A Data Sets

A.1 Details of Training Samples

To generate training samples for the machine learning model, we first created a set of 2L-VRPTW and 2L-CVRP instances. The 2L-VRPTW were generated as per the following characteristics.

1. **Customer distribution:** This is the distribution from which customer coordinates in $G = (V, E)$ are sampled. We designed three different scenarios: *pure random*, *clustered*, and *mixed*. Specifically, for *pure random*, the depot is fixed at (35,35) while all customer coordinates are uniformly distributed in the range [0,100]. In *clustered*, the depot is at (40,50), each cluster center is sampled from [10,90], and the number of customers per cluster ranges from 8 to 9. Customers within a cluster are randomly placed at a distance from the cluster center, determined by a uniform distribution in the range [3,5]. For *mixed*, 50% of the customers are generated randomly, and the rest are clustered.

2. **Time windows:** Generating time windows involves two steps. First, we determine the distribution for the time window 'diameter' (half of the width). The mean is a fraction of the maximum return time to the depot, and the standard deviation is a product of the maximum return time and a value sampled from [0, 0.05, 0.10, 0.15, 0.20, 0.25]. The center of the time window is then uniformly sampled between the earliest and latest possible times to reach the customer. The time window is obtained by adding and subtracting the diameter from the center.

3. **Rectangular items:** Items are generated based on Table 1. The column *PC* (Packing Class) specifies the packing class for each instance. The column $|M_i|$ indicates the number of items, where M_i is the set of items for customer i. The master column *Vertical* suggests that items should feature $h_{i,m} \geq w_{i,m}$ on average, *Homogeneous* implies square items, and *Horizontal* indicates items should feature $h_{i,m} \leq w_{i,m}$ generally.

The 2L-CVRP instances for training were generated similarly, except that time windows were not needed for customers. To generate the packing problems, we used the branch-and-price algorithm from [Zhang *et al.*, 2022b] with the ML model disabled for solving 2L-VRPTW, and the branch-and-price-and-cut algorithm from [Zhang *et al.*, 2022a] for 2L-CVRP.

PC	$\lvert M_i \rvert$	Vertical		Homogeneous		Horizontal	
		$h_{i,m}$	$w_{i,m}$	$h_{i,m}$	$w_{i,m}$	$h_{i,m}$	$w_{i,m}$
2	[1,2]	[4H/10, 9H/10]	[W/10, 2W/10]	[2H/10, 5W/10]	[2W/10, 5W/10]	[H/10, 2H/10]	[4W/10, 9W/10]
3	[1,3]	[3H/10, 8H/10]	[W/10, 2W/10]	[2H/10, 4H/10]	[2W/10, 4W/10]	[H/10, 2H/10]	[3W/10, 8W/10]
4	[1,4]	[2H/10, 7H/10]	[W/10, 2W/10]	[3H/10, 4H/10]	[W/10, 4W/10]	[H/10, 2H/10]	[2W/10, 7W/10]
5	[1,5]	[H/10, 6H/10]	[W/10, 2W/10]	[H/10, 3H/10]	[W/10, 3W/10]	[H/10, 2H/10]	[W/10, 6W/10]

Table 1: Description of item sizes for each instance family

A.2 Details of Benchmark Instances

The benchmark instances were generated by [Iori *et al.*, 2007] following Table 1 for item numbers and sizes. For customer coordinates, they used existing CVRP instances, generating one instance for each packing class from each CVRP instance. The number of vehicles for each instance was determined by solving a 2D-BPP problem to find the minimum number of bins needed to pack all items in G. The final vehicle count is the maximum of the 2D-BPP solution and the number of vehicles in the original CVRP instance.

B Experimental Results

Instance	SOTA Run Time (s)	NCG Run Time (s)	Performance Increase (%)	PC
2l-cvrp0102-90&3	18.61	18.18	2.38	2
2l-cvrp1902-160&11	79.69	72.73	9.57	2
2l-cvrp1802-2010&9	209.80	272.98	-23.14	2
2l-cvrp1702-60&14	8.03	7.76	3.43	2
2l-cvrp1602-67&11	4.52	3.82	18.11	2
2l-cvrp1402-8000&7	70.26	77.25	-9.05	2
2l-cvrp1302-38000&7	22.38	21.98	1.82	2
2l-cvrp1202-68&9	27.87	26.87	3.73	2
2l-cvrp1102-4500&6	61.24	55.78	9.79	2
2l-cvrp1002-4500&6	36.84	27.62	33.35	2
2l-cvrp0902-48&8	2.61	2.09	25.06	2
2l-cvrp0802-4500&5	9.70	10.00	-2.99	2
2l-cvrp1502-8000&6	135.67	146.51	-7.40	2
2l-cvrp0202-55&5	0.64	0.58	9.98	2
2l-cvrp0602-4000&6	4.65	4.47	4.16	2
2l-cvrp0702-4500&5	14.45	13.21	9.38	2
2l-cvrp0502-6000&4	11.18	8.69	28.65	2
2l-cvrp0302-85&5	3.49	2.69	29.79	2
2l-cvrp0402-58&6	2.75	2.28	20.34	2
2l-cvrp1903-160&11	123.66	106.06	16.60	3
2l-cvrp0103-90&3	45.78	15.34	198.34	3
2l-cvrp1803-2010&10	80.36	87.62	-8.28	3
2l-cvrp1703-60&14	6.56	9.09	-27.87	3
2l-cvrp0203-55&5	1.42	1.10	29.25	3
2l-cvrp1603-67&11	6.07	3.68	64.95	3
2l-cvrp1503-8000&6	344.18	363.51	-5.32	3
2l-cvrp1403-8000&7	80.15	62.15	28.95	3
2l-cvrp0703-4500&5	15.73	9.64	63.13	3
2l-cvrp0603-4000&6	4.90	3.25	50.70	3
2l-cvrp1303-38000&7	33.65	23.15	45.36	3
2l-cvrp1203-68&9	10.49	9.20	14.04	3
2l-cvrp0403-58&6	4.16	2.97	40.16	3
2l-cvrp1103-4500&7	29.95	24.22	23.69	3
2l-cvrp1003-4500&6	47.06	31.31	50.29	3
2l-cvrp0503-6000&4	17.40	9.66	80.10	3
2l-cvrp0903-48&8	8.44	5.82	44.97	3
2l-cvrp0803-4500&5	15.96	9.64	65.66	3
2l-cvrp0303-85&5	8.78	5.29	65.87	3
2l-cvrp1204-68&9	23.39	12.44	88.05	4
2l-cvrp1304-38000&7	151.96	88.45	71.79	4
2l-cvrp1404-8000&7	309.57	136.66	126.53	4
2l-cvrp1804-2010&10	184.45	191.46	-3.66	4
2l-cvrp1604-67&11	24.70	12.40	99.22	4
2l-cvrp1704-60&14	10.65	8.24	29.17	4
2l-cvrp1104-4500&7	141.24	127.72	10.59	4
2l-cvrp1504-8000&8	218.22	162.76	34.07	4
2l-cvrp1004-4500&7	85.22	33.68	153.04	4
2l-cvrp0504-6000&4	34.82	11.02	216.00	4
2l-cvrp0804-4500&5	32.42	9.49	241.83	4
2l-cvrp0704-4500&5	41.94	18.28	129.38	4
2l-cvrp0604-4000&6	17.48	8.56	104.22	4
2l-cvrp0404-58&6	8.32	5.15	61.80	4
2l-cvrp0304-85&5	5.38	3.39	58.64	4
2l-cvrp0204-55&5	3.32	1.01	227.34	4
2l-cvrp0104-90&4	7.94	2.70	194.15	4
2l-cvrp0904-48&8	9.33	5.08	83.80	4
2l-cvrp1904-160&12	142.75	69.12	106.53	4

Table 2: Comparison of Runtime Speedup between SOTA and NCG Algorithms

Instance	SOTA CG Iteration	NCG Iteration	Iteration Increase (%)	PC
2l-cvrp0102-90&3	46	47	2.17	2
2l-cvrp1902-160&11	117	173	47.86	2
2l-cvrp1802-2010&9	132	246	86.36	2
2l-cvrp1702-60&14	69	66	-4.35	2
2l-cvrp1602-67&11	56	63	12.50	2
2l-cvrp1402-8000&7	80	126	57.50	2
2l-cvrp1302-38000&7	65	92	41.54	2
2l-cvrp1202-68&9	68	77	13.24	2
2l-cvrp1102-4500&6	95	121	27.37	2
2l-cvrp1002-4500&6	71	85	19.72	2
2l-cvrp0902-48&8	34	31	-8.82	2
2l-cvrp0802-4500&5	53	63	18.87	2
2l-cvrp1502-8000&6	136	175	28.68	2
2l-cvrp0202-55&5	24	25	4.17	2
2l-cvrp0602-4000&6	42	45	7.14	2
2l-cvrp0702-4500&5	69	66	-4.35	2
2l-cvrp0502-6000&4	61	58	-4.92	2
2l-cvrp0302-85&5	42	48	14.29	2
2l-cvrp0402-58&6	30	31	3.33	2
2l-cvrp1903-160&11	123	162	31.71	3
2l-cvrp0103-90&3	47	38	-19.15	3
2l-cvrp1803-2010&10	64	103	60.94	3
2l-cvrp1703-60&14	54	56	3.70	3
2l-cvrp0203-55&5	25	30	20	3
2l-cvrp1603-67&11	59	62	5.08	3
2l-cvrp1503-8000&6	136	239	75.74	3
2l-cvrp1403-8000&7	70	70	0	3
2l-cvrp0703-4500&5	61	63	3.28	3
2l-cvrp0603-4000&6	39	46	17.95	3
2l-cvrp1303-38000&7	52	62	19.23	3
2l-cvrp1203-68&9	58	57	-1.72	3
2l-cvrp0403-58&6	35	35	0	3
2l-cvrp1103-4500&7	52	79	51.92	3
2l-cvrp1003-4500&6	58	68	17.24	3
2l-cvrp0503-6000&4	60	53	-11.67	3
2l-cvrp0903-48&8	54	59	9.26	3
2l-cvrp0803-4500&5	70	71	1.43	3
2l-cvrp0303-85&5	42	48	14.29	3
2l-cvrp1204-68&9	73	75	2.74	4
2l-cvrp1304-38000&7	67	121	80.60	4
2l-cvrp1404-8000&7	73	71	-2.74	4
2l-cvrp1804-2010&10	103	224	117.48	4
2l-cvrp1604-67&11	68	81	19.12	4
2l-cvrp1704-60&14	104	108	3.85	4
2l-cvrp1104-4500&7	82	165	101.22	4
2l-cvrp1504-8000&8	46	117	154.35	4
2l-cvrp1004-4500&7	52	48	-7.69	4
2l-cvrp0504-6000&4	72	74	2.78	4
2l-cvrp0804-4500&5	47	55	17.02	4
2l-cvrp0704-4500&5	77	80	3.90	4
2l-cvrp0604-4000&6	44	46	4.55	4
2l-cvrp0404-58&6	30	34	13.33	4
2l-cvrp0304-85&5	30	29	-3.33	4
2l-cvrp0204-55&5	24	24	0	4
2l-cvrp0104-90&4	22	21	-4.55	4
2l-cvrp0904-48&8	58	57	-1.72	4
2l-cvrp1904-160&12	89	128	43.82	4

Table 3: Comparison of CG iteration increase between SOTA and NCG Algorithms

C Model Details

C.1 Training Details of the ML Model

The table below (Table 4) outlines the key parameters and their respective values used during the training process.

Parameter	Value
Hidden Dimension	16
Number of Heads in MHA	4
Number of Layers in MHA	2
Number of GRU Layers	1
Batch Size	512
Learning Rate of Adam	0.0001
Data Augmentation Multiplicity	10

Table 4: Training Details of the ML Model

C.2 Loss Function

Our machine learning model employs a weighted binary cross-entropy loss function, tailored for class imbalance. The loss is computed as follows:

$$\text{Loss}(p, y) = -\frac{1}{N}\sum_{i=1}^{N}\left[w_i \cdot y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)\right] \tag{1}$$

where p denotes the model's probability output, y is the target label, and N is the number of observations. The weight w_i for each observation is determined as the ratio of the number of positive samples to negative samples in the dataset. This weighting strategy aims to mitigate the impact of class imbalance on the loss function.

C.3 Ablation Study Details

In our ablation study, we evaluate the impact of various components of our ML model by comparing it against three variants:

Without Data Augmentation This model variant, labeled as 'w/o augmentation', is identical to our primary model except for the data augmentation aspect. Here, we set the Data Augmentation Multiplicity parameter to 1, effectively disabling the permutation-based data augmentation strategy.

Without Attention Mechanism In this variant, labeled as 'w/o attention mechanism', the attention mechanism is replaced with a MLP. The MLP consists of two linear layers with a ReLU activation function in between. The first linear layer maps the 2-dimensional input (normalized width and length of items) to the hidden size, followed by the ReLU activation and another linear layer that maps back to the hidden size. The computation is as follows:

$$\text{MLP}(x) = W_2 \cdot \text{ReLU}\left(W_1 \cdot x + b_1\right) + b_2, \tag{2}$$

where x is the input vector, W_1, W_2 are weight matrices, and b_1, b_2 are bias vectors.

Without Recurrence Mechanism The third variant, labeled as 'w/o recurrence mechanism', replaces the GRU-based recurrence mechanism with a Transformer encoder layer equipped with sinusoidal positional encoding. The positional encoding is added to each item's representation to incorporate sequence information. After adding positional encoding, the model employs a single-layer MHA mechanism. The output from the MHA layer is then subjected to mean pooling to derive a global representation of the state. This global representation is then processed similarly to our primary model, passing through a feed-forward layer and a sigmoid function to predict the probability. The computation is as follows:

$$\overline{h}_{i,m} = h_{i,m} + \text{PE}(i), \tag{3}$$

$$\widetilde{h}_{i,m} = \text{MHA}_{i,m}\left(\overline{h}_{1,1}, ..., \overline{h}_{n,|M_n|}\right), \tag{4}$$

$$\textit{probability} = \text{sigmoid}\left(\text{FF}\left(\frac{1}{\sum_{i=1}^{n}|M_i|}\sum_{i=1}^{n}\sum_{m=1}^{|M_i|}\widetilde{h}_{i,m}\right)\right), \tag{5}$$

where PE is defined for each position i in the sequence as a vector of dimension d_{model}, with each element given by:

$$\text{PE}(i)[2k] = \sin\left(\frac{i}{10000^{2k/d_{\text{model}}}}\right), \tag{6}$$

$$\text{PE}(i)[2k+1] = \cos\left(\frac{i}{10000^{2k/d_{\text{model}}}}\right), \tag{7}$$

for $k = 0, 1, \ldots, \frac{d_{\text{model}}}{2} - 1$. Here, i represents the position, and d_{model} is the dimension of the model. For the detailed implementation of positional encoding, please refer to [Vaswani *et al.*, 2017].

References

[Iori *et al.*, 2007] Manuel Iori, Juan-José Salazar-González, and Daniele Vigo. An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, page 253–264, May 2007.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Zhang *et al.*, 2022a] Xiangyi Zhang, Lu Chen, Michel Gendreau, and André Langevin. A branch-and-price-and-cut algorithm for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, 56(6):1618–1635, 2022.

[Zhang *et al.*, 2022b] Xiangyi Zhang, Lu Chen, Michel Gendreau, and André Langevin. Learning-based branch-and-price algorithms for the vehicle routing problem with time windows and two-dimensional loading constraints. *INFORMS Journal on Computing*, 34(3):1419–1436, 2022.