.186-2.163-2.220	8.759
scale 100	case1	Ori obj.	3.444	5/9	2.728-3.132-2.496-3.092-2.642	14.091
		Bln obj.	**0.916**	5/9	2.736-2.742-2.829-2.842-2.915	14.063
	case2	Ori obj.	2.495	5/9	3.008-3.344-3.063-3.487-3.353	16.256
		Bln obj.	**0.373**	5/9	3.045-3.015-2.987-2.987-2.967	15.001
	case3	Ori obj.	5.310	5/9	3.743-2.622-2.985-3.335-2.922	15.606
		Bln obj.	**1.641**	5/9	3.043-3.099-3.056-3.249-3.358	15.808
	case4	Ori obj.	8.711	5/9	3.273-3.398-4.455-2.599-2.754	16.479
		Bln obj.	**1.896**	5/9	3.492-3.465-3.404-3.709-3.755	17.825

B.4 Further Discussion

In this study, we extend the exploration of the LRP by addressing a real-world challenge: the generation of depots when no predefined candidates are presented. For this purpose, a generative DRL framework comprising two models is proposed. Specifically, the DGM, based on customer requests data, enables proactive depot generation with dual operational modes flexibly- the exact mode ensures precision when necessary, while the Gaussian mode introduces sampling variability, enhancing the model's generalization and robustness to diverse customer distributions. Meanwhile, the MDLRAM subsequently facilitates rapid planning of LRP routes from the generated depots for serving the customers, minimizing both depot-related and route-related costs. Our framework represents a transition from traditional depot selection to proactive depot generation, showcasing cost reductions and enhanced adaptability in real-world scenarios like disaster relief, which necessitates quick depot establishment and flexible depot adjustment.

The framework's detachability offers flexible extension for its application. The DGM's depot-generating ability can be fine-tuned to adapt different LRP variants by jointing with other downstream models, making DGM a versatile tool in real-world logistics. Meanwhile, the end-to-end nature of MDLRAM enable its flexible usage on addressing LRP variants with requirements of adjusting inter-depot cost distribution, which has been detailed in Appendix B.3.

Based on the framework design details and the application scenario description, we spot following limitatioins and arranging a research landscape for future works.

Limitation: While the MDLRAM model has the ability to select a flexible number of depots from the generated depot set when planning routes for vehicle from the generated depot set, the number of depots generated by the DGM is currently set fixed during training. Incorporating an adaptive mechanism within the DGM to dynamically determine the optimal number of depots based on customer demands and logistical factors could further enhance the framework's flexibility and efficiency. Achieving this adaptive depot generation may require a more conjugated and interactive integration between the DGM and the MDLRAM's route planning process.

Future work: Future research will focus on expanding DGM's applicability by incorporating a wider range of depot constraints to reflect more real-world scenarios accurately. For example, in this study, we consider the distance between depots should adhere to a specific range requirements, preventing the depots from being too close or too distant with each other. Additional constraints on depots can be emphasized on the forbidden area within the map, such as ensuring the depots are not situated in specific regions or must be placed within designated zones.

Additionally, leveraging the framework's modular design to adapt to various routing tasks presents an exciting avenue for exploration. This includes generating depots which can generally achieve satisfying performance across multiple concurrent routing tasks, which would further extend the framework's utility in complex and dynamic real-world logistics environments.