ROTATIONAL EQUIVARIANCE IMPROVES NEURAL ROUTING SOLVERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Although rotational-equivariant neural networks have proven to be an effective architectural bias across domains from chemistry to robotics, such approaches remain largely under-explored in the design of neural solvers for combinatorial optimization. In this work, we focus on Euclidean routing problems—specifically, the traveling salesman problem and the capacitated vehicle routing problem. For these problems, we experimentally demonstrate that converting existing neural solvers into their rotational-equivariant versions can boost their performance and generalization capabilities.

1 INTRODUCTION

In recent years, designing neural solvers for combinatorial optimization problems (e.g., traveling salesman problem, vehicle routing problem) have attracted growing interest (Ba et al., 2026), with the promise of obtaining fast heuristics that can specialize on a specific instance distribution or generalize to large-size instances. Progress has been driven by advances in novel neural architectures (Fang et al., 2024; Jin et al., 2023; Zhou et al., 2026) and training methods (Li et al., 2025; Kwon et al., 2020; Min et al., 2023; Luo et al., 2024; Pan et al., 2025), notably exploiting problem symmetries, which can provide useful prior knowledge to build efficient solvers. For instance, attention-based neural solvers are invariant node permutations (Deudon et al., 2018; Kool et al., 2019) or multi-start sampling (Kwon et al., 2020) exploits invariance with respect to the choice of the initial node for the solution generation.

Euclidean routing problems possess additional geometric symmetries: rotations, reflections, and translations preserve pairwise distances, feasible routes, and the set of optimal solutions. Previous work mainly exploits this invariance via canonicalization (Deudon et al., 2018; Ouyang et al., 2021) and data augmentation (Kwon et al., 2020; Kim et al., 2022). Canonicalization transforms an instance into a standard form before solving it; for instance, a neural solver become translational invariant by centering coordinates at the node centroid. With data augmentation, multiple transformed copies of an instance are solved at training or inference time to find the best solution. However, augmentation alone does not guarantee consistent predictions under arbitrary rotations.

Equivariant neural networks encode symmetry in their architecture. Despite successful applications in several geometric learning domains ranging from chemistry (Liao & Smidt, 2022; Liao et al., 2023) to robotics (Wang & Jörnsten, 2024; Gao et al., 2024), their use in neural routing remains underexplored. This raises a broader question: *To what extent can rotational equivariance improve existing neural routing solvers?*

We investigate this question through controlled adaptations of two established solvers: POMO (Kwon et al., 2020), an attention-based neural solver, trained via reinforcement learning, which autoregressively constructs solutions, and DRHG (Li et al., 2025), a destroy and repair method, where candidate solutions are repeatedly partially destroyed routes then repaired by an attention-based neural solver trained via supervised learning. We convert the neural architectures of these neural solvers into equivariant versions (Eq-POMO and Eq-DRHG), leaving their respective training and evaluation process mostly unchanged. Our equivariant construction is inspired by SE(3)-Transformer (Fuchs et al., 2020), but requires adaptation of its feature representations to planar rotations. Moreover, we modify its attention architecture to reduce computational cost. Our experiments on TSP and CVRP show that these rotational-equivariant adaptations can improve solution quality and gen-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

eralization across sizes and distributions. These results provide empirical evidence that rotational equivariance is a useful structural bias across distinct neural routing architectures.

To summarize, our contributions are threefold:

- We provide a reusable package for building two-dimensional equivariant neural networks (e.g., linear map, normalization, and self-, cross-, and pointer-attention modules). It can be used to adapt other neural solvers, but also in other application domains.
- Using this package, we develop Eq-POMO and Eq-DRHG, adapting the original solvers to their equivariant versions with minimal architectural and training scheme changes.
- We compare both adaptations with their original solvers on TSP and CVRP, evaluating solution quality and generalization across instance sizes and distributions, as well as real-world instances in TSPlib and CVRPlib.

2 Related Work

We review equivariant neural architectures and neural routing solvers, then position our work relative to methods that exploit routing symmetries.

Equivariant architectures and attention. Group-equivariant convolutions extend parameter sharing of standard CNN beyond translations, and steerable networks allow feature channels to transform under different group representations (Cohen & Welling, 2016; Weiler & Cesa, 2019). Tensor Field Networks construct equivariant messages for three-dimensional point clouds, while the SE(3)-Transformer weights those messages with invariant attention scores (Thomas et al., 2018; Fuchs et al., 2020). General kernel constructions and the ESCNN library support analogous layers for a wider range of groups (Cesa et al., 2022). We use these established representation and kernel constraints for $\mathrm{SO}(2)$, integrating into routing policies with nodewise attention projections and task-specific invariant outputs. Section 4.4 provides a comparison with the SE(3)-Transformer.

Neural combinatorial optimization. Neural routing methods can be organized by how they construct or improve a solution, separately from whether they use supervised or reinforcement learning and autoregressive or non-autoregressive prediction (Ba et al., 2026). Constructive methods select nodes sequentially: the Attention Model (Kool et al., 2019) uses an attention encoder and a pointer-style decoder, POMO (Kwon et al., 2020) adds multiple starting nodes and a shared reinforcement-learning baseline, and LEHD (Luo et al., 2023) pairs a light encoder with a heavy decoder to support generalization across problem sizes. Improvement methods instead modify an existing solution. DACT (Ma et al., 2021) learns local route edits, while DRHG (Li et al., 2025) repairs partially destroyed routes by reconnecting retained segments. Heatmap-based methods predict edge scores that guide subsequent tour decoding. DIFUSCO (Sun & Yang, 2023) generates these heatmaps through diffusion-based denoising, allowing it to represent diverse candidate solutions. POMO and DRHG let us examine a common architectural bias in both constructive and repair policies, under different training objectives.

Symmetry in routing. Existing routing solvers exploit geometric symmetry through augmentation, training objectives, and architectural design. For instance, POMO (Kwon et al., 2020) exploits multiple rollouts from different starting nodes and instance augmentation during inference, whereas Sym-NCO (Kim et al., 2022) encourages consistent behavior across symmetric instances through its training losses. In addition, INViT (Fang et al., 2024) constructs normalized local views to obtain invariant representations. Our work establishes the empirical value of rotational equivariance as a structural bias through controlled adaptations of existing neural routing solvers.

3 Preliminaries

We first define the routing tasks and recall the original POMO and DRHG architectures. We then introduce the established equivariant operations needed to understand our adaptations in Section 4.

3.1 EUCLIDEAN ROUTING AND AUTOREGRESSIVE POLICIES

An Euclidean routing instance $\mathcal{I}$ is defined by a set $\mathcal{V}$ of N nodes with coordinates $\boldsymbol{x}_i \in \mathbb{R}^2$, and travel costs $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$. For TSP, $\mathcal{V} = \{1, \dots, n\}$ and a feasible solution is a cyclic ordering π of all nodes, with length $\sum_{t=1}^{n} d_{\pi_t, \pi_{t+1}}$, where $\pi_{n+1} = \pi_1$. For CVRP, $\mathcal{V} = \{0, 1, \dots, n\}$ including depot 0, each customer i has demand $q_i > 0$, and vehicles have capacity Q. A feasible solution comprises depot-to-depot routes serving each customer exactly once without exceeding capacity on any route. Both tasks minimize the total travel cost $C(S; \mathcal{I})$ over feasible solutions $S \in \mathcal{S}(\mathcal{I})$.

An autoregressive policy $p_\theta(a_t \mid s_t, \mathcal{I})$ selects an action a_t based on the current partially-constructed route state s_t. This state records the partial route endpoints and visited nodes for TSP, and includes furthermore the remaining capacity c_t^{rem} for CVRP. We write $\mathcal{A}(s_t)$ for the allowed actions and v_t for the current node before selecting a_t. For TSP, the policy directly selects the next node, whereas for CVRP, it also specifies whether to visit that node directly or through the depot.

3.2 ORIGINAL POMO AND DRHG SOLVERS

We recall two routing solvers: POMO and DRHG, which we will adapt to rotational equivariant versions. Both use an attention-based encoder-decoder architecture. We select POMO since it is a classic simple autoregressive neural solver, and consider DHRG since it is a recent state-of-the-art neural solver using a destroy and repair technique. Both models use an encoder–decoder architecture with multi-head attention (MHA), which combines several independently projected attention heads. We write $\mathrm{MHA}(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{V})$ when queries are projected from $\boldsymbol{U}$ and keys and values from $\boldsymbol{V}$.

POMO. It combines an attention-based policy with multiple rollouts per instance (Kwon et al., 2020). Following the Attention Model (Kool et al., 2019), an affine map embeds each coordinate, normalized to the unit square; CVRP adds normalized customer demand q_i/Q and uses a separate depot embedding. At a layer ℓ, with $\boldsymbol{H}^\ell$ being the node embeddings:

$$\widetilde{\boldsymbol{H}}^\ell = \mathrm{Norm}^\ell_{\mathrm{ord},1}\Big(\boldsymbol{H}^\ell + \mathrm{MHA}^\ell(\boldsymbol{H}^\ell, \boldsymbol{H}^\ell, \boldsymbol{H}^\ell)\Big), \boldsymbol{H}^{\ell+1} = \mathrm{Norm}^\ell_{\mathrm{ord},2}\Big(\widetilde{\boldsymbol{H}}^\ell + \mathrm{MLP}^\ell(\widetilde{\boldsymbol{H}}^\ell)\Big) \quad (1)$$

where $\mathrm{Norm}_{\mathrm{ord}}$ is the baseline normalization: the released POMO architecture uses instance normalization, which standardizes each hidden coordinate across nodes within an instance and applies learned affine parameters. Its decoder uses the first and current node embeddings for TSP, or the current node embedding and remaining capacity for CVRP; it omits the graph-mean context used by the original Attention Model. The context attends to feasible nodes, and a pointer-style scoring layer assigns next-node probabilities (Vinyals et al., 2015).

POMO assigns distinct first nodes for TSP, or distinct first customers after the depot for CVRP. For M rollouts, let $R^{(m)} = -C(S^{(m)}; \mathcal{I})$ and $\bar{R} = M^{-1} \sum_m R^{(m)}$. Its REINFORCE loss uses the shared within-instance baseline:

$$\mathcal{L}_{\mathrm{POMO}} = -\frac{1}{M} \sum_{m=1}^{M} \mathrm{sg}\Big(R^{(m)} - \bar{R}\Big) \sum_t \log p_\theta(a_t^{(m)} \mid s_t^{(m)}, \mathcal{I}). \quad (2)$$

Here sg stops gradients through the sampled advantage, and the inner sum excludes the assigned opening actions. Inference returns the lowest-cost rollout. Appendix B.1 recalls the full decoder.

DRHG. It improves a complete solution by repeated removing randomly-selected edges and learning to reconnect the resulting segments (subpaths or isolated nodes) (Li et al., 2025). Its encoder represents each segment by its two endpoints. For CVRP, the input also includes $\widetilde{q}_i/Q$, where $\widetilde{q}_i$ is the total demand in a segment. Before embedding, DRHG translates and uniformly rescales the reduced coordinates using their bounding box and may exchange the coordinate axes.

At each repair step, the decoder uses *context tokens* $\boldsymbol{O}_t$ (projected first/current node embeddings for TSP, or depot/current node embeddings for CVRP) to rebuild the solution. The combined token sequence $\boldsymbol{I}_t$ contains these context tokens and the remaining candidate tokens. Each decoder layer first updates the context from the combined set, then updates the combined set from the new context:

$$\begin{aligned} \boldsymbol{O}_t^{\ell+1} &= \mathcal{B}^\ell_{\mathrm{ord}}\Big(\boldsymbol{O}_t^\ell, \mathrm{MHA}^\ell(\boldsymbol{O}_t^\ell, \boldsymbol{I}_t^\ell, \boldsymbol{I}_t^\ell)\Big), \\ \boldsymbol{I}_t^{\ell+1} &= \mathcal{B}'^\ell_{\mathrm{ord}}\Big(\boldsymbol{I}_t^\ell, \mathrm{MHA}'^\ell(\boldsymbol{I}_t^\ell, \boldsymbol{O}_t^{\ell+1}, \boldsymbol{O}_t^{\ell+1})\Big). \end{aligned} \quad (3)$$

Here $\mathcal{B}_{\mathrm{ord}}$ denotes the ordinary residual attention and feed-forward update. A linear layer produces one score per candidate in TSP. In CVRP, it produces two scores per candidate: one for connecting directly and one for connecting via the depot. A single softmax over these scores assigns a probability to each available action. This model is trained by supervised learning, with an optimizer update after each repair step. Appendix B.2 provides the architecture and training details.

3.3 Equivariant neural networks

We use the representation-based construction of steerable networks (Weiler & Cesa, 2019; Cesa et al., 2022) and the invariant-weight attention principle of the SE(3)-Transformer (Fuchs et al., 2020). This subsection states the constraints; Section 4.1 specifies our implementation choices.

Equivariance and Invariance. Let F_θ be a neural-network layer, and let $\rho_{\mathrm{in}}(\varphi)$ and $\rho_{\mathrm{out}}(\varphi)$ denote the linear transformations applied to its input and output features under a rotation by φ. The layer is *rotation equivariant* if transforming the input feature vector $\boldsymbol{f}$ before applying the layer is equivalent to transforming the layer's output:

$$F_\theta(\rho_{\mathrm{in}}(\varphi)\boldsymbol{f}) = \rho_{\mathrm{out}}(\varphi)F_\theta(\boldsymbol{f}) \qquad \text{for every angle } \varphi. \tag{4}$$

A map ρ assigning each rotation φ a linear transformation $\rho(\varphi)$ is a *group representation* if it satisfies $\rho(\varphi_1 + \varphi_2) = \rho(\varphi_1)\rho(\varphi_2)$ and $\rho(0) = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. If $\rho_{\mathrm{out}}(\varphi) = \boldsymbol{I}$ for all angles, the layer is called *rotation invariant*. Using these properties, we can allow internal features to rotate while requiring action scores to remain fixed. The target routing policy is rotation invariant and therefore assigns the same probability to each node label after rotating the instance.

Feature types. For $\mathrm{SO}(2)$ (group of 2D rotations), the building blocks of features are scalar type-0 channels and 2D type-k channels, $k \geq 1$ (Weiler & Cesa, 2019). Their transformation laws are

$$\rho_0(\varphi) = 1, \qquad \rho_k(\varphi) = \boldsymbol{R}_{k\varphi} = \begin{pmatrix} \cos(k\varphi) & -\sin(k\varphi) \\ \sin(k\varphi) & \cos(k\varphi) \end{pmatrix}, \quad k \geq 1. \tag{5}$$

Thus demands are type-0, ordinary coordinates are type-1, and a type-2 channel rotates through twice the input angle. These indivisible transformation blocks are called *irreducible representations*. A typed hidden feature concatenates chosen numbers m_k of each block; its dimension is $m_0 + 2\sum_{k\geq 1} m_k$. We denote different type-k channels by $\boldsymbol{f}^{(k,a)}$ (a scalar when $k = 0$), $a \in \{1, \ldots, m_k\}$. The explicit block transformation and the resulting constraints on equivariant linear maps are derived in Appendix A.

Equivariant operations on typed features. Every operation in an equivariant network must obey the property in Equation (4) for its input and output feature types. Our models use equivariant linear maps, steerable projections, channelwise nonlinearities and normalization.

An *equivariant linear map* $\boldsymbol{W} \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ satisfies $\boldsymbol{W}\rho_{\mathrm{in}}(\varphi) = \rho_{\mathrm{out}}(\varphi)\boldsymbol{W}$; for a feature space $\mathcal{F}$, it can mix copies of the same type but cannot map between different types; an affine bias is allowed only in scalar output channels.

A *steerable projection* overcomes this restriction by allowing the operator to depend on a coordinate $\boldsymbol{z} \in \mathbb{R}^2$, with $F_{\boldsymbol{K}}(\boldsymbol{z}, \boldsymbol{f}) = \boldsymbol{K}(\boldsymbol{z})\boldsymbol{f}$ where a steerable kernel $\boldsymbol{K}(\boldsymbol{z})$ is a matrix in $\mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ and $\boldsymbol{K}(\boldsymbol{z})\boldsymbol{f}$ is an equivariant embedding. Equivariance of $F_{\boldsymbol{K}}$ then requires the kernel constraint:

$$\boldsymbol{K}(\boldsymbol{R}_\varphi \boldsymbol{z})\rho_{\mathrm{in}}(\varphi) = \rho_{\mathrm{out}}(\varphi)\boldsymbol{K}(\boldsymbol{z}). \tag{6}$$

Because rotations mix the two components of each non-scalar channel, *channelwise nonlinearities* act on that channel as a whole rather than componentwise. Each non-scalar channel is rescaled by a scalar function $g_{k,a}$ of rotation-invariant quantities $\boldsymbol{\iota}(\boldsymbol{f})$:

$$\boldsymbol{\iota}(\boldsymbol{f}) = \left((f^{(0,a)})_{a=1}^{m_0}, (\|\boldsymbol{f}^{(k,a)}\|_2)_{k>0,\, a=1}^{m_k} \right). \tag{7}$$

Every learned scalar function $g_{k,a}$ gives an equivariant rescaling

$$\boldsymbol{f}^{(k,a)} \longmapsto g_{k,a}(\boldsymbol{\iota}(\boldsymbol{f}))\, \boldsymbol{f}^{(k,a)}, \qquad k > 0. \tag{8}$$

Normalization follows the same principle: standard LayerNorm is applied to the invariant summary $\boldsymbol{\iota}(\boldsymbol{f})$, and its outputs are used as scalar factors to rescale the corresponding feature channels,

$$u_b = \max\{|\iota(\boldsymbol{f})_b|, \epsilon\}, \qquad [\mathcal{N}(\boldsymbol{f})]_b = \frac{[\mathrm{LN}(\boldsymbol{\iota}(\boldsymbol{f}))]_b}{u_b}\boldsymbol{f}_b. \tag{9}$$

Here b enumerates all representation blocks, with $\boldsymbol{f}_b \in \mathbb{R}^{d_b}$ and $d_b \in \{1, 2\}$.

Invariant attention weights. Dot-product attention preserves equivariance when the query, key, and value projections are equivariant and the queries and keys have matching feature types. Their inner products are then rotation invariant, giving invariant attention weights:

$$\alpha_{ij}^h = \mathrm{softmax}_j\left(\frac{\langle \boldsymbol{q}_i^h, \boldsymbol{k}_j^h\rangle}{\sqrt{D_h}}\right), \qquad \boldsymbol{o}_i^h = \sum_j \alpha_{ij}^h \boldsymbol{v}_j^h. \tag{10}$$

Because the weights are invariant, the weighted sum $\boldsymbol{o}_i^h$ transforms equivariantly like the values $\boldsymbol{v}_j^h$. Multi-head attention remains equivariant when heads contain whole representation blocks.

4 Equivariant routing models

Using the tools of Section 3.3, we now present how to build equivariant adaptations[1] of the neural solvers used in POMO and DHRG. We first specify the shared components, then describe each solver adaptation and the relation to the SE(3)-transformer.

4.1 Coordinates and input embeddings

For $N = |\mathcal{V}|$ nodes of a full or reduced problem, we center the coordinates $\boldsymbol{x}_i$ of each node $i \in \mathcal{V}$ and divide by the maximum distance to their centroid:

$$\boldsymbol{z}_i = \frac{\boldsymbol{x}_i - \bar{\boldsymbol{x}}}{s} \qquad \text{where } \bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i\in\mathcal{V}} \boldsymbol{x}_i, \qquad s = \max\left\{\max_{i\in\mathcal{V}} \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|_2, \epsilon\right\}. \tag{11}$$

The floor $\epsilon > 0$ handles a degenerate scale, and a rotation and translation of the input change $\boldsymbol{z}_i$ only by the corresponding rotation. Given a typed input $\boldsymbol{f}_i$, the initial embedding is $\boldsymbol{h}_i^0 = \mathcal{N}_0(\boldsymbol{K}(\boldsymbol{z}_i)\boldsymbol{f}_i)$, where $\boldsymbol{K}$ is the steerable kernel of Equation (6) and $\mathcal{N}_0$ is the equivariant normalization in Equation (9). This coordinate-dependent embedding allows an input feature of a given type to be embeded into any designated type.

For any coordinate $\boldsymbol{z} \neq \boldsymbol{0}$, expressed in polar form, $\boldsymbol{z} = r(\cos\phi, \sin\phi)^\top$, a kernel block from input type p to output type k has the established expansion following Weiler et al. (2018),

$$\boldsymbol{K}^{kp}(\boldsymbol{z}) = \sum_\nu w_\nu^{kp}(r)\boldsymbol{B}_\nu^{kp}(\phi). \tag{12}$$

The radial coefficients w_ν^{kp}, parameterized by an MLP, are invariant since they depend only on the invariant radius r. Terms $\boldsymbol{B}_\nu^{kp}$ are fixed angular basis matrices of size $d_k \times d_p$, with $d_0 = 1$ and $d_k = 2$ for $k > 0$. Each input/output channel pair has its own set of allowed basis matrices indexed by ν, satisfying Equation (6) by construction. Appendix A gives the explicit $\mathrm{SO}(2)$ bases.

4.2 Eq-POMO: equivariant route construction

Eq-POMO is POMO with equivariant attention. For TSP, its initial feature $\boldsymbol{f}_i$ is the type-1 vector $\boldsymbol{z}_i$; for CVRP, each customer additionally carries the normalized demand q_i/Q as a type-0 scalar, while a separate coordinate embedding handles the depot.

For the matrix $\boldsymbol{Z} = (\boldsymbol{z}_i)_{i\in\mathcal{V}}$ of normalized coordinates, the encoder computes for a layer ℓ:

$$\widetilde{\boldsymbol{H}}^\ell = \mathcal{N}_1^\ell\Big(\mathrm{Skip}^\ell(\boldsymbol{H}^\ell) + \mathrm{MHA}_{\mathrm{eq}}^\ell(\boldsymbol{H}^\ell, \boldsymbol{Z}, \boldsymbol{Z})\Big), \quad \boldsymbol{H}^{\ell+1} = \mathcal{N}_2^\ell\Big(\widetilde{\boldsymbol{H}}^\ell + \mathrm{FFN}_{\mathrm{eq}}^\ell(\widetilde{\boldsymbol{H}}^\ell)\Big). \tag{13}$$

[1] Our implementation uses ESCNN (Weiler & Cesa, 2019; Cesa et al., 2022) for feature representations and constrained linear maps, together with explicit planar steerable kernel bases.

Here Skip^{ℓ} is a learned equivariant linear projection adopted from the SE(3)-transformer. $\mathcal{N}_1^{\ell}$ and $\mathcal{N}_2^{\ell}$ normalize the attention and feed-forward residuals. The equivariant multi-head self-attention adds a scalar distance bias term as in ICAM (Zhou et al., 2026):

$$e_{ij}^{h} = \frac{\langle \boldsymbol{q}_i^h, \boldsymbol{k}_j^h \rangle}{\sqrt{D_h}} - \eta_h \widehat{d}_{ij} \log N, \qquad \text{where } \widehat{d}_{ij} = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2. \tag{14}$$

Here η_h is learned, $\widehat{d}_{ij}$ is measured in the normalized coordinate frame. This bias is also applied to the pointer attention in the decoder.

Let $\boldsymbol{h}_i$ be the node embedding generated by the encoder for a node i. At decoding step t, let b_t be the first selected node for TSP or the depot for CVRP; v_t remains the current node. The decoder builds context $\boldsymbol{c}_t$ and query $\boldsymbol{q}_t$ with

$$\boldsymbol{c}_t = \boldsymbol{W}_g \left(\frac{1}{N} \sum_{i \in \mathcal{V}} \boldsymbol{h}_i \right) + \boldsymbol{W}_e [\boldsymbol{h}_{b_t} \oplus \boldsymbol{h}_{v_t}], \qquad \boldsymbol{q}_t = \begin{cases} \boldsymbol{c}_t, & \text{TSP}, \\ \boldsymbol{c}_t + a_\theta(c_t^{\mathrm{rem}}/Q) \boldsymbol{W}_c \boldsymbol{c}_t, & \text{CVRP}. \end{cases} \tag{15}$$

Here $\boldsymbol{W}_g$ and $\boldsymbol{W}_e$ are equivariant linear maps for the graph mean and endpoints of the current partial solution. The scalar function a_θ conditions a constrained equivariant residual on the remaining-capacity ratio c_t^{rem}/Q. Subject to the invariant attention weight and equivariant component used, Eq-POMO retains equivariant hidden features until the final invariant score.

4.3 Eq-DRHG: equivariant route repair

Eq-DRHG replaces DRHG's neural repair policy while retaining its destroy-and-repair workflow. On the reduced problem, let $\tau(i)$ be node i's paired endpoint for subpaths, with $\tau(i) = i$ for an isolated node. Its initial embedding receives $\boldsymbol{f}_i = \boldsymbol{z}_i \oplus \boldsymbol{z}_{\tau(i)}$ for TSP, with the additional scalar channel $\widetilde{q}_i/Q$ for CVRP.

Equivariant linear maps construct context tokens $\boldsymbol{O}_t$ and combined token set $\boldsymbol{I}_t$ (Section 3.2). Each layer keeps DRHG's alternating cross attention structure but uses equivariant operations: for layer ℓ

$$\begin{aligned} \boldsymbol{O}_t^{\ell+1} &= \mathcal{B}_{\mathrm{eq}}^{\ell}\Big(\boldsymbol{O}_t^{\ell}, \mathrm{MHA}_{O \leftarrow I}^{\mathrm{eq},\ell}(\boldsymbol{O}_t^{\ell}, \boldsymbol{I}_t^{\ell}, \boldsymbol{I}_t^{\ell})\Big), \\ \boldsymbol{I}_t^{\ell+1} &= \mathcal{B}_{\mathrm{eq}}^{\prime\ell}\Big(\boldsymbol{I}_t^{\ell}, \mathrm{MHA}_{I \leftarrow O}^{\mathrm{eq},\ell}(\boldsymbol{I}_t^{\ell}, \boldsymbol{O}_t^{\ell+1}, \boldsymbol{O}_t^{\ell+1})\Big). \end{aligned} \tag{16}$$

Here $\mathcal{B}_{\mathrm{eq}}^{\ell}, \mathcal{B}_{\mathrm{eq}}^{\prime\ell}$ add a Post-LayerNorm to the original attention and residual block. The invariant summary $\boldsymbol{\iota}(\boldsymbol{f})$ is fed into the final linear layer to generate invariant candidate scores.

The supervised learning task is unchanged, but Eq-DRHG differs by holding parameters fixed throughout a batch of reduced-problem rollouts and makes one optimizer update per batch rather per decoding step Appendix B.3.

4.4 Comparison with the SE(3)-Transformer

Our attention shares the SE(3)-Transformer's central construction: invariant attention weights aggregate equivariant values (Fuchs et al., 2020). Adapting it to planar rotations changes the feature types: a non-scalar type of order k has $2k+1$ components in three dimensions, but only two in the plane. The more consequential architectural change concerns how keys and values are computed. In the SE(3)-Transformer, they depend on the displacement from query node i to neighbor node j:

$$\boldsymbol{q}_i = \boldsymbol{W}_Q \boldsymbol{f}_i, \quad \boldsymbol{k}_{ij} = \boldsymbol{K}_K(\boldsymbol{x}_j - \boldsymbol{x}_i)\boldsymbol{f}_j, \quad \boldsymbol{v}_{ij} = \boldsymbol{K}_V(\boldsymbol{x}_j - \boldsymbol{x}_i)\boldsymbol{f}_j, \quad \boldsymbol{o}_i = \sum_{j \in \mathcal{J}_i} \alpha_{ij} \boldsymbol{v}_{ij}, \tag{17}$$

Here $\boldsymbol{W}_Q$ is an equivariant linear map, $\boldsymbol{K}_K$ and $\boldsymbol{K}_V$ are steerable kernels, $\mathcal{J}_i$ is the neighborhood of node i, and α_{ij} is an invariant attention weight. Our models instead compute $\boldsymbol{q}_i = \boldsymbol{W}_Q \boldsymbol{h}_i$, $\boldsymbol{k}_j = \boldsymbol{W}_K \boldsymbol{h}_j$, and $\boldsymbol{v}_j = \boldsymbol{W}_V \boldsymbol{h}_j$ using equivariant linear maps. Geometric information is carried by the node features $\boldsymbol{h}_j$, while each key and value is computed once per node, independently of the query node. Dense attention still requires $O(N^2)$ query–key interactions, but computing keys and values once per node avoids evaluating steerable kernels for every node pair and all compatible input–output feature-type pair.

Table 1: Cross-size generalization on uniform TSP/CVRP (train on TSP/CVRP100; test on 10,000 instances for TSP/CVRP100, 128 instances for TSP/CVRP200, TSP/CVRP500, TSP/CVRP1000)

Method	TSP100			TSP200			TSP500			TSP1000		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
LKH3	7.763	0.000%	0.34s	10.704	0.000%	1.88s	16.522	0.000%	15.0s	23.12	0.00%	1.7m
Concorde	7.763	0.000%	0.20s	10.704	0.000%	1.41s	16.522	0.000%	15.0s	23.12	0.00%	1m
RI	8.513	9.662%	0.00s	11.948	11.627%	0.01s	18.546	12.252%	0.1s	26.11	12.90%	1s
DIFUSCO	7.780	0.240%	-	-	-	-	16.800	1.490%	1.7s	23.39	1.17%	11.5s
Fast T2T	7.764	0.010%	-	-	-	-	16.941	2.540%	-	24.015	3.870%	-
DACT	7.892	1.653%	0.07s	12.870	20.252%	0.41s	20.846	26.171%	1.6s	29.03	25.56%	7.8s
BQ	7.764	0.015%	0.17s	10.717	0.129%	0.94s	16.617	0.579%	5.5s	23.43	1.37%	13s
LEHD	7.763	0.002%	1.04s	10.706	0.0182%	4.92s	16.550	0.167%	33.8s	23.29	0.72%	3.3m
POMO	7.792	0.379%	0.00s	10.999	2.753%	0.05s	19.206	16.243%	0.32s	44.794	93.745%	0.51s
Eq-POMO	7.781	0.229%	0.01s	10.806	0.955%	0.12s	18.433	11.565%	0.72s	29.671	28.333%	4.54s
DRHG	7.763	0.001%	1.53s	10.705	0.013%	5.66s	16.540	0.115%	5.93s	23.174	0.234%	6.02s
Eq-DRHG	**7.763**	**0.000%**	2.15s	**10.704**	**0.004%**	12.09s	**16.537**	**0.093%**	12.25s	**23.168**	**0.210%**	12.68s

Method	CVRP100			CVRP200			CVRP500			CVRP1000		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
LKH3	15.647	0.00%	4.32s	20.173	0.00%	59.06s	37.229	0.00%	154.69s	37.090	0.00%	199.7s
HGS	15.563	-0.53%	1.62s	19.946	-1.13%	39.38s	36.561	-1.79%	112.50s	36.288	-2.16%	149.1s
Sweep	20.606	32.14%	0.01s	0.269	32.78%	0.01s	46.839	25.81%	0.01s	49.166	32.56%	0.1s
DACT	16.202	3.55%	0.14s	23.230	14.71%	1.02s	46.393	24.98%	3.83s		OOM	
BQ	15.806	1.02%	0.11s	20.362	0.94%	0.56s	37.606	1.01%	3.47s	38.147	2.88%	8.7s
LEHD	15.629	-0.11%	1.01s	**20.095**	**-0.38%**	19.69s	**37.100**	**-0.35%**	56.25s	**37.432**	**0.92%**	202.5s
POMO	15.900	1.619%	0.00s	23.373	15.860%	0.07s	44.794	20.320%	0.51s	78.427	111.450%	3.01s
Eq-POMO	15.849	1.295%	0.01s	22.974	13.886%	0.09s	39.902	7.180%	0.83s	57.672	55.489%	5.20s
DRHG	15.644	-0.020%	1.52s	20.123	-0.245%	8.16s	37.655	1.143%	10.97s	39.463	6.397%	10.69s
Eq-DRHG	**15.624**	**-0.143%**	4.82s	20.115	-0.286%	33.75s	37.625	1.063%	39.38s	39.675	6.968%	42.19s

5 Experimental Validation

In this section, we empirically demonstrate the effectiveness of rotational equivariance as an architectural bias. Our experiments are designed to answer three research questions: (**Q1**) Can our equivariant adaptations improve existing neural model performance? (**Q2**) Can they improve generalization abilities on unseen problem sizes and instance distributions? (**Q3**) Can improvements obtained by equivariance transfer beyond synthetic data to real-world scenarios?

5.1 Experimental Set-Up

Baseline. To answer these questions, we compare Eq-POMO and Eq-DRHG with their respective counterpart. For the usual evaluation on uniform TSP and CVRP, we also consider other baselines. *Heuristic solvers*: LKH3 (Helsgaun, 2017) is an extension of the famous Lin–Kernighan–Helsgaun heuristic, known for its excellent performance on TSP problems; HGS (Vidal, 2022) is one of the strongest heuristics for CVRP; Random Insertion (RI) and Sweep are constructive heuristic solvers. *Neural solvers*: In addition to POMO (Kwon et al., 2020) and DRHG (Li et al., 2025), we also select several representative methods, including DIFUSCO (Sun & Yang, 2023), Fast T2T (Li et al., 2024), DACT (Ma et al., 2021), BQ (Drakulic et al., 2023), and LEHD (Luo et al., 2023);

Training. We train Eq-POMO and Eq-DRHG following closely the training protocols of their respective methods. We keep most default hyperparameter values, but set the learning rate to $1e{-}3$ and $5e{-}3$ for Eq-POMO and Eq-DRHG, respectively. Also, we change the embedding dimension to 256, because we use 64 copies of both type-1 and type-2 representations ($64 \times 2 \times 2$. As a consequence, we expand the feedforward hidden dimension to 1024 to accommodate the larger embedding dimension. Note that the number of trainable parameters remains smaller for the equivariant variants. More detailed hyperparameter settings can be found in Appendix C.1. All the experiments are run on a single NVIDIA RTX 4090 GPU.

Table 2: Cross-distribution results on TSP100 and CVRP100 (evaluation on 128 instances per distribution, and times are reported per instance).

Distribution	Uniform				Normal				Cluster				Beta			
Method	TSP100		CVRP100		TSP100		CVRP100		TSP100		CVRP100		TSP100		CVRP100	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
POMO	7.79	0.00s	15.90	0.00s	6.88	0.00s	12.13	0.00s	3.81	0.00s	8.06	0.00s	5.06	0.00s	9.32	0.00s
Eq-POMO	7.78	0.01s	15.85	0.01s	6.83	0.01s	12.06	0.01s	3.73	0.01s	7.98	0.01s	5.01	0.01s	9.24	0.01s
DRHG	**7.75**	2.76s	15.48	9.27s	**6.75**	2.78s	11.93	9.64s	**3.64**	2.91s	7.57	9.74s	**4.96**	2.98s	8.99	9.44s
Eq-DRHG	**7.75**	6.98s	**15.45**	21.06s	**6.75**	6.68s	**11.91**	22.34s	**3.64**	6.98s	**7.56**	22.61s	**4.96**	9.07s	**8.98**	21.96s

Distribution	Exponential				Poisson				Explosion				Implosion			
Method	TSP100		CVRP100		TSP100		CVRP100		TSP100		CVRP100		TSP100		CVRP100	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
POMO	3.18	0.00s	5.65	0.01s	475.3	0.00s	887.4	0.01s	6.53	0.00s	13.64	0.00s	7.05	0.00s	15.38	0.00s
Eq-POMO	3.08	0.01s	5.54	0.01s	103.9	0.01s	186.5	0.01s	6.52	0.01s	13.60	0.01s	7.06	0.01s	15.42	0.01s
DRHG	**2.97**	2.66s	5.54	12.30s	103.2	2.68s	183.9	9.20s	**6.51**	3.02s	13.61	9.45s	**6.98**	2.77s	14.93	9.52s
Eq-DRHG	**2.97**	6.62s	**5.53**	23.81s	**103.0**	6.84s	**183.0**	21.59s	**6.51**	7.09s	**13.59**	20.88s	**6.98**	6.98s	**14.92**	20.68s

Evaluation. For Eq-POMO, we use greedy sampling under multistart and x8 instance augmentation. For Eq-DRHG, we consistently set $k \in [20, min(200, n)]$ for k-nn destruction in both TSP and CVRP, where n is the problem size. For improved comparability, we rerun POMO and DRHG, following their default configurations. For other baselines, we use their reported results. In the evaluation tables, we report the objective value (*Obj.*), the optimal gap (*Gap*) and the inference time (*Time*). *Obj.* measures average tour length while *Gap* measures the average relative gap to best known solutions. *Time* is computed as average per-instance time. To demonstrate the cross-size generalization ability, we follow previous practice and use different different number of instances for the evaluation: 10,000 instances for problem size 100 and 128 instances for sizes from 200 to 1000. We also consider various instance distributions to illustrate the cross-distribution generalization; a detailed description can be found in Appendix C.2.

5.2 Experimental results

Effectiveness of equivariant model. To answer the first research question Q1, we present the main results for TSP and CVRP in Table 1. On TSP100, Eq-POMO reduces the reported gap from 0.379% to 0.229%, corresponding to a 39.6% relative reduction in gap. Eq-DRHG also improves upon DRHG, reducing the reported gap from 0.001% to 0.000%. On CVRP100, the equivariant variants exhibit a similar trend, improving the gap from 1.619% to 1.295% for POMO, and from -0.020% to -0.143% for DRHG. The improvements persist as the problem size increases to 1000. Except that for CVRP1000, Eq-DRHG is slightly worse than original DRHG. Notably, on TSP1000, Eq-POMO reduced the gap from 93.745% to 28.333%, demonstrating a substantial improvement over POMO. Together, these results support the effectiveness of incorporating equivariance into existing neural routing solvers. We hypothesize that these gains arise from the structural bias introduced by rotational equivariance, which encourages the model to learn more expressive latent representations. Note that the effect of this bias is slightly less clear in eq-DHRG, since the neural solver is used inside a destroy and repair method, which limits the problem size of the reduced instances solved by equivariant model.

Generalization across distribution and size. To answer the second research question, we train the models using a uniform distribution and evaluate them across eight distributions. As shown in Table 2, Eq-POMO achieves lower average objectives than POMO on seven of the eight distributions for both problems. For example, on the Cluster distribution, Eq-POMO reduces the average objective from 3.81 to 3.73 for TSP100 and from 8.06 to 7.98 for CVRP100. Eq-DRHG exhibits smaller but consistent benefits relative to DRHG. For example, on the Poisson distribution, Eq-DRHG improves the objective from 103.2 to 103.0 for TSP100 and from 183.9 to 183.0 for CVRP100. The results therefore support improved cross-distribution performance for both Eq-POMO and Eq-DRHG. We hypothesize that rotational equivariance encourages geometric representations that remain useful across different instance distributions. We also consider cross-size generalization under different distributions, we extend the problem size from 100 to 1000. Detailed results are listed in Appendix C.3.

Table 3: Average gaps on TSPLIB/CVRPLIB clustered by problem size.

TSPLIB	$(0, 100)$	$[100, 200)$	$[200, 500)$	$[500, 1000]$	$(1000, \infty)$	**All**
POMO	0.59%	2.41%	8.80%	20.11%	39.74%	16.87%
Eq-POMO	0.22%	1.34%	7.22%	18.55%	33.25%	13.64%
DRHG	0.48%	0.15%	0.36%	0.26%	2.09%	0.95%
Eq-DRHG	**0.00%**	**0.01%**	**0.16%**	**0.23%**	**1.51%**	**0.61%**

CVRPLIB	**A**	**B**	**E**	**F**	**M**	**P**	**All**
POMO	4.97%	4.75%	11.40%	15.97%	4.86%	15.53%	8.68%
Eq-POMO	**1.66%**	**2.63%**	**2.05%**	13.37%	3.07%	**2.49%**	**2.61%**
DRHG	7.99%	7.66%	15.74%	24.00%	**2.91%**	11.62%	9.99%
Eq-DRHG	5.03%	5.73%	3.06%	**5.45%**	4.27%	6.21%	5.24%

Table 4: Fiber and bias ablations on four distributions. Results use 128 paired instances per experiment (seed 1234) and eight-start greedy decoding. Entries are mean tour lengths (lower is better); Δ is the relative increase without bias. Fiber labels give input/query–key multiplicities for orders 1 and 2.

(a) Fiber widths

N	Fiber	Uniform	Normal	Exponential	Explosion
100	$64/64$	7.8007	6.8920	3.2103	6.4829
	$64/32$	7.7986	6.8844	3.1325	6.4731
	$48/24$	7.8187	6.8951	3.2071	6.4823
200	$64/64$	10.8447	9.9438	4.7505	8.6597
	$64/32$	10.8345	9.9147	4.6543	8.6574
	$48/24$	10.8449	9.9007	4.8201	8.6417
500	$64/64$	18.4651	17.7709	8.3730	13.8322
	$64/32$	18.6961	17.6241	8.0863	14.0834
	$48/24$	18.4116	17.3288	8.3657	13.7925

(b) Bias

N	Setting	Uniform	Normal	Exponential	Explosion
100	Bias	7.7942	6.8237	3.1092	6.5648
	No bias	7.8296	6.8542	3.4237	6.5980
	$\Delta(\%)$	+0.45	+0.45	+10.12	+0.51
200	Bias	10.8620	9.9462	4.6135	8.4975
	No bias	10.9903	10.0508	5.2793	8.5973
	$\Delta(\%)$	+1.18	+1.05	+14.43	+1.17
500	Bias	18.7062	17.6666	7.9885	14.1502
	No bias	19.5684	18.3680	10.1818	14.5015
	$\Delta(\%)$	+4.61	+3.97	+27.46	+2.48

Transfer to TSPLIB and CVRPLIB benchmark. To answer the last research question Q3, we further extend the experimental evaluation to real-world datasets, i.e., TSPLIB and CVRPLIB, with results shown in Table 3. On TSPLIB, Eq-POMO reduces the overall average gap from 16.87% to 13.64%, while Eq-DRHG reduces it from 0.95% to 0.61%. Both equivariant variants improve upon their baselines across all five problem sizes. Notably, Eq-DRHG achieves the lowest average gap among the compared methods. On CVRPLIB, Eq-POMO reduces the overall average gap from 8.68% to 2.61%, while Eq-DRHG reduces it from 9.99% to 5.24%. In general, the rotational equivariant models improve their baselines. The exception is category M for Eq-DRHG, where the gaps increase from 2.91% to 4.27%. After inspection of instances under category M, we found that its instances frequently exhibit reflection symmetry. More detailed figures of TSPLIB and CVRPLIB can be found in Appendix C.4

Ablation on feature types The $64/32$ fiber gives the lowest mean on all four shown TSP-100 distributions, while $48/24$ leads on three of four at TSP-500. Removing bias raises the mean cost throughout the shown settings, most strongly on Exponential (up to 27.46% at TSP-500).

6 Conclusion

In this work, we explained how 2D rotational equivariant attention-based neural networks can be formulated. We adapted two existing neural solvers, POMO and DRHG, and formulated their equivariant versions. Our experimental evaluation on TSP and CVRP demonstrated that the equivariant POMO and DHRG can surpass their respective original versions in terms of cross-size and cross-distribution generalization. Moreover, they are also superior on average when evaluated on real problem instances. The improvements across both solvers suggest that our equivariance architectural bias is mostly agnostic to model architectures and training methods.

Future work for routing includes converting other neural solvers into their equivariant versions, optimizing the equivariant neural architecture for improved performance and speed, but also extending it to include other Euclidean transformations. Also, it could be applied to other application domains.

7 AI USE STATEMENT

In this work, we used generative AI tools for:

- implementing some of the equivariant modules,
- polishing the writing of the paper.

. We have not used generative AI tools for

- developing theoretical models or conceptual frameworks,
- designing or providing feedback on research methodology or experiments.
- interpreting results

and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for literature search and bibtex generation. We have reviewed all AI-assisted work. Notably, the information in the generated bibtex items was checked for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

8 ETHICS STATEMENT

Our work tests the effectiveness of rotational equivariance architectural bias in neural solvers. Since it investigates a more fundamental research question, we do not foresee any direct ethics issue related to our work.

9 REPRODUCIBILITY STATEMENT

To support reproducibility, we plan to release our code after the publication of our paper.

REFERENCES

Yunpeng Ba, Xi Lin, Changliang Zhou, Ruihao Zheng, Zhenkun Wang, Xinyan Liang, Zhichao Lu, Jianyong Sun, Yuhua Qian, and Qingfu Zhang. Survey on neural routing solvers. *arXiv preprint arXiv:2602.21761*, 2026.

Jieyi Bi, Yining Ma, Jiahai Wang, Zhiguang Cao, Jinbiao Chen, Yuan Sun, and Yeow Meng Chee. Learning generalizable models for vehicle routing problems via knowledge distillation. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Gabriele Cesa, Leon Lang, and Maurice Weiler. A program to build E(N)-equivariant steerable CNNs. In *International Conference on Learning Representations*, 2022.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/cohenc16.html.

Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis Martin Rousseau. Learning heuristics for the TSP by policy gradient. In *CPAIOR*, volume 10848 LNCS, pp. 170–181. Springer Verlag, 2018.

Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:77416–77429, 2023.

Han Fang, Zhihao Song, Paul Weng, and Yutong Ban. Invit: a generalizable routing problem solver with invariant nested view transformer. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1970–1981. Curran Associates, Inc., 2020.

Chongkai Gao, Zhengrong Xue, Shuying Deng, Tianhai Liang, Siqi Yang, Lin Shao, and Huazhe Xu. RiEMann: Near real-time SE(3)-equivariant robot manipulation without point cloud segmentation. In *8th Annual Conference on Robot Learning*, 2024.

Keld Helsgaun. *An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems: Technical report*. Roskilde Universitet, December 2017.

Yan Jin, Yuandong Ding, Xuanhao Pan, Kun He, Li Zhao, Tao Qin, Lei Song, and Jiang Bian. Pointerformer: Deep reinforced multi-pointer transformer for the traveling salesman problem. In *AAAI*, 2023.

Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35:1936–1949, 2022.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Ke Li, Fei Liu, Zhenkun Wang, and Qingfu Zhang. Destroy and repair using hyper-graphs for routing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18341–18349, 2025.

Yang Li, Jinpei Guo, Runzhong Wang, Hongyuan Zha, and Junchi Yan. Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. *Advances in Neural Information Processing Systems*, 37:30179–30206, 2024.

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs, 2022.

Yi-Lun Liao, Brandon M Wood, Abhishek Das, and Tess Smidt. Equiformerv2: Improved equivariant transformer for scaling to higher-degree representations, 2023.

Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36:8845–8864, 2023.

Fu Luo, Xi Lin, Zhenkun Wang, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Self-improved learning for scalable neural combinatorial optimization. *CoRR*, abs/2403.19561, 2024. URL https://arxiv.org/abs/2403.19561.

Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. *Advances in Neural Information Processing Systems*, 34:11096–11107, 2021.

Yimeng Min, Yiwei Bai, and Carla P Gomes. Unsupervised learning for solving the travelling salesman problem. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=lAEc7aIW20.

Wenbin Ouyang, Yisen Wang, Shaochen Han, Zhejian Jin, and Paul Weng. Improving generalization of deep reinforcement learning-based tsp solvers. In *IEEE SSCI ADPRL*, 2021.

Mingjun Pan, Guanquan Lin, You-Wei Luo, Bin Zhu, Zhien Dai, Lijun Sun, and Chun Yuan. Preference optimization for combinatorial optimization problems. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 2025.

Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. *Advances in neural information processing systems*, 36:3706–3731, 2023.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Thibaut Vidal. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Comput. Oper. Res.*, 140(C), April 2022. ISSN 0305-0548. doi: 10.1016/j.cor.2021.105643. URL https://doi.org/10.1016/j.cor.2021.105643.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

Ziming Wang and Rebecka Jörnsten. SE(3)-bi-equivariant transformers for point cloud assembly. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Maurice Weiler and Gabriele Cesa. *General E(2)-equivariant steerable CNNs*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Changliang Zhou, Xi Lin, Zhenkun Wang, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Instance-conditioned adaptation for large-scale generalization of neural routing solver. *IEEE Transactions on Intelligent Transportation Systems*, 2026.

A Technical details for rotation-equivariant layers

A.1 Block form of features and linear maps

The main text specifies a hidden feature by the multiplicity of each type-k irrep. Let $m_k^\star$ denote this multiplicity for $\star \in \{\mathrm{in}, \mathrm{out}\}$. The complete feature space and its explicit block action are

$$\mathcal{F}_\star = \mathbb{R}^{m_0^\star} \oplus \bigoplus_{k=1}^{K} (\mathbb{R}^2)^{\oplus m_k^\star},$$
$$\rho_\star(\varphi) = \boldsymbol{I}_{m_0^\star} \oplus \bigoplus_{k=1}^{K} \rho_k(\varphi)^{\oplus m_k^\star}, \qquad \star \in \{\mathrm{in}, \mathrm{out}\}. \tag{18}$$

The superscript $\oplus m$ denotes m repeated block-diagonal copies, so each type-k feature block is transformed by the same matrix $\rho_k(\varphi)$. The feature dimension is $d_\star = m_0^\star + 2\sum_{k>0} m_k^\star$.

Partition a coordinate-independent linear map $\boldsymbol{W} : \mathcal{F}_{\mathrm{in}} \to \mathcal{F}_{\mathrm{out}}$ into blocks $\boldsymbol{W}_{k\ell}$ from input type ℓ to output type k. Equivariance requires the intertwining condition

$$\boldsymbol{W}\rho_{\mathrm{in}}(\varphi) = \rho_{\mathrm{out}}(\varphi)\boldsymbol{W} \qquad \text{for every } \varphi. \tag{19}$$

For the real $\mathrm{SO}(2)$ irreps in Equation (5), this condition first implies

$$\boldsymbol{W}_{k\ell} = \boldsymbol{0} \quad (k \neq \ell), \qquad \boldsymbol{W}_{00} = \boldsymbol{A}_0. \tag{20}$$

For $k > 0$, let a index an input copy and b an output copy of type k. Every corresponding 2×2 subblock has the form

$$(\boldsymbol{W}_{kk})_{ba} = a_{ba}^{(k)} \boldsymbol{I}_2 + b_{ba}^{(k)} \boldsymbol{J}, \qquad \boldsymbol{J} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{21}$$

Table 5: Angular basis matrices $\boldsymbol{B}_{\nu}^{k\ell}(\phi)$ for $\mathrm{SO}(2)$-steerable kernels from input type ℓ to output type k satisfying Equation (23). Rows index output types and columns index input types. Adapted from Weiler & Cesa (2019).

Output $k \setminus$ input ℓ	Type 0	Type ℓ, $\ell \geq 1$	
Type 0	$[\,1\,]$	$[\,\cos(\ell\phi) \quad \sin(\ell\phi)\,]$,	$[\,-\sin(\ell\phi) \quad \cos(\ell\phi)\,]$
Type k, $k \geq 1$	$\begin{bmatrix}\cos(k\phi)\\ \sin(k\phi)\end{bmatrix}$,	$\begin{bmatrix}-\sin(k\phi)\\ \cos(k\phi)\end{bmatrix}$	$\begin{bmatrix}\cos((k-\ell)\phi) & -\sin((k-\ell)\phi)\\ \sin((k-\ell)\phi) & \cos((k-\ell)\phi)\end{bmatrix}$, $\begin{bmatrix}\cos((k+\ell)\phi) & \sin((k+\ell)\phi)\\ \sin((k+\ell)\phi) & -\cos((k+\ell)\phi)\end{bmatrix}$
			$\begin{bmatrix}-\sin((k-\ell)\phi) & -\cos((k-\ell)\phi)\\ \cos((k-\ell)\phi) & -\sin((k-\ell)\phi)\end{bmatrix}$, $\begin{bmatrix}-\sin((k+\ell)\phi) & \cos((k+\ell)\phi)\\ \cos((k+\ell)\phi) & \sin((k+\ell)\phi)\end{bmatrix}$

The learned coefficients $a_{ba}^{(k)}$ and $b_{ba}^{(k)}$ mix copies of the same type, while $\boldsymbol{J}$ is a fixed quarter-turn within each two-dimensional block. Thus a coordinate-independent linear layer cannot mix different types, and its bias can be nonzero only in the type-0 output channels. Its weight parameter count is $m_0^{\mathrm{out}} m_0^{\mathrm{in}} + 2\sum_{k>0} m_k^{\mathrm{out}} m_k^{\mathrm{in}}$, instead of the $d_{\mathrm{out}} d_{\mathrm{in}}$ parameters of an unconstrained matrix. Geometry-conditioned steerable kernels overcome the zero cross-type blocks by using the input coordinates as an additional transforming argument (Weiler & Cesa, 2019; Cesa et al., 2022).

A.2 Nonlinearities and normalization

Componentwise nonlinearities generally break equivariance because a rotation mixes the components of every type-k block with $k > 0$. For example, with $\boldsymbol{v} = (1, -1)^{\top}$, $\mathrm{ReLU}(\boldsymbol{R}_{\pi/2}\boldsymbol{v}) = (1, 1)^{\top}$ but $\boldsymbol{R}_{\pi/2}\,\mathrm{ReLU}(\boldsymbol{v}) = (0, 1)^{\top}$. A sufficient family of equivariant nonlinearities rescales each complete non-scalar block by an invariant scalar:

$$\boldsymbol{f}^{(k,a)} \longmapsto g_{k,a}(\boldsymbol{\iota}(\boldsymbol{f}))\,\boldsymbol{f}^{(k,a)}, \qquad k > 0, \tag{22}$$

where $\boldsymbol{\iota}(\boldsymbol{f})$ collects scalar channels and norms of complete non-scalar blocks. The gain $g_{k,a}$ may therefore depend on all invariant information at the node without selecting a preferred orientation. Scalar channels admit ordinary scalar nonlinearities, and normalization may center or rescale them directly. For $k > 0$, normalization instead uses invariant block statistics and applies one scalar gain to the entire two-component block, without an additive non-scalar bias.

A.3 Steerable kernels between different types

A coordinate-dependent kernel from input type ℓ to output type k must satisfy

$$\boldsymbol{K}^{k\ell}(\boldsymbol{R}_{\varphi}\boldsymbol{z}) = \rho_k(\varphi)\boldsymbol{K}^{k\ell}(\boldsymbol{z})\rho_{\ell}(\varphi)^{-1}. \tag{23}$$

For a nonzero coordinate $\boldsymbol{z} = r(\cos\phi, \sin\phi)^{\top}$, let $\boldsymbol{A}_{\nu}^{k\ell} = \boldsymbol{B}_{\nu}^{k\ell}(0)$ be constant reference matrices at polar angle zero. The corresponding angle-dependent basis matrices and kernel expansion are

$$\begin{aligned}\boldsymbol{B}_{\nu}^{k\ell}(\phi) &= \rho_k(\phi)\boldsymbol{A}_{\nu}^{k\ell}\rho_{\ell}(\phi)^{-1},\\ \boldsymbol{K}^{k\ell}(\boldsymbol{z}) &= \sum_{\nu} w_{\nu}^{k\ell}(r)\boldsymbol{B}_{\nu}^{k\ell}(\phi).\end{aligned} \tag{24}$$

The reference matrices $\boldsymbol{A}_{\nu}^{k\ell}$ span maps between the two irrep spaces, while the learned coefficients $w_{\nu}^{k\ell}$ depend only on the invariant radius. Substituting $\phi + \varphi$ into the definition of $\boldsymbol{B}_{\nu}^{k\ell}$ gives Equation (23) directly. Table 5 lists the resulting angular basis matrices for all scalar and non-scalar input–output type combinations.

The lower-left cell gives the radial and tangential directions for scalar-to-type-k lifting. For two non-scalar types, the four basis matrices split into angular modes $k - \ell$ and $k + \ell$.

A.4 Invariant attention and policy composition

Queries and keys in one attention head use matching orthogonal representations. Their dot product is invariant because

$$\langle \rho(\varphi)\boldsymbol{q}, \rho(\varphi)\boldsymbol{k} \rangle = \boldsymbol{q}^\top \rho(\varphi)^\top \rho(\varphi)\boldsymbol{k} = \boldsymbol{q}^\top \boldsymbol{k}. \tag{25}$$

Invariant distances and any geometry-independent feasibility mask therefore leave the attention logits and softmax weights α_{ij} unchanged. If the values transform equivariantly, the attended output does as well:

$$\sum_j \alpha_{ij} \rho_V(\varphi)\boldsymbol{v}_j = \rho_V(\varphi) \sum_j \alpha_{ij}\boldsymbol{v}_j. \tag{26}$$

This is the planar counterpart of the invariant-weight construction used by the SE(3)-Transformer (Fuchs et al., 2020). Equivariant residual blocks, norm-based nonlinearities, and this attention operation remain equivariant under composition. An invariant scoring rule then produces identical logits for corresponding rotated states. Equality of the action distributions does not force independently sampled rollouts to make identical choices; samplewise equality requires coupled randomness.

A.5 The zero-radius case

The angular formula in Equation (24) assumes $\boldsymbol{z} \neq \boldsymbol{0}$. At the origin every rotation fixes the kernel coordinate, so covariance reduces to

$$\boldsymbol{K}(\boldsymbol{0})\rho_{\text{in}}(\varphi) = \rho_{\text{out}}(\varphi)\boldsymbol{K}(\boldsymbol{0}) \qquad \text{for every } \varphi. \tag{27}$$

Thus $\boldsymbol{K}(\boldsymbol{0})$ must itself have the coordinate-independent block form in Equation (21); assigning an arbitrary numerical angle to $\boldsymbol{z} = \boldsymbol{0}$ does not satisfy this condition. For a nondegenerate instance with a joint continuous coordinate distribution, a node lying exactly at the centroid is a probability-zero event, and our symmetry statement uses this general-position assumption.

The distinction still matters when the local input is nonzero at that coordinate. A CVRP customer can have $\boldsymbol{z}_i = \boldsymbol{0}$ but positive demand, so an arbitrary scalar-to-vector angle would create a preferred direction. A DRHG endpoint can have $\boldsymbol{z}_i = \boldsymbol{0}$ while its paired coordinate $\boldsymbol{z}_{\tau(i)}$ is nonzero, allowing the same issue in a vector-to-scalar or vector-to-vector kernel. A coordinate-only homogeneous lift instead vanishes at the origin. The numerical floor in Equation (11) stabilizes the scale but does not define a valid angular kernel there.

B Routing baselines and their equivariant adaptations

B.1 POMO's attention-based policy

The POMO baseline considered here uses an AM-style attention encoder (Kool et al., 2019) and the multiple-start reinforcement-learning procedure of Kwon et al. (2020). For TSP, AM embeds each coordinate with an affine map, $\boldsymbol{h}_i^0 = \boldsymbol{W}_x\boldsymbol{x}_i + \boldsymbol{b}_x$. For CVRP, the customer embedding instead receives $[\boldsymbol{x}_i \oplus q_i/Q]$, while the depot has a separate affine embedding. Let $\boldsymbol{H}^\ell \in \mathbb{R}^{N\times d}$ stack the node embeddings at encoder layer ℓ. An ordinary attention head computes

$$\text{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}} + \boldsymbol{M}\right)\boldsymbol{V}, \tag{28}$$

where $\boldsymbol{M}$ is zero on allowed pairs and $-\infty$ on masked pairs. Multi-head attention applies Equation (28) to separately projected queries, keys, and values and combines the head outputs with a learned projection. The baseline encoder applies self-attention, identity skip connections, batch normalization, and a pointwise feed-forward network as written in Equation (1). Because its affine embeddings and attention projections are unconstrained, the original encoder does not prescribe how its hidden features transform under a coordinate rotation.

The final node embeddings $\boldsymbol{h}_i$ are computed once, and their mean $\bar{\boldsymbol{h}} = N^{-1}\sum_i \boldsymbol{h}_i$ provides a graph-level summary to every rollout. After fixing the first TSP node a_1, the decoder uses the first and current endpoints; for CVRP, it replaces the first-node endpoint with the invariant remaining-capacity ratio:

$$\boldsymbol{c}_t = \begin{cases} [\bar{\boldsymbol{h}} \oplus \boldsymbol{h}_{a_t} \oplus \boldsymbol{h}_{a_1}], & \text{TSP}, \\ [\bar{\boldsymbol{h}} \oplus \boldsymbol{h}_{a_t} \oplus c_t^{\text{rem}}/Q], & \text{CVRP}. \end{cases} \tag{29}$$

At the depot, the CVRP current-node embedding is the depot embedding. The context is projected to a query, attends to the encoded nodes to form a glimpse, and is converted to clipped pointer scores:

$$\boldsymbol{q}_t = \boldsymbol{W}_c \boldsymbol{c}_t, \qquad \boldsymbol{g}_t = \mathrm{MHA}(\boldsymbol{q}_t, \boldsymbol{H}, \boldsymbol{H}),$$
$$u_{ti} = C \tanh\left(\frac{(\boldsymbol{W}_g \boldsymbol{g}_t)^\top \boldsymbol{k}_i^{\mathrm{ptr}}}{\sqrt{d_k}}\right), \qquad p_\theta(i \mid s_t, \mathcal{I}) = \mathrm{softmax}_{i \in \mathcal{A}(s_t)}(u_{ti}). \tag{30}$$

The pointer keys $\boldsymbol{k}_i^{\mathrm{ptr}}$ are learned projections of $\boldsymbol{h}_i$; the action mask removes visited nodes and, for CVRP, customers whose demands exceed the remaining capacity.

POMO evaluates M decoder rollouts over the shared node embeddings. TSP assigns a different first node to each rollout, whereas CVRP begins at the depot and assigns different first customers. These forced choices are not sampled from the policy and therefore do not contribute to the rollout log-probability sum. For reward $R^{(m)} = -C(S^{(m)}; \mathcal{I})$ and within-instance baseline $\bar{R} = M^{-1} \sum_m R^{(m)}$, the policy-gradient estimator for maximizing expected reward is

$$\widehat{\nabla_\theta J} = \frac{1}{M} \sum_{m=1}^{M} \big(R^{(m)} - \bar{R}\big) \sum_t \nabla_\theta \log p_\theta(a_t^{(m)} \mid s_t^{(m)}, \mathcal{I}). \tag{31}$$

At inference, the lowest-cost completed rollout is returned.

Eq-POMO keeps the encoder–decoder construction and multiple-start objective. Coordinate lifting replaces the affine input map; feature fibers replace unconstrained hidden channels; and equivariant linear maps and norm-based operations replace the corresponding ordinary layers. Its attention residual uses the learned equivariant projection Skip^ℓ, in Equation (1), while its feed-forward residual adds the block input directly. Its query uses the graph summary and the two endpoint embeddings, with the depot serving as the CVRP anchor. The pointer distance bias is computed in the encoder's normalized frame in both single-start decoding and the cached multiple-start path. Positive uniform rescaling of an instance therefore leaves this bias unchanged when the normalization floor is inactive; route costs are still evaluated in the original coordinate system.

B.2 DRHG'S PAIRED INPUTS AND REPAIR DECODER

DRHG improves an incumbent solution by removing selected edges and solving the resulting reduced repair problem (Li et al., 2025). The interiors of intact route segments are omitted temporarily, while their two endpoints are retained and paired. Write $\tau(i)$ for the partner of reduced node i, with $\tau(i) = i$ for an isolated node, and let $e_i \in \{0, 1\}$ indicate an endpoint of a nontrivial segment. Selecting i deterministically traverses the retained segment to $\tau(i)$; the next learned decision reconnects that endpoint to another reduced node. Up to channel order, the original learned inputs are

$$\boldsymbol{r}_i^{\mathrm{TSP}} = [\boldsymbol{x}_i \oplus \boldsymbol{x}_{\tau(i)} \oplus e_i], \qquad \boldsymbol{r}_i^{\mathrm{CVRP}} = [\boldsymbol{x}_i \oplus \boldsymbol{x}_{\tau(i)} \oplus e_i \oplus \widetilde{q}_i / Q], \tag{32}$$

Here $\widetilde{q}_i$ is an isolated customer's demand or an intact segment's aggregate demand. An ordinary affine map $\boldsymbol{h}_i^0 = \boldsymbol{W}_{\mathrm{in}} \boldsymbol{r}_i + \boldsymbol{b}_{\mathrm{in}}$ embeds each reduced node.

At decoding step t, let $\mathcal{U}_t$ be the selectable reduced nodes and let f_t and c_t be the first and current nodes of the repaired partial route. The TSP implementation creates one projection of $\boldsymbol{h}_{f_t}^0$ and sixteen independent projections of $\boldsymbol{h}_{c_t}^0$. The CVRP implementation creates depot and current-node representatives, each conditioned on c_t^{rem}/Q before projection. Writing these route representatives as $\boldsymbol{O}_t^0$, the combined token sequence is

$$\boldsymbol{I}_t^0 = \begin{cases} [\boldsymbol{O}_t^0 \,;\, (\boldsymbol{h}_i^0)_{i \in \mathcal{U}_t}], & \mathrm{TSP}, \\ [\boldsymbol{o}_{t,\mathrm{depot}}^0 \,;\, (\boldsymbol{h}_i^0)_{i \in \mathcal{U}_t} \,;\, \boldsymbol{o}_{t,\mathrm{current}}^0], & \mathrm{CVRP}. \end{cases} \tag{33}$$

Thus TSP uses seventeen route representatives, whereas CVRP uses two; in either case their number is independent of $|\mathcal{U}_t|$.

Each decoder layer first lets the route representatives aggregate from all tokens, $\boldsymbol{O} \leftarrow \boldsymbol{I}$, and then lets all tokens attend to the updated representatives, $\boldsymbol{I} \leftarrow \boldsymbol{O}$. Neither cross-attention direction uses an attention mask in the original decoder. For either direction, the original residual block is

$$\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{A}, \qquad \mathcal{B}_{\mathrm{ord}}(\boldsymbol{X}, \boldsymbol{A}) = \boldsymbol{Y} + \mathrm{FFN}(\boldsymbol{Y}), \tag{34}$$

where $\boldsymbol{A}$ is the multi-head attention output after the learned head-combination projection. The feed-forward network uses componentwise ReLU, and the original decoder applies no normalization in this block. Because the number of route representatives is fixed, the two cross-attention operations cost $O(r|\mathcal{U}_t|)$ per layer rather than forming all candidate–candidate attention pairs.

Only candidate rows of $\boldsymbol{I}_t^L$ are scored:

$$u_{ti} = \boldsymbol{w}^\top \boldsymbol{I}_{t,i}^L + b \quad (\text{TSP}), \qquad (u_{ti}^{\text{direct}}, u_{ti}^{\text{depot}}) = \boldsymbol{W}_{\text{out}} \boldsymbol{I}_{t,i}^L \quad (\text{CVRP}). \tag{35}$$

A softmax over unselected candidate/action pairs gives the next learned decision, with route-context tokens excluded. The original CVRP decoder does not apply a capacity-feasibility mask to these logits. Forced partner traversals and the final restoration of omitted segment interiors are handled outside the network. For a reference repair action a_t^* on reduced instance $\mathcal{H}$, supervised training minimizes

$$\ell_t(\theta) = -\log p_\theta(a_t^* \mid s_t, \mathcal{H}), \tag{36}$$

with forced and padded steps excluded. The original procedure performs an optimizer update after each learned step; Appendix B.3 details the rollout-level update used by Eq-DRHG.

Our equivariant replacement keeps this token organization and the forced endpoint transitions. It lifts the two coordinates as vector features, retains demand as a scalar, uses equivariant cross-attention, and reads out scalar fields and squared norms. The repair controller still uses the endpoint flag e_i and the paired-node map, including $\tau(i) = i$ for isolated nodes. We omit e_i only from the learned lift; it remains available for forced transitions and for excluding those steps from the loss. Decoder normalization is applied after every attention residual and every feed-forward residual. With $\mathcal{N}_1, \mathcal{N}_2$ denoting those normalizations, the post-attention map in Equation (16) is explicitly

$$\mathcal{B}(u) = \mathcal{N}_2(\mathcal{N}_1(u) + \text{FFN}(\mathcal{N}_1(u))) \,. \tag{37}$$

The CVRP remaining-capacity ratio modulates an equivariant query residual in each cross-attention direction. The Eq-CVRP decoder additionally masks capacity-infeasible direct and via-depot choices before its softmax.

B.3 Optimizer updates for reduced-problem training

For a batch of reduced instances, let $\mathcal{B}_t$ contain the examples with a learned, non-padding target at step t, and let $\mathcal{T} = \{t : |\mathcal{B}_t| > 0\}$. The implemented reduction is

$$\begin{aligned} \overline{\ell}_t(\theta) &= -\frac{1}{|\mathcal{B}_t|} \sum_{b \in \mathcal{B}_t} \log p_\theta(a_{b,t}^* \mid s_{b,t}, \mathcal{H}_b), \\ \mathcal{L}_{\text{repair}}(\theta) &= \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \overline{\ell}_t(\theta). \end{aligned} \tag{38}$$

Thus each valid decoding position contributes its batch mean, followed by an average over valid positions. This differs from pooling all valid example/position pairs with equal weight when $|\mathcal{B}_t|$ varies. The two given opening positions in TSP, the fixed opening position in CVRP, and forced or padded transitions do not create learned loss terms.

Eq-DRHG accumulates this gradient with parameters held fixed and makes one optimizer update after the batch of reduced-problem rollouts. In the original per-step procedure, each $\overline{\ell}_t$ is followed by an optimizer update, so later decisions are evaluated at different parameter values. Backpropagating accumulated losses in smaller windows does not change the update boundary. These schedules share the supervised prediction task but are not the same optimization trajectory. We therefore report the update schedule alongside the architecture when comparing convergence and training time.

B.4 Retained DRHG preprocessing

The original DRHG coordinate transformation is retained in both models for consistency with the baseline. It translates the reduced coordinates, rescales them by a bounding-box quantity, and can exchange the two axes. The subsequent centroid and unit-disk normalization removes translation and positive uniform scale. A conditional axis exchange is a separate operation: an input rotation can change whether that exchange occurs. Consequently, equality of the preprocessing procedures is

Table 6: Hyperparameter configurations of Eq-POMO and Eq-DRHG.

Hyperparameter	**Eq-POMO**	**Eq-DRHG**
Embedding type	{1: 64, 2: 64}	{1: 64, 2: 64}
Q/K type	{1: 32, 2: 32}	{1: 32, 2: 32}
Encoder layers	6	1
Decoder layers	1	6
Attention heads	8	8
Hidden dimension	1024	1024
Batch size	128	1024
Training samples/epoch	100,000	1,000,000
Algorithm	REINFORCE	Cross-entropy loss
Baseline	Shared	–
Augmentation	dihedral-8	–
Optimizer	Adam	Adam
Learning rate	1e−3	5e−3
Weight decay	1e−6	1e−6
LR scheduler	ExponentialLR	ExponentialLR
LR gamma	0.99	0.97
Max epochs	200	100

a comparison choice, rather than a proof that the composite raw-input pipeline commutes with every rotation. Our equivariance statement applies to the neural model acting on its geometric input and corresponding repair state. Tests of the complete solver must additionally account for the retained preprocessing and destruction rules.

C Additional experiment details

C.1 Hyper-parameter configuration

We list all the hyperparameters used for Eq-POMO and Eq-DRHG in Table 6.

C.2 Instance distributions

We consider various distributions in routing problems. Let $X = \{x_i\}_{i=0}^{n} \subset [0,1]^2$ denote the node coordinates; Specifically, let x_0 denote the depot and the other denote customers for CVRP. We also give a figure to demonstrate how the instances differ under various distributions in Figure 1.

Uniform. We denote a continuous uniform random variable by $X \sim U(a,b)$, where $a = 0, b = 1$ in our experiment. This assigns equal probability density throughout the unit square $[0,1]^2$ and therefore produces approximately homogeneous node distributions without an explicit spatial preference.

Normal. Coordinates are sampled from a single Gaussian distribution $X \sim N(\mu, \sigma^2)$, where μ is the mean and σ is the variance. The distribution is symmetric around μ, with σ controlling its spread. We set $\mu = 0.5, \sigma = 0.2$ in our experiment. When independently applied to the two coordinate dimensions, it produces a spatial concentration around a central location, with density decreasing smoothly away from the mean.

Cluster. Clustered instances are generated using multiple Gaussian components. Following the setting of Bi et al. (2022), we use $k = 3$ clusters, sample cluster center as $\mu_k \sim U([0.2, 0.8]^2)$, and sample nodes from isotropic Gaussians with $\sigma = 0.07$. This produces compact high-density regions separated by relatively sparse areas.

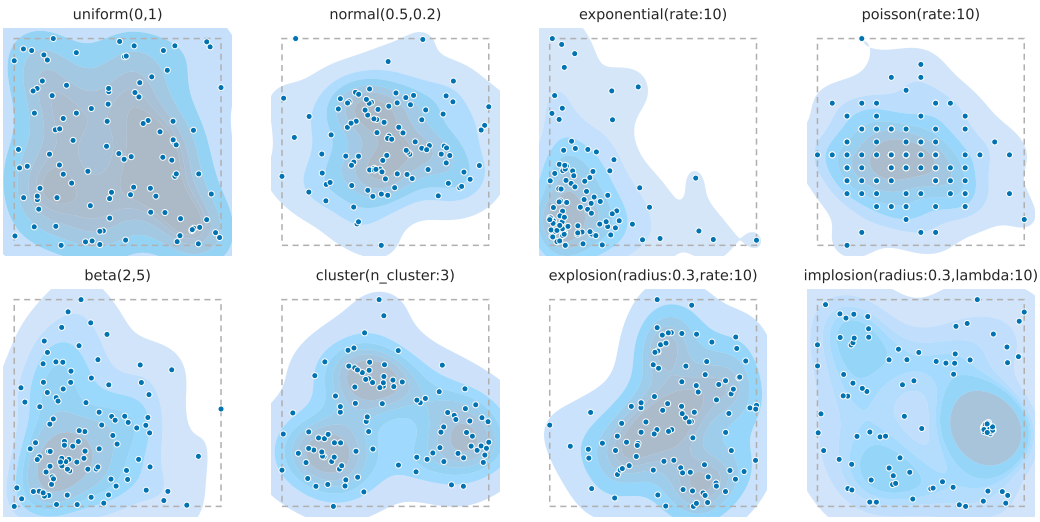

Figure 1: Illustrative instances for the Uniform, Normal, Exponential, Poisson, Beta, Cluster, Explosion, and Implosion distributions.

Beta. We denote a beta-distributed random variable by $X \sim Beta(\alpha, \beta)$, where $(\alpha, \beta) > 0$ are shape parameters. We set $\alpha = 2, \beta = 5$. Different choices of (α, β) can produce symmetric, skewed, center-concentrated, or boundary-concentrated spatial patterns.

Exponential. We denote an exponentially distributed random variable by $X \sim \mathrm{Exp}(\lambda)$, where $\lambda > 0$ is the rate parameter. We set $\lambda = 1$. The distribution places most probability mass near zero and has a long positive tail, producing a strongly skewed and non-uniform spatial pattern.

Poisson. We denote a poisson-distributed random variable by $X \sim \mathrm{Pois}(\lambda)$, where $\lambda > 0$ is the rate parameter. The Poisson distribution is discrete, with both mean and variance equal to λ. We set $\lambda = 1$. When applied to node-coordinate generation, it produces discrete spatial locations and can therefore introduce repeated or regularly quantized coordinate values.

Explosion. An explosion instance distribution is initialized by sampling all node coordinates uniformly from the unit square. A centroid is then sampled uniformly from the same square, and a circle of fixed radius r is placed around it. Every node inside the circle is moved outside while preserving its radial direction. Its new distance from the center is $r+\delta_i$, where $\delta_i \sim \mathrm{Exp}(\lambda)$. Following the setting in Bi et al. (2022), we set $r = 0.3, \lambda = 10$. Nodes outside the selected circle remain unchanged. The resulting distribution contains a sparse central region surrounded by outwardly displaced nodes.

Implosion. An implosion instance distribution also begins with uniformly sampled node coordinates. A centroid is selected uniformly, and a circle of fixed radius r is formed around it. Every node inside the circle is moved toward the centroid by dividing its radial distance by a multiplier λ. Following the setting in Bi et al. (2022), we set $r = 0.3, \lambda = 10$. Nodes outside the circle remain unchanged. Larger multiplier values produce stronger contraction toward the center.

C.3 Model generalization ability

C.4 TSPLIB and CVRPLIB Results

We give the detailed results on TSPLIB and CVRPLIB in Table 9 and Table 10.

Table 7: Cross-distribution & Cross-size generalization; trained on TSP100; evaluated each TSP problem with 128 instances.

Distribution	Uniform								Normal							
Method	TSP100		TSP200		TSP500		TSP1000		TSP100		TSP200		TSP500		TSP1000	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
POMO	7.79	0.00s	11.00	0.05s	19.21	0.32s	44.79	0.51s	6.88	0.00s	10.14	0.02s	18.25	0.24s	35.63	0.46s
Eq-POMO	7.78	0.01s	10.81	0.12s	18.43	0.72s	29.67	4.54s	6.83	0.01s	9.86	0.04s	17.33	0.58s	27.56	4.39s
DRHG	7.75	2.76s	10.70	5.43s	16.56	5.72s	23.18	5.80s	6.75	2.78s	9.66	5.47s	15.46	5.71s	22.05	5.76s
Eq-DRHG	7.75	6.98s	10.70	11.83s	16.56	12.25s	23.18	12.30s	6.75	6.68s	9.66	11.91s	15.46	12.32s	22.05	12.04s

Distribution	Cluster								Beta							
Method	TSP100		TSP200		TSP500		TSP1000		TSP100		TSP200		TSP500		TSP1000	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
POMO	3.81	0.00s	5.62	0.02s	9.98	0.24s	23.95	0.43s	5.06	0.00s	7.42	0.02s	12.95	0.24s	26.85	0.40s
Eq-POMO	3.73	0.01s	5.33	0.04s	9.42	0.58s	14.55	4.39s	5.01	0.01s	7.18	0.04s	12.43	0.58s	19.68	4.39s
DRHG	3.64	2.91s	5.11	5.63s	8.11	5.93s	11.55	6.12s	4.96	2.98s	7.04	5.66s	11.11	5.81s	15.73	5.92s
Eq-DRHG	3.64	6.98s	5.11	11.95s	8.11	12.75s	11.54	12.30s	4.96	9.07s	7.03	11.95s	11.11	12.75s	15.72	12.30s

Distribution	Exponential								Poisson							
Method	TSP100		TSP200		TSP500		TSP1000		TSP100		TSP200		TSP500		TSP1000	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
POMO	3.18	0.00s	4.66	0.02s	8.11	0.24s	21.51	0.60s	475.3	0.00s	979.5	0.02s	2483.3	0.24s	4330.3	0.70s
Eq-POMO	3.08	0.01s	4.49	0.04s	7.89	0.58s	11.99	4.39s	103.9	0.01s	141.1	0.04s	200.6	0.58s	249.8	4.39s
DRHG	2.97	2.66s	4.12	5.34s	6.49	5.58s	9.14	5.65s	103.2	2.68s	140.8	5.45s	226.8	5.70s	282.9	5.66s
Eq-DRHG	2.97	6.62s	4.12	11.25s	6.49	12.89s	9.13	12.88s	103.0	6.84s	140.4	12.10s	227.2	12.91s	283.6	12.30s

Distribution	Explosion								Implosion							
Method	TSP100		TSP200		TSP500		TSP1000		TSP100		TSP200		TSP500		TSP1000	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
POMO	6.53	0.00s	8.71	0.02s	14.19	0.24s	34.20	0.39s	7.05	0.00s	9.64	0.02s	16.20	0.24s	42.17	0.43s
Eq-POMO	6.52	0.01s	8.54	0.04s	13.83	0.58s	20.79	4.39s	7.06	0.01s	9.55	0.04s	15.65	0.58s	24.78	4.39s
DRHG	6.51	3.02s	8.41	5.70s	12.11	5.67s	16.03	5.88s	6.98	2.77s	9.36	5.45s	14.21	5.67s	19.51	5.98s
Eq-DRHG	6.51	7.09s	8.41	12.29s	12.11	12.77s	16.03	12.21s	6.98	6.98s	9.36	11.99s	14.22	12.72s	19.52	12.69s

Table 8: Cross-distribution & Cross-size generalization; trained on CVRP100; evaluated each CVRP problem with 128 instances. Times are reported per instance.

Distribution	Uniform								Normal							
Method	CVRP100		CVRP200		CVRP500		CVRP1000		CVRP100		CVRP200		CVRP500		CVRP1000	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
POMO	15.90	0.00s	23.37	0.07s	44.79	0.51s	78.43	3.01s	12.13	0.00s	18.27	0.03s	35.63	0.46s	75.04	3.51s
Eq-POMO	15.85	0.01s	22.97	0.09s	39.90	0.83s	57.67	5.20s	12.06	0.01s	17.91	0.06s	31.43	0.70s	47.89	5.18s
DRHG	15.48	9.27s	22.29	16.48s	37.74	21.91s	52.90	25.68s	11.93	9.64s	17.33	16.91s	30.10	21.98s	42.97	26.14s
Eq-DRHG	15.45	21.06s	22.27	32.88s	37.67	47.16s	53.00	61.49s	11.91	22.34s	17.32	33.93s	29.92	48.83s	42.68	60.44s

Distribution	Cluster								Beta							
Method	CVRP100		CVRP200		CVRP500		CVRP1000		CVRP100		CVRP200		CVRP500		CVRP1000	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
POMO	8.06	0.00s	11.83	0.03s	23.95	0.43s	45.85	3.28s	9.32	0.00s	14.07	0.03s	26.85	0.40s	49.21	3.21s
Eq-POMO	7.98	0.01s	11.47	0.06s	21.37	0.71s	30.13	5.25s	9.24	0.01s	13.74	0.06s	23.62	0.70s	35.79	5.16s
DRHG	7.57	9.74s	11.07	16.68s	19.22	22.95s	28.00	27.13s	8.99	9.44s	13.29	16.69s	22.21	22.25s	32.81	25.73s
Eq-DRHG	7.56	22.61s	11.05	35.43s	19.18	47.79s	27.70	59.91s	8.98	21.96s	13.26	35.64s	22.10	51.03s	32.51	59.11s

Distribution	Exponential								Poisson							
Method	CVRP100		CVRP200		CVRP500		CVRP1000		CVRP100		CVRP200		CVRP500		CVRP1000	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
POMO	5.65	0.01s	9.00	0.06s	21.51	0.60s	52.75	4.00s	887.4	0.01s	1726.1	0.06s	4330.3	0.70s	8418.0	4.47s
Eq-POMO	5.54	0.01s	8.38	0.06s	15.36	0.71s	24.47	5.19s	186.5	0.01s	264.9	0.06s	461.7	0.71s	632.2	5.20s
DRHG	5.54	12.30s	7.90	24.09s	13.48	32.74s	20.63	32.05s	183.9	9.20s	262.2	16.02s	820.9	20.45s	1675.2	23.33s
Eq-DRHG	5.53	23.81s	7.91	39.25s	13.61	59.63s	20.83	63.70s	183.0	21.59s	260.2	35.46s	816.0	49.45s	1678.2	55.90s

Distribution	Explosion								Implosion							
Method	CVRP100		CVRP200		CVRP500		CVRP1000		CVRP100		CVRP200		CVRP500		CVRP1000	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
POMO	13.64	0.00s	18.52	0.03s	34.20	0.39s	54.96	2.99s	15.38	0.00s	21.90	0.03s	42.17	0.43s	72.92	3.03s
Eq-POMO	13.60	0.01s	18.23	0.06s	31.02	0.71s	42.56	5.21s	15.42	0.01s	21.72	0.06s	38.45	0.77s	55.62	5.71s
DRHG	13.61	9.45s	17.62	16.24s	28.40	22.23s	37.64	25.95s	14.93	9.52s	21.00	16.51s	35.99	24.57s	50.34	29.48s
Eq-DRHG	13.59	20.88s	17.59	35.57s	28.32	47.48s	37.42	58.77s	14.92	20.68s	21.11	33.97s	36.52	49.99s	51.39	69.21s

Table 9: Per-instance results of Eq-POMO and Eq-DRHG on TSPLIB.

Instance	Eq-POMO	Eq-DRHG	Instance	Eq-POMO	Eq-DRHG
a280	2.8306%	0.0000%	berlin52	0.0000%	0.0000%
bier127	1.0965%	0.0000%	brd14051	OOM	2.8644%
ch130	0.2619%	0.0000%	ch150	0.5055%	0.0000%
d198	16.6286%	0.0000%	d493	41.7548%	0.3914%
d657	20.3365%	0.0859%	d1291	40.8870%	0.1831%
d1655	34.0635%	0.1803%	d2103	31.4543%	1.7850%
d15112	OOM	2.8770%	d18512	OOM	3.3110%
eil51	1.1737%	0.0000%	eil76	0.0000%	0.0000%
eil101	0.1590%	0.0000%	fl417	20.0067%	0.0084%
fl1400	31.2814%	1.0583%	fl1577	41.7861%	1.6315%
fl3795	51.0184%	3.4547%	fnl4461	OOM	1.0862%
gil262	1.6400%	0.0000%	kroA100	0.0000%	0.0000%
kroA150	0.1583%	0.0000%	kroA200	0.8819%	0.0000%
kroB100	0.6233%	0.0000%	kroB150	0.2067%	0.0077%
kroB200	1.0769%	0.0000%	kroC100	0.0096%	0.0000%
kroD100	0.2771%	0.0000%	kroE100	0.3127%	0.0000%
lin105	0.1669%	0.0000%	lin318	3.8283%	0.2712%
linhp318	5.5460%	1.6544%	nrw1379	23.9115%	0.8316%
p654	23.3929%	0.1241%	pcb442	9.3032%	0.0000%
pcb1173	28.3467%	0.6802%	pcb3038	34.0661%	0.6965%
pr76	0.0000%	0.0000%	pr107	0.7042%	0.0000%
pr124	0.2558%	0.0000%	pr136	1.0644%	0.0000%
pr144	0.5057%	0.0000%	pr152	1.4128%	0.0000%
pr226	1.5292%	0.0000%	pr264	4.6545%	0.0000%
pr299	2.8595%	0.0000%	pr439	8.8475%	0.0755%
pr1002	25.1435%	0.5424%	pr2392	40.1675%	0.7145%
rat99	0.1652%	0.0000%	rat195	2.8412%	0.2152%
rat575	14.8088%	0.1919%	rat783	20.1908%	0.3634%
rd100	0.0000%	0.0000%	rd400	6.3870%	0.0393%
rl1304	39.9216%	0.9543%	rl1323	36.8269%	0.5866%
rl1889	43.4744%	0.7525%	rl5915	OOM	1.9628%
rl5934	OOM	2.0083%	rl11849	OOM	3.5023%
st70	0.0000%	0.0000%	ts225	0.5748%	0.0000%
tsp225	3.8560%	0.0511%	u159	0.9529%	0.0000%
u574	13.0741%	0.3089%	u724	19.5061%	0.3269%
u1060	24.3135%	0.2619%	u1432	22.6933%	0.3850%
u1817	34.2739%	1.0489%	u2152	31.6608%	1.4458%
u2319	11.6471%	0.2873%	usa13509	OOM	7.9785%
vm1084	31.9369%	0.3464%	vm1748	39.2734%	0.4023%

Table 10: Per-instance results of Eq-POMO and Eq-DRHG on CVRPLIB.

Instance	Eq-POMO	Eq-DRHG	Instance	Eq-POMO	Eq-DRHG
A-n32-k5	0.5102%	8.5459%	A-n33-k5	1.6641%	1.5129%
A-n33-k6	0.1348%	10.2426%	A-n34-k5	1.2853%	0.8997%
A-n36-k5	0.8761%	8.2603%	A-n37-k5	0.5979%	0.2990%
A-n37-k6	1.1591%	7.4816%	A-n38-k5	2.1918%	0.5479%
A-n39-k5	1.2165%	7.4209%	A-n39-k6	0.6017%	2.4067%
A-n44-k6	0.4269%	1.7076%	A-n45-k6	2.9661%	3.0720%
A-n45-k7	1.3962%	3.3159%	A-n46-k7	1.9694%	7.1116%
A-n48-k7	3.8211%	4.0075%	A-n53-k7	3.6634%	6.1386%
A-n54-k7	2.3993%	3.9417%	A-n55-k9	2.2367%	11.7428%
A-n60-k9	1.1817%	5.5391%	A-n61-k9	3.7718%	4.2553%
A-n62-k8	1.3199%	8.6957%	A-n63-k9	1.4233%	6.5594%
A-n63-k10	1.5221%	5.4033%	A-n64-k9	2.5696%	5.7102%
A-n65-k9	0.9370%	3.5775%	A-n69-k9	1.0354%	2.5884%
A-n80-k10	1.9853%	4.7079%	B-n31-k5	2.0833%	3.8690%
B-n34-k5	3.5533%	5.4569%	B-n35-k5	3.0366%	11.2042%
B-n38-k6	3.6025%	9.4410%	B-n39-k5	0.3643%	8.1967%
B-n41-k6	1.9300%	7.2376%	B-n43-k6	1.0782%	4.5822%

Continued on the next page

Table 10 – continued

Instance	Eq-POMO	Eq-DRHG	Instance	Eq-POMO	Eq-DRHG
B-n44-k7	1.7602%	5.2805%	B-n45-k5	2.2636%	4.7936%
B-n45-k6	6.6372%	7.3746%	B-n50-k7	1.8893%	3.1039%
B-n50-k8	1.9817%	0.4573%	B-n51-k7	0.5814%	0.5814%
B-n52-k7	2.0080%	10.3079%	B-n56-k7	1.5559%	8.4866%
B-n57-k7	2.5152%	0.0000%	B-n57-k9	1.0638%	2.5657%
B-n63-k10	4.3449%	10.2941%	B-n64-k9	7.7816%	8.1301%
B-n66-k9	1.2158%	2.5076%	B-n67-k10	2.7132%	8.7209%
B-n68-k9	2.6730%	6.5252%	B-n78-k10	3.7674%	2.7846%
E-n22-k4	0.8000%	21.0667%	E-n23-k3	0.1757%	1.9332%
E-n30-k3	-3.9326%	-4.8689%	E-n33-k4	3.8323%	2.1557%
E-n51-k5	5.5662%	1.1516%	E-n76-k7	2.0528%	2.4927%
E-n76-k8	1.2245%	0.9524%	E-n76-k10	3.8554%	1.3253%
E-n76-k14	3.0362%	2.4486%	E-n101-k8	2.2086%	0.6135%
E-n101-k14	3.7488%	4.4049%	F-n45-k4	6.9061%	3.7293%
F-n72-k4	9.7046%	2.1097%	F-n135-k7	23.4940%	10.4991%
M-n101-k10	1.2195%	0.9756%	M-n121-k7	2.0309%	9.5745%
M-n151-k12	1.9704%	3.3498%	M-n200-k16	5.1020%	3.7677%
M-n200-k17	5.0196%	3.6863%	P-n16-k8	3.7778%	5.5556%
P-n19-k2	3.3019%	7.0755%	P-n20-k2	2.7778%	4.1667%
P-n21-k2	2.3697%	16.5877%	P-n22-k2	2.7778%	3.2407%
P-n22-k8	5.3068%	12.4378%	P-n23-k8	5.1040%	32.1361%
P-n40-k5	1.7467%	0.8734%	P-n45-k5	1.1765%	0.7843%
P-n50-k7	1.8051%	1.9856%	P-n50-k8	1.7433%	4.2789%
P-n50-k10	2.5862%	7.4713%	P-n51-k10	1.0796%	5.3981%
P-n55-k7	2.1127%	1.5845%	P-n55-k10	1.4409%	4.4669%
P-n55-k15	-3.4378%	6.9767%	P-n60-k10	2.8226%	7.6613%
P-n60-k15	1.2397%	8.1612%	P-n65-k10	1.0101%	1.5152%
P-n70-k10	1.9347%	3.7485%	P-n76-k4	4.3845%	3.5413%
P-n76-k5	5.1037%	2.2329%	P-n101-k4	4.9927%	1.0279%