EVOLVING NEURAL SOLVERS FOR MULTI-OBJECTIVE COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-objective combinatorial optimization (MOCO) seeks high-quality solutions that represent different trade-offs among conflicting objectives. Existing neural methods typically train a dedicated model for each preference or condition a single model on different preferences. Training dedicated models becomes more costly as the number of preferences increases, while a shared model can produce duplicate solutions for different preferences, limiting the benefit of denser preference queries. We propose the Evolving Neural Solver (ENS), which trains only a small set of preference-specific models and takes advantage of evolutionary computation to evolve them into a diverse collection of solvers for better solution quality and Pareto front coverage. ENS first trains a model at an estimated knee-point preference to target a balanced compromise, then obtains the remaining initial models through parameter transfer toward the extremes, so as to preliminarily approximate the Pareto front. It then evolves these models to find more candidates that can generate non-dominated solutions through evolution strategies. Experimental results on classic MOCO problems show that ENS outperforms state-of-the-art neural MOCO methods in terms of hypervolume in most evaluated settings.

1 INTRODUCTION

Many combinatorial optimization problems involve conflicting objectives. In vehicle routing, for example, minimizing total travel distance may conflict with minimizing the longest vehicle route. Similar trade-offs arise in scheduling and knapsack problems (Zajac & Huber, 2021; Türkyılmaz et al., 2020; Ishibuchi et al., 2014). Multi-objective combinatorial optimization (MOCO) therefore seeks Pareto-optimal solutions, each representing a trade-off that cannot be improved in one objective without worsening another. Since computing the complete Pareto set is often intractable, the practical goal is to obtain a high-quality approximation that represents a broad range of trade-offs.

Neural approaches to MOCO follow two main directions. Single-model methods condition one shared network on the preference vector, so that arbitrarily many preferences can be queried after a single training run (Lin et al., 2022; Wang et al., 2023; Fan et al., 2024; Chen et al., 2025a). A single policy, however, has to cover the entire preference simplex, and neighboring preferences frequently decode to identical solutions (Chen et al., 2023b), so querying more preferences does not necessarily enrich the resulting front. Multi-model methods avoid this ambiguity by assigning a dedicated policy to each preference (Li et al., 2020; Zhang et al., 2022; Chen et al., 2023a), but every such policy has to be obtained through reinforcement learning, which converges slowly and makes the acquisition of each additional solver inefficient.

Considering the strength of population-based search in evolutionary computation, we propose the Evolving Neural Solver (ENS), a two-stage framework that combines reinforcement learning with evolutionary computation. ENS trains only a small set of preference-specific models, termed *anchors*, and then expands them into a diverse collection of solvers. In the first stage, since the objectives of a MOCO problem differ in optimization difficulty and in magnitude, ENS starts from the estimated knee-point preference and trains the first anchor there. The remaining anchors are then obtained by successive transfer toward the extremes. This small set of anchors already provides a coarse approximation of the Pareto front. In the second stage, the anchors serve as the starting points of a population-based search, in which new solvers are repeatedly generated around

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.

the existing ones and retained only when they contribute a solution that is non-dominated among all candidates, thereby enriching the Pareto front.

The contributions of this paper are summarized as follows:

- We propose ENS, which expands a small set of preference-specific neural models through evolutionary search. Joint non-dominated selection across populations retains solvers that contribute different trade-offs, extending the solution set without separate training for every preference.
- We introduce an anchor placement and training strategy that starts from an estimated knee-point preference and transfers successively toward the extremes. Anchoring the limited training budget at the most critical region of the preference space makes the fullest use of each trained model and provides stronger starting points for the subsequent evolutionary search.

Experiments on classic MOCO problems show that the proposed framework outperforms state-of-the-art neural MOCO methods in terms of hypervolume in most evaluated settings.

2 Related Work

Conventional MOCO methods. Conventional MOCO methods include exact algorithms and heuristics. Exact algorithms provide optimality guarantees, but computing the complete Pareto set can become prohibitively expensive as problem size grows (Ehrgott, 2005). Evolutionary frameworks approximate the Pareto front through population search. NSGA-II uses non-dominated sorting and crowding distance (Deb et al., 2002), whereas MOEA/D coordinates scalarized subproblems (Zhang & Li, 2007). SPEA2 and PAES use archive-based mechanisms (Zitzler et al., 2001; Knowles & Corne, 2000), while NSGA-III and RVEA maintain diversity using reference points or vectors (Deb & Jain, 2014; Cheng et al., 2016). MOEA/D-DE and GDE3 incorporate differential evolution (Li & Zhang, 2008; Kukkonen & Lampinen, 2005), and MOPSO employs particle-swarm search (Coello et al., 2004). These general-purpose frameworks can be adapted to discrete problems through suitable representations and operators. Genetic and Pareto local search further exploit problem structure (Jaszkiewicz, 2002; Shi et al., 2022). Although effective, these methods generally require substantial search for each new instance, motivating learned policies that reuse experience across instances.

Neural MOCO methods. Neural combinatorial optimization was initially developed around single-objective problems. Pointer Networks learn sequential solution construction (Vinyals et al., 2015), while the attention model of Kool et al. (2019) uses Transformer-based representations (Vaswani et al., 2017) and reinforcement learning. POMO improves training through multiple starting points and a shared reward baseline (Kwon et al., 2020). Subsequent work improves representations and generalization (Wang et al., 2025; Gao et al., 2024; Zhou et al., 2026; Meng et al., 2025; 2026), with further advances in routing policies and multi-task learning (Li et al., 2023; Zhou et al., 2024; Liao et al., 2025).

For MOCO, single-model methods share one network across preferences. Lin et al. (2022) uses a preference-conditioned hypernetwork to generate decoder parameters, whereas Wang et al. (2023) directly conditions the policy on objective weights. Fan et al. (2024) extends conditioning to support generalization across preferences and problem sizes. More recent methods improve preference representations through weight embeddings (Chen et al., 2025a) and enhance instance representations through graph-image multimodal fusion (Chen et al., 2025b). POCCO introduces conditional computation and pairwise preference optimization (Fan et al., 2026), while WeCon strengthens weight conditioning in both the encoder and decoder (Wu et al., 2026). Multi-model methods instead obtain specialized policies for different preferences. Li et al. (2020) combines scalarization with parameter transfer between neighboring preferences, while meta-learning methods learn initializations that support efficient adaptation (Zhang et al., 2022; Chen et al., 2023a). Neural evolutionary methods also optimize learned policies: Zhang et al. (2021) evolves model parameters and retains a fixed-size population using non-dominated sorting and crowding distance, while Shao et al. (2021) combines neural learning with evolutionary optimization for better performance.

3 PRELIMINARIES

3.1 MULTI-OBJECTIVE COMBINATORIAL OPTIMIZATION

Problem Definition. A MOCO problem is defined by a discrete feasible set $\mathcal{X}$, whose elements are combinatorial structures such as permutations or subsets induced by the problem constraints, together with $m \geq 2$ objective functions $f_i : \mathcal{X} \to \mathbb{R}$. It minimizes the objective vector:

$$\min_{x \in \mathcal{X}} \mathbf{f}(x) = \big(f_1(x), \ldots, f_m(x)\big). \tag{1}$$

Pareto Dominance and Optimality. A solution x_1 dominates x_2, denoted by $x_1 \prec x_2$, if $f_i(x_1) \leq f_i(x_2)$ for all i and the inequality is strict for at least one objective. A feasible solution is Pareto-optimal if it is not dominated by any feasible solution. The set of Pareto-optimal solutions is the Pareto set $\mathcal{X}^*$, and its image $\{\mathbf{f}(x) : x \in \mathcal{X}^*\}$ is the Pareto front.

Extreme Points and Knee Point. For each objective i, let $x^{(i)}$ be a Pareto-optimal solution among the minimizers of f_i alone, $x^{(i)} \in \arg\min_{x \in \mathcal{X}} f_i(x), x^{(i)} \in \mathcal{X}^*$, and call $\mathbf{f}(x^{(i)})$ the i-th *extreme point* of the Pareto front, namely the objective vector attained when objective i alone is optimized. Let H be the hyperplane through the m extreme points and $\hat{\mathbf{n}}$ its unit normal pointing in the direction of decreasing objectives. Following Das & Dennis (1996); Das (1999), the signed distance from H and the knee solution $x^\dagger$ are defined as:

$$b(x) = \hat{\mathbf{n}}^\top \big(\mathbf{f}(x) - \mathbf{p}\big), \qquad x^\dagger \in \arg\max_{x \in \mathcal{X}^*} b(x), \tag{2}$$

where $\mathbf{p} \in H$ is arbitrary and $b(x)$ is independent of its choice. The objective vector $\mathbf{f}(x^\dagger)$ is the *knee point*, the Pareto-optimal trade-off that improves most upon the linear interpolation of the m extreme points.

3.2 PREFERENCE SCALARIZATION AND NEURAL CONSTRUCTION

For a preference vector $\boldsymbol{\lambda} \in \Delta^{m-1}$, where $\Delta^{m-1} = \{\boldsymbol{\lambda} \in \mathbb{R}^m_{\geq 0} : \sum_{i=1}^m \lambda_i = 1\}$, weighted-sum scalarization minimizes $\sum_{i=1}^m \lambda_i f_i(x)$. A common alternative is Tchebycheff scalarization, which minimizes $\max_i \lambda_i |f_i(x) - z_i^*|$. For strictly positive weights, any global optimum of the weighted-sum problem is Pareto-optimal, although weighted sums do not necessarily recover the entire Pareto front.

For a fixed preference, a neural constructive solver builds a solution $x = (a_1, \ldots, a_T)$ sequentially for an instance s:

$$\pi_\theta(x \mid s) = \prod_{t=1}^{T} \pi_\theta(a_t \mid a_{1:t-1}, s), \tag{3}$$

where a_t is the action at step t and T is the number of construction steps. Under weighted-sum scalarization, π_θ is trained to maximize $\mathbb{E}_{s \sim \mathcal{D},\, x \sim \pi_\theta(\cdot|s)}[-\sum_{i=1}^m \lambda_i f_i(x)]$. Thus, serving J target preferences with dedicated policies requires training or fine-tuning J models.

4 METHODOLOGY

4.1 OVERVIEW AND SOLVER BACKBONE

Figure 1 summarizes the ENS pipeline. ENS proceeds in two stages. In the first stage, ENS estimates the knee-point preference of the problem, trains one anchor there from scratch, and obtains the remaining $M-1$ anchors by successive transfer toward the extreme preferences, so that a small number of trained models already cover the preference space coarsely. In the second stage, each anchor becomes the starting point of a population that evolves in a low-dimensional subspace of its decoder. Candidates from all anchors are evaluated jointly, and only those contributing a non-dominated solution are retained, which extends the initial M anchors into a much larger collection of solvers. At inference, every retained solver is rolled out on the test instance, and the non-dominated solutions

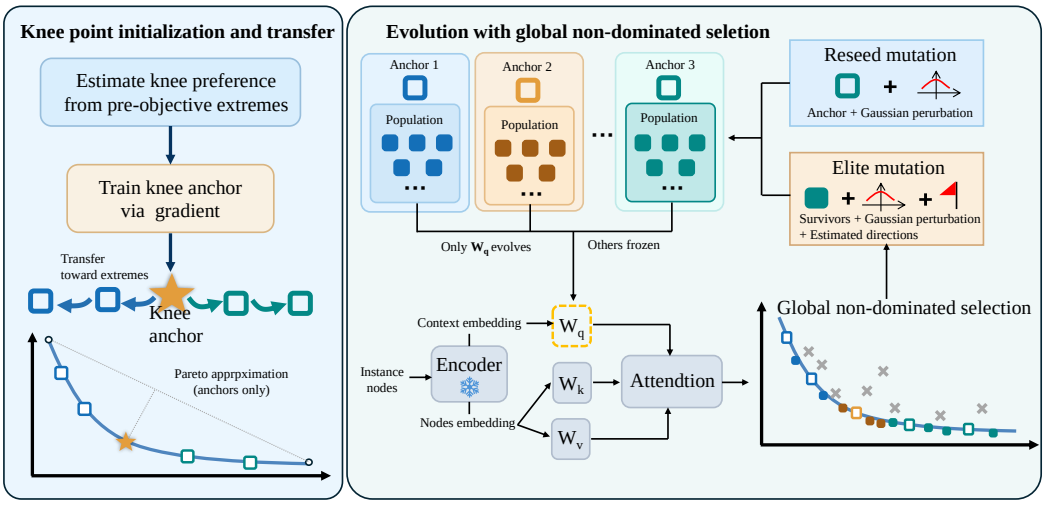


Figure 1: Overview of ENS

among all produced trajectories form the final Pareto front approximation, without requiring any preference as input.

Every solver in ENS shares the same encoder-decoder backbone adopted from POMO (Kwon et al., 2020), and they differ only in their parameters. The encoder maps an instance s to node embeddings $h_1, \ldots, h_n \in \mathbb{R}^d$, which depend only on the instance and are computed once per rollout. The decoder holds three linear projections $W_q, W_k, W_v \in \mathbb{R}^{d\times d}$. W_k and W_v are applied once to all node embeddings and cached, while W_q is applied at every decoding step t to the context embedding h_c formed from the current decoding state:

$$q_t = W_q\, h_c, \qquad k_i = W_k\, h_i, \qquad v_i = W_v\, h_i. \tag{4}$$

Multi-head attention over (q_t, k, v), followed by a learned output projection, is then dotted with the node embeddings to give the next-node probabilities:

$$u_{t,i} = C \cdot \tanh\Big(\frac{\mathrm{MHA}(q_t, k, v)^\top h_i}{\sqrt{d}}\Big), \qquad \pi_\theta(a_t{=}i \mid a_{1:t-1}, s) = \frac{\exp(u_{t,i})}{\sum_j \exp(u_{t,j})}, \tag{5}$$

where C is the tanh clipping constant.

4.2 Stage One: Knee-Point Initialization and Transfer

Locating the knee point. Estimating the knee-point preference requires the per-objective extreme points $\mathbf{f}(x^{(i)})$. At an extreme preference the MOCO problem reduces to a single-objective problem, which we solve with a strong heuristic or exact method for the problem at hand on a small number of sampled instances, averaging the resulting objective vectors into one representative extreme point per objective (per-problem details are given in Appendix D). Let $\mathbf{z}^*$ denote the ideal point, $z_i^* = f_i(x^{(i)})$, and let $\mathbf{e} = (1, \ldots, 1)^\top$. The extreme points and the ideal point form the payoff matrix:

$$\Phi = \begin{pmatrix} f_1(x^{(1)}) - z_1^* & f_1(x^{(2)}) - z_1^* & \cdots & f_1(x^{(m)}) - z_1^* \\ f_2(x^{(1)}) - z_2^* & f_2(x^{(2)}) - z_2^* & \cdots & f_2(x^{(m)}) - z_2^* \\ \vdots & \vdots & \ddots & \vdots \\ f_m(x^{(1)}) - z_m^* & f_m(x^{(2)}) - z_m^* & \cdots & f_m(x^{(m)}) - z_m^* \end{pmatrix}, \tag{6}$$

whose i-th column $\mathbf{f}(x^{(i)}) - \mathbf{z}^*$ records how far the i-th extreme solution stands from the ideal point on every objective. Das (1999) shows that the preference vector $\boldsymbol{\lambda}^\dagger$ associated with the knee point is the normalized solution of the $m \times m$ linear system:

$$\Phi^\top \boldsymbol{\omega} = \mathbf{e}, \qquad \boldsymbol{\lambda}^\dagger = \boldsymbol{\omega} / \textstyle\sum_{i=1}^m \omega_i. \tag{7}$$

Anchor placement. All anchors are positioned relative to $\boldsymbol{\lambda}^\dagger$, so that none of them sits at a fixed geometric location. For a bi-objective problem, the segment between $\boldsymbol{\lambda}^\dagger$ and each extreme preference is divided into $(M+1)/2$ equal sub-intervals, where M is the number of anchors. The sub-interval adjacent to $\boldsymbol{\lambda}^\dagger$ is absorbed into the region that the knee anchor itself already represents, and each of the remaining sub-intervals receives one anchor at its midpoint. Implementation details are described in Appendix D.

Transfer training. Under a fixed preference $\boldsymbol{\lambda}$, the weighted-sum scalarized cost of a solution is $g(x \mid \boldsymbol{\lambda}) = \boldsymbol{\lambda}^\top \mathbf{f}(x)$, and an anchor with parameters θ is trained to minimize its expectation,

$$\mathcal{L}(\theta \mid \boldsymbol{\lambda}) = \mathbb{E}_{s\sim\mathcal{D}}\, \mathbb{E}_{x\sim\pi_\theta(\cdot|s)} \big[g(x \mid \boldsymbol{\lambda}) \big], \tag{8}$$

with REINFORCE following the POMO training paradigm (Kwon et al., 2020), in which a rollout starts from N different first nodes and the resulting N trajectories serve as baselines for one another. Only the knee anchor is trained from a random initialization for E_0 epochs,

$$\theta^\dagger = \arg\min_\theta \; \mathcal{L}(\theta \mid \boldsymbol{\lambda}^\dagger). \tag{9}$$

The remaining anchors are then trained one at a time, outwards from the knee point. Each anchor c is warm-started from the anchor nearest to it among those already trained, $\theta^{(c)} \leftarrow \theta^{(c')}$, and adapted to its own preference $\boldsymbol{\lambda}^{(c)}$ for E_1 epochs, so that in the bi-objective case the anchors propagate along a chain on either side of the knee anchor. Such a warm start converges in far fewer epochs than training every anchor from scratch.

4.3 Stage Two: Evolutionary Extension

The number of parameters in a full solver is far beyond an evolution algorithm can search efficiently, since its sample cost grows rapidly with the dimension of the search space. The subspace that is evolved should therefore be as small as possible while still being the part that determines a solver's trade-off. In a complete solver, the encoder extracts the geometric and demand features of an instance, a representation that is only weakly related to the objective weighting, so the encoder is kept frozen throughout this stage. Within the decoder, a change of preference amounts to a change in what kind of node the current state should attend to, which in the attention mechanism corresponds directly to the semantic space of the query, W_q. The other two projections play a different role: W_k maps the node embeddings into the key space and mainly performs a fixed feature alignment, while W_v only aggregates information once the attention weights have been determined. We therefore take W_q as the individual that undergoes evolution, and keep all other weights frozen. The evolutionary extension repeats three steps over G generations: solution generation, solution evaluation, and searching direction update. Algorithm 1 in Appendix F gives the complete procedure.

Solution generation. We flatten $W_q \in \mathbb{R}^{d\times d}$ into a vector $\mathbf{w} \in \mathbb{R}^D$ with $D = d^2$, which serves as an individual of the evolutionary population, while the encoder and all other decoder weights are inherited unchanged from the corresponding anchor. Starting from the M anchor vectors $\{\mathbf{w}_c\}_{c=1}^M$, each anchor maintains its own population. Offspring are generated in one of two ways, which play different roles. Reseed offspring mutate the anchor vector itself and keep exploring broadly around the anchor:

$$\mathbf{w}'_{\text{reseed}} = \mathbf{w}_c + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I), \tag{10}$$

whereas elite offspring mutate a survivor of the previous generation, drawn cyclically from S_c, so as to further refine solvers that have already proven competitive:

$$\mathbf{w}'_{\text{elite}} = \mathbf{w}_{\text{parent}} + \boldsymbol{\epsilon} + \alpha\, \boldsymbol{\delta}_i^c, \qquad \mathbf{w}_{\text{parent}} \in S_c,\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I),\ i \in \{1, \dots, m\}, \tag{11}$$

where elite offspring are split evenly among the m search directions $\boldsymbol{\delta}_1^c, \dots, \boldsymbol{\delta}_m^c$ maintained by anchor c. In the first generation there are no survivors yet, so all offspring are generated from the anchor; from the second generation onward, offspring are allocated between the two ways in the ratio $r : (1-r)$. Together with the survivors, they form the candidate pool P_c of the current generation.

Solution evaluation. A candidate is not scored by a single fitness value, but by a small set of representative points. A discrete set of target preferences is fixed in advance, and each of them is

assigned to its nearest anchor, so that every anchor c owns a set Λ_c of preferences for which its candidates are held responsible. Each candidate $\mathbf{w}' \in P_c$ is reshaped back into W_q, substituted into the otherwise frozen backbone, and rolled out on the same batch of U instances, which returns N trajectories per instance.

One representative point is then computed for each preference $\boldsymbol{\lambda} \in \Lambda_c$ in two steps. On every instance, the N trajectories may have mutually non-dominated objective vectors, so one of them is selected as the trajectory with the lowest scalarized cost $g(x \mid \boldsymbol{\lambda})$, a step we call the *internal selection*. The objective vectors of the U selected trajectories, one per instance, are then averaged, which gives the representative point $\bar{\mathbf{f}}(\mathbf{w}' \mid \boldsymbol{\lambda}) \in \mathbb{R}^m$ and reduces the evaluation noise of a single instance. Repeating this over Λ_c describes the candidate by $|\Lambda_c|$ representative points, which summarize how it performs across the region of the preference space its anchor is responsible for.

Selection is then made over all candidates at once. A candidate may possess from 1 to $|\Lambda_c|$ distinct representative points. The representative points of every candidate are pooled into a single set. A candidate survives into the next generation if it has at least one globally non-dominated representative point. Finally, each survivor of anchor c is assigned to the group $\mathcal{B}_i^c$ of whichever objective i its non-dominated points improve upon the anchor on, and a survivor may belong to several groups.

Searching direction update. The groups $\mathcal{B}_i^c$ provide the signal for updating the search directions, which follow the search-gradient estimator of natural evolution strategies (Wierstra et al., 2014; Schaul et al., 2011). For a group indicator $\mathbb{1}(\boldsymbol{\epsilon}) \in \{0, 1\}$, marking whether an offspring perturbed by $\boldsymbol{\epsilon}$ falls into the group, the gradient of the expected group membership with respect to the parent vector is:

$$\nabla_{\mathbf{w}} \, \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)} \big[\mathbb{1}(\boldsymbol{\epsilon}) \big] = \frac{1}{\sigma^2} \, \mathbb{E} \big[\mathbb{1}(\boldsymbol{\epsilon}) \, \boldsymbol{\epsilon} \big], \tag{12}$$

which is estimated from the current generation by the mean perturbation of the offspring in that group. Each direction is then refreshed by an exponential moving average:

$$\boldsymbol{\delta}_i^c \leftarrow \beta \, \boldsymbol{\delta}_i^c + (1 - \beta) \, \frac{\bar{\boldsymbol{\epsilon}}_i^c}{\|\bar{\boldsymbol{\epsilon}}_i^c\|}, \qquad \bar{\boldsymbol{\epsilon}}_i^c = \frac{1}{|\mathcal{B}_i^c|} \sum_{k \in \mathcal{B}_i^c} \boldsymbol{\epsilon}_k, \tag{13}$$

so that the elite offspring of the next generation are pushed towards the region where survivors have recently been found on that side of the anchor. After G generations, each anchor's surviving population is merged with its own vector, giving the final set of solvers.

5 Experiment

5.1 Experiment Settings

Problems. We evaluate the multi-objective traveling salesman problem (MOTSP) (Lust & Teghem, 2010) in its bi-objective (Bi-TSP) and tri-objective (Tri-TSP) forms, the multi-objective capacitated vehicle routing problem (MOCVRP) (Zajac & Huber, 2021) in its bi-objective form (Bi-CVRP), and the multi-objective knapsack problem (MOKP) (Ishibuchi et al., 2014) in its bi-objective form (Bi-KP); full problem definitions appear in Appendix A. Bi-TSP, Bi-CVRP, and Tri-TSP are each tested at sizes $n = 20/50/100$, and Bi-KP at $n = 50/100/200$.

Hyperparameters. Most of our hyperparameters follow prior work (Kwon et al., 2020; Lin et al., 2022). For the bi-objective problems, we use $M = 5$ anchors, except for Bi-TSP100, Bi-CVRP100 and Bi-KP200, where $M = 7$ anchors are used; for Tri-TSP we use $M = 16$ anchors, except for Tri-TSP100, where $M = 28$. The knee anchor is trained from scratch for $E_0 = 200$ epochs, each epoch drawing 100,000 randomly sampled instances, while every other anchor is warm-started from its nearest trained neighbor and adapted for $E_1 = 100$ epochs when $M = 5$, or $E_1 = 50$ epochs otherwise. All anchors are optimized with Adam (Kinga et al., 2015) using a learning rate of 10^{-4} and a weight decay of 10^{-6}, and a rollout starts from $N = n$ different first nodes following POMO. In the evolutionary stage, each anchor generates $K = 1{,}000$ offspring per generation, with a reseed ratio of $r = 0.5$, an EMA rate of $\beta = 0.9$, and evaluates them on a batch of $U = 100$ sampled instances, for a total of $G = 100$ generations.

Baselines. We compare ENS against three groups of representative methods. The single-model group consists of **PMOCO** (Lin et al., 2022) and the weight-embedding method **WE** (Chen et al., 2025a), for which we report both of its variants, **WE-Add** and **WE-CA**. The multi-model group consists of **DRL-MOA** (Li et al., 2020), **MDRL** (Zhang et al., 2022) and **EMNH** (Chen et al., 2023a). The non-learnable group covers general-purpose evolutionary algorithms, **MOEA/D** (Zhang & Li, 2007) and **NSGA-II** (Deb et al., 2002), both run for 4,000 iterations; evolutionary algorithms tailored to MOCO, **MOGLS** (Jaszkiewicz, 2002), run for 4,000 iterations with 100 local search steps per iteration, and **PPLS/D-C** (Shi et al., 2022), run for 200 generations; **WS-LKH** and **WS-DP**, which combine weighted-sum scalarization with the strong LKH heuristic (Helsgaun, 2000; Tinós et al., 2018) for MOTSP and with a dynamic programming solver for MOKP. Two further single-model methods, **MORAM** (Wang et al., 2023) and the size-generalizing **CNH** (Fan et al., 2024), are included only in the generalization experiments of Appendix E. Every neural baseline is trained under the same weighted-sum scalarization as ENS to keep the comparison fair. All experiments are conducted on a server equipped with two NVIDIA RTX 4090 GPUs.

Metrics. We evaluate solution quality with the normalized hypervolume (HV) (While et al., 2006) indicator, where a larger value indicates a Pareto front that is both closer to the true front and more diverse; its definition, together with the reference and normalization point used for each problem and size, is given in Appendix B. For every method we report the average HV, its gap relative to the ENS variant on that problem, and the total time needed to solve 200 test instances. Results obtained with instance augmentation (Appendix C), which we apply to every problem except MOKP, are marked with the suffix "-Aug". In all tables, the best and the second-best results in each column are highlighted in **bold** and underlined, respectively, and tied results are marked in the same way.

5.2 Results and Analyses

Table 1 reports results on Bi-TSP, Bi-CVRP, Bi-KP and Tri-TSP. ENS attains the best HV on ten of the twelve problem-size combinations, the two exceptions being Bi-TSP100, where the near-exact WS-LKH remains 0.24% ahead, and Bi-KP200, where EMNH leads by 0.0002. On Bi-TSP100 ENS-Aug still surpasses every neural baseline, augmented or not, and on Bi-TSP20 ENS alone (0.6292) already exceeds every augmented baseline, including WE-Add-Aug (0.6272). On Bi-CVRP, ENS-Aug leads the strongest single-model baseline WE-CA-Aug at every size, and on Bi-KP, ENS alone surpasses every baseline at $n = 50$ and $n = 100$.

The margin of ENS widens with the number of objectives. On Tri-TSP, ENS-Aug attains the best hypervolume at every size, leading the strongest neural baseline by 1.40%, 2.12% and 1.68% at $n = 20/50/100$, and it is the only learning-based method that surpasses WS-LKH on Tri-TSP100 (0.5122 against 0.5076). With three objectives, the Pareto front is a surface rather than a curve, so a fixed set of preferences covers it far more thinly than in the bi-objective case; the evolutionary stage, which admits any candidate that contributes a non-dominated solution, is not tied to such a set and therefore benefits the most exactly where the front is hardest to cover.

Generalization to larger sizes and to real-world instances is reported in Appendix E.

5.3 Ablation Study

Effect of evolution and of the evolved weights. Table 2 reports the ablation results on Bi-TSP50. Without the evolutionary stage, the trained anchors alone lose 2.13% of hypervolume, more than twice the largest gap among the choices of evolved weights. Evolving W_q is the best choice, and evolving all three projections jointly is no better, although it triples the dimension of the search space: the query projection is the smallest subspace that carries the decision preference, so enlarging the search beyond it costs more in sample efficiency than the added expressiveness returns.

Qualitative comparison of the resulting fronts. Figure 2 shows the fronts obtained on one test instance of each ablation. In Figure 2a, the anchors alone produce a sparse and fragmented front, with solutions clustered around the few preferences they were trained for, whereas evolution turns it into a dense and continuous one, which is the per-instance picture behind the 2.13% gap above.

Figure 2b compares the two placement rules on Bi-CVRP50. Uniform anchors spread their budget evenly and leave a sparser front, thin in particular around the knee, which is where the front bends

Table 1: Results on Bi-TSP, Bi-CVRP, Bi-KP and Tri-TSP.

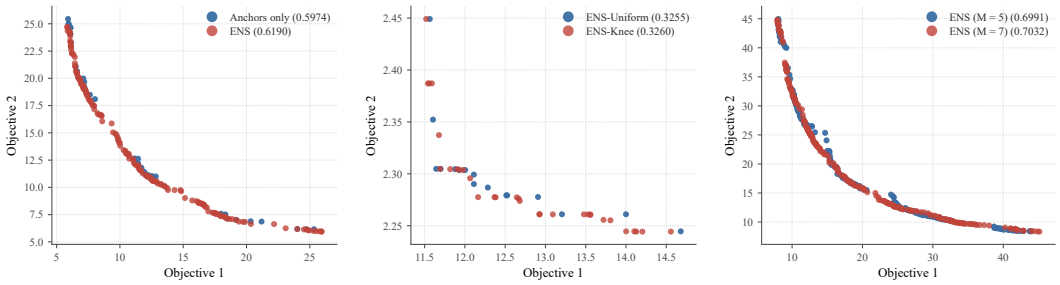
		Bi-TSP20			Bi-TSP50			Bi-TSP100		
	Method	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
Non-learnable	WS-LKH	0.6270	0.44%	10m	0.6415	0.31%	1.8h	**0.7090**	**-0.24%**	6.0h
	MOEA/D	0.6241	0.91%	1.7h	0.6316	1.85%	1.8h	0.6899	2.46%	2.2h
	NSGA-II	0.6258	0.64%	6.0h	0.6120	4.90%	6.1h	0.6692	5.39%	6.9h
	MOGLS	0.6279	0.30%	1.6h	0.6330	1.63%	3.7h	0.6854	3.10%	11h
	PPLS/D-C	0.6256	0.67%	26m	0.6282	2.38%	2.8h	0.6844	3.24%	11h
Single-model	PMOCO	0.6259	0.62%	6s	0.6351	1.31%	10s	0.6957	1.64%	19s
	WE-Add	0.6270	0.44%	6s	0.6386	0.76%	10s	0.7024	0.69%	19s
	WE-CA	0.6270	0.44%	6s	0.6391	0.68%	10s	0.7039	0.48%	20s
Multi-model	DRL-MOA	0.6257	0.65%	6s	0.6360	1.17%	9s	0.6970	1.46%	21s
	MDRL	0.6271	0.43%	5s	0.6364	1.10%	9s	0.6969	1.47%	17s
	EMNH	0.6271	0.43%	5s	0.6364	1.10%	9s	0.6969	1.47%	16s
	ENS	0.6292	0.10%	2s	0.6392	0.67%	8s	0.7029	0.62%	45s
Single-model-Aug	PMOCO-Aug	0.6270	0.44%	1.1m	0.6395	0.62%	3.2m	0.7016	0.81%	15m
	WE-Add-Aug	0.6272	0.41%	1.1m	0.6411	0.37%	3.2m	0.7058	0.21%	15m
	WE-CA-Aug	0.6271	0.43%	1.1m	0.6412	0.36%	3.3m	0.7067	0.08%	15m
Multi-model-Aug	MDRL-Aug	0.6271	0.43%	33s	0.6408	0.42%	1.7m	0.7022	0.72%	14m
	EMNH-Aug	0.6271	0.43%	33s	0.6408	0.42%	1.7m	0.7023	0.71%	14m
	ENS-Aug	**0.6298**	**0.00%**	1.8m	**0.6435**	**0.00%**	7.5m	0.7073	0.00%	36m
		Bi-CVRP20			Bi-CVRP50			Bi-CVRP100		
	Method	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
Non-learnable	MOEA/D	0.4255	1.21%	2.3h	0.4000	2.65%	2.9h	0.3953	3.28%	5.0h
	NSGA-II	0.4275	0.74%	6.4h	0.3896	5.18%	8.8h	0.3620	11.43%	9.4h
	MOGLS	0.4278	0.67%	9.0h	0.3984	3.04%	20h	0.3875	5.19%	72h
	PPLS/D-C	0.4287	0.46%	1.6h	0.4007	2.48%	9.7h	0.3946	3.45%	38h
Single-model	PMOCO	0.4267	0.93%	7s	0.4036	1.78%	12s	0.3913	4.26%	27s
	WE-Add	0.4292	0.35%	6s	0.4089	0.49%	12s	0.4061	0.64%	26s
	WE-CA	0.4293	0.33%	6s	0.4090	0.46%	12s	0.4072	0.37%	25s
Multi-model	DRL-MOA	0.4287	0.46%	10s	0.4076	0.80%	12s	0.4055	0.78%	33s
	MDRL	0.4291	0.37%	8s	0.4082	0.66%	13s	0.4056	0.76%	32s
	EMNH	0.4299	0.19%	7s	0.4098	0.27%	13s	0.4072	0.37%	31s
	ENS	0.4297	0.23%	3s	0.4097	0.29%	6s	0.4078	0.22%	14s
Single-model-Aug	PMOCO-Aug	0.4294	0.30%	14s	0.4080	0.71%	36s	0.3969	2.89%	2.7m
	WE-Add-Aug	0.4300	0.16%	14s	0.4103	0.15%	38s	0.4079	0.20%	2.5m
	WE-CA-Aug	0.4301	0.14%	14s	0.4104	0.12%	37s	0.4082	0.12%	2.5m
Multi-model-Aug	MDRL-Aug	0.4294	0.30%	11s	0.4092	0.41%	36s	0.4072	0.37%	2.8m
	EMNH-Aug	0.4302	0.12%	11s	0.4106	0.07%	35s	0.4079	0.20%	2.8m
	ENS-Aug	**0.4307**	**0.00%**	8s	**0.4109**	**0.00%**	34s	**0.4087**	**0.00%**	1.8m
		Bi-KP50			Bi-KP100			Bi-KP200		
	Method	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
Non-learnable	WS-DP	0.3561	0.14%	22m	0.4532	0.11%	2.0h	0.3601	0.00%	5.8h
	MOEA/D	0.3540	0.73%	1.6h	0.4508	0.64%	1.7h	0.3581	0.56%	1.8h
	NSGA-II	0.3547	0.53%	7.8h	0.4520	0.37%	8.0h	0.3590	0.31%	8.4h
	MOGLS	0.3540	0.73%	5.8h	0.4510	0.60%	10h	0.3582	0.53%	18h
	PPLS/D-C	0.3528	1.07%	18m	0.4480	1.26%	47m	0.3541	1.67%	1.5h
Single-model	PMOCO	0.3552	0.39%	9s	0.4523	0.31%	19s	0.3595	0.17%	1.3m
	WE-Add	0.3558	0.22%	8s	0.4530	0.15%	20s	0.3600	0.03%	1.1m
	WE-CA	0.3561	0.14%	9s	0.4533	0.09%	19s	0.3601	0.00%	1.1m
Multi-model	DRL-MOA	0.3559	0.20%	9s	0.4531	0.13%	18s	0.3601	0.00%	1.0m
	MDRL	0.3530	1.01%	6s	0.4532	0.11%	21s	0.3601	0.00%	1.2m
	EMNH	0.3561	0.14%	6s	0.4535	0.04%	21s	**0.3603**	**-0.06%**	1.2m
	ENS	**0.3566**	**0.00%**	7s	**0.4537**	**0.00%**	44s	0.3601	0.00%	3m
		Tri-TSP20			Tri-TSP50			Tri-TSP100		
	Method	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
Non-learnable	WS-LKH	0.4712	1.40%	12m	0.4440	1.92%	1.9h	0.5076	0.90%	6.6h
	MOEA/D	0.4702	1.61%	1.9h	0.4314	4.71%	2.2h	0.4511	11.93%	2.4h
	NSGA-II	0.4238	11.32%	7.1h	0.2858	36.87%	7.5h	0.2824	44.87%	9.0h
	MOGLS	0.4701	1.63%	1.5h	0.4211	6.98%	4.1h	0.4254	16.95%	13h
	PPLS/D-C	0.4698	1.69%	1.4h	0.4174	7.80%	3.9h	0.4376	14.56%	14h
Single-model	PMOCO	0.4693	1.80%	5s	0.4315	4.68%	8s	0.4858	5.15%	18s
	WE-Add	0.4705	1.55%	6s	0.4379	3.27%	10s	0.4943	3.49%	20s
	WE-CA	0.4707	1.51%	5s	0.4389	3.05%	9s	0.4976	2.85%	19s
Multi-model	DRL-MOA	0.4699	1.67%	6s	0.4303	4.95%	9s	0.4806	6.17%	19s
	MDRL	0.4699	1.67%	5s	0.4317	4.64%	9s	0.4852	5.27%	16s
	EMNH	0.4699	1.67%	5s	0.4324	4.48%	9s	0.4866	5.00%	16s
	ENS	0.4719	1.26%	5s	0.4317	4.64%	8s	0.4937	3.61%	27s
Single-model-Aug	PMOCO-Aug	0.4712	1.40%	5.1m	0.4409	2.61%	28m	0.4956	3.24%	1.7h
	WE-Add-Aug	0.4712	1.40%	5.1m	0.4429	2.16%	29m	0.5016	2.07%	1.8h
	WE-CA-Aug	0.4712	1.40%	5.2m	0.4431	2.12%	29m	0.5036	1.68%	1.7h
Multi-model-Aug	MDRL-Aug	0.4712	1.40%	2.6m	0.4408	2.63%	25m	0.4958	3.20%	1.7h
	EMNH-Aug	0.4712	1.40%	2.6m	0.4418	2.41%	25m	0.4973	2.91%	1.7h
	ENS-Aug	**0.4779**	**0.00%**	4.6m	**0.4527**	**0.00%**	25m	**0.5122**	**0.00%**	3.3h

and contributes the most hypervolume per solution; knee-based placement concentrates the solutions exactly there.

Figure 2c varies the number of anchors on Bi-TSP100. Since a single query projection has a limited expressive range, the region an anchor can reach through evolution is bounded, and on larger instances that range no longer suffices: with $M = 5$ the front dents at the boundaries between neighboring anchors, and each dent costs hypervolume, while $M = 7$ fills them. Anchors therefore do more than supply good starting points, as they also compete with one another through the global non-dominated selection, so denser anchors improve the quality of the front.

Table 2: Effect of evolution and of the evolved weights on Bi-TSP50.

Method	HV↑	Gap↓	Time↓
Anchors only	0.6256	2.13%	3s
Evolve W_q (ours)	**0.6392**	**0.00%**	8s
Evolve W_k	0.6379	0.20%	7s
Evolve W_v	0.6335	0.89%	8s
Evolve W_q, W_k, W_v	0.6378	0.22%	7s



(a) Effect of evolution (b) Effect of anchor placement (c) Effect of the number of anchors

Figure 2: Pareto fronts obtained on a single test instance. (a) Bi-TSP50, the anchors alone against the full ENS; (b) Bi-CVRP50, anchors placed uniformly against anchors placed relative to the estimated knee point; (c) Bi-TSP100, $M = 5$ against $M = 7$ anchors.

6 Conclusion

The goal of this paper was to effectively solve MOCO problems with high-quality solutions and broad coverage of the Pareto front. This goal has been successfully achieved by proposing the Evolving Neural Solver (ENS), which combines reinforcement learning with evolutionary computation. Specifically, ENS first trains a small set of anchors, starting from an estimated knee-point preference and transferring successively toward the extremes, so that the limited training budget is concentrated on the most critical region of the preference space. It then evolves the decoder query projections of these anchors and retains every candidate that contributes a non-dominated solution, expanding a few trained models into a diverse collection of solvers. Comparisons with strong neural and conventional methods show that ENS achieves the best hypervolume in most evaluated settings, and on the three-objective TSP it is the only learning-based method that surpasses WS-LKH. Further analyzes verify each component's effectiveness and the evolved solvers' ability to generalize to larger sizes and real-world instances.

Beyond the problems studied here, we believe that the paradigm embodied by ENS, in which learning provides strong initial solvers and evolution extends them into a diverse population, can transfer to a broader range of combinatorial optimization problems and to other domains like intelligent control and multi-agent systems.

AI USE STATEMENT

We used a large language model to help polish the writing of this paper, including improving clarity and wording in the text.

REFERENCES

Jinbiao Chen, Jiahai Wang, Zizhen Zhang, Zhiguang Cao, Te Ye, and Siyuan Chen. Efficient meta neural heuristic for multi-objective combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:56825–56837, 2023a.

Jinbiao Chen, Zizhen Zhang, Zhiguang Cao, Yaoxin Wu, Yining Ma, Te Ye, and Jiahai Wang. Neural multi-objective combinatorial optimization with diversity enhancement. *Advances in Neural Information Processing Systems*, 36:39176–39188, 2023b.

Jinbiao Chen, Zhiguang Cao, Jiahai Wang, Yaoxin Wu, Hanzhang Qin, Zizhen Zhang, and Yue-Jiao Gong. Rethinking neural multi-objective combinatorial optimization via neat weight embedding. In *International Conference on Learning Representations*, volume 2025, pp. 48510–48529, 2025a.

Jinbiao Chen, Jiahai Wang, Zhiguang Cao, and Yaoxin Wu. Neural multi-objective combinatorial optimization via graph-image multimodal fusion. In *International Conference on Learning Representations*, volume 2025, pp. 33964–33983, 2025b.

Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791, 2016.

Carlos A Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, 2004.

I Das and JE Dennis. Normal-boundary intersection: A new method for generating pareto-optimal points in multieriteria optimization problems. *SIAM J. Optimiz*, 1996.

Indraneel Das. On characterizing the "knee" of the pareto curve based on normal-boundary intersection. *Structural optimization*, 18(2):107–115, 1999.

Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, 2002.

Matthias Ehrgott. *Multicriteria Optimization*. Springer, 2 edition, 2005.

Mingfeng Fan, Yaoxin Wu, Zhiguang Cao, Wen Song, Guillaume Sartoretti, Huan Liu, and Guohua Wu. Conditional neural heuristic for multiobjective vehicle routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 36(3):4677–4689, 2024.

Mingfeng Fan, Jianan Zhou, Yifeng Zhang, Yaoxin Wu, Jinbiao Chen, and Guillaume Sartoretti. Preference-driven multi-objective combinatorial optimization with conditional computation. *Advances in Neural Information Processing Systems*, 38:11594–11627, 2026.

Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. In *International Joint Conference on Artificial Intelligence*, pp. 6914–6922, 2024.

Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European journal of operational research*, 126(1):106–130, 2000.

Hisao Ishibuchi, Naoya Akedo, and Yusuke Nojima. Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. *IEEE Transactions on Evolutionary Computation*, 19(2):264–283, 2014.

Andrzej Jaszkiewicz. Genetic local search for multi-objective combinatorial optimization. *European journal of operational research*, 137(1):50–71, 2002.

Diederik Kinga, Jimmy Ba Adam, et al. A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Joshua D Knowles and David W Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary computation*, 8(2):149–172, 2000.

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

Saku Kukkonen and Jouni Lampinen. Gde3: The third evolution step of generalized differential evolution. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pp. 443–450. IEEE, 2005.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302, 2008.

Jingwen Li, Yining Ma, Zhiguang Cao, Yaoxin Wu, Wen Song, Jie Zhang, and Yeow Meng Chee. Learning feature embedding refiner for solving vehicle routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(11):15279–15291, 2023.

Kaiwen Li, Tao Zhang, and Rui Wang. Deep reinforcement learning for multiobjective optimization. *IEEE Transactions on Cybernetics*, 51(6):3103–3114, 2020.

Zijun Liao, Jinbiao Chen, Debing Wang, Zizhen Zhang, and Jiahai Wang. BOPO: Neural combinatorial optimization via best-anchored and objective-guided preference optimization. In *Forty-second International Conference on Machine Learning*, 2025.

Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*, 2022.

Thibaut Lust and Jacques Teghem. The multiobjective traveling salesman problem: A survey and a new approach. In *Advances in Multi-Objective Nature Inspired Computing*, pp. 119–141. Springer, 2010.

Dian Meng, Zhiguang Cao, Yaoxin Wu, Yaqing Hou, Hongwei Ge, and Qiang Zhang. Eformer: an effective edge-based transformer for vehicle routing problems. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, 2025.

Dian Meng, Zhiguang Cao, Jie Gao, Yaoxin Wu, and Yaqing Hou. Uniteformer: unifying node and edge modalities in transformers for vehicle routing problems. *Advances in Neural Information Processing Systems*, 38:93830–93854, 2026.

Tom Schaul, Tobias Glasmachers, and Jürgen Schmidhuber. High dimensions and heavy tails for natural evolution strategies. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 845–852, 2011.

Yinan Shao, Jerry Chun-Wei Lin, Gautam Srivastava, Dongdong Guo, Hongchun Zhang, Hu Yi, and Alireza Jolfaei. Multi-objective neural evolutionary algorithm for combinatorial optimization problems. *IEEE Transactions on Neural Networks and Learning Systems*, 34(4):2133–2143, 2021.

Jialong Shi, Jianyong Sun, Qingfu Zhang, Haotian Zhang, and Ye Fan. Improving pareto local search using cooperative parallelism strategies for multiobjective combinatorial optimization. *IEEE Transactions on Cybernetics*, 54(4):2369–2382, 2022.

Renato Tinós, Keld Helsgaun, and Darrell Whitley. Efficient recombination in the lin-kernighan-helsgaun traveling salesman heuristic. In *International Conference on Parallel Problem Solving from Nature*, pp. 95–107. Springer, 2018.

Alper Türkyılmaz, Özlem Şenvar, Irem Ünal, and Serol Bulkan. A research survey: heuristic approaches for solving multi objective flexible job shop problems. *Journal of Intelligent Manufacturing*, 31(8):1949–1983, 2020.

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.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in Neural Information Processing Systems*, 28, 2015.

Yang Wang, Ya-Hui Jia, Wei-Neng Chen, and Yi Mei. Distance-aware attention reshaping for enhancing generalization of neural solvers. *IEEE Transactions on Neural Networks and Learning Systems*, 36(10):18900–18914, 2025.

Zhenkun Wang, Shunyu Yao, Genghui Li, and Qingfu Zhang. Multiobjective combinatorial optimization using a single deep reinforcement learning model. *IEEE Transactions on Cybernetics*, 54(3):1984–1996, 2023.

Lyndon While, Philip Hingston, Luigi Barone, and Simon Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, 2006.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Xuan Wu, Jinbiao Chen, Yang Li, Lijie Wen, Chunguo Wu, Yuanshu Li, Yubin Xiao, Chunyan Miao, You Zhou, and Di Wang. Wecon: An efficient weight-conditioned neural solver for multi-objective combinatorial optimization problems. *arXiv preprint arXiv:2605.22876*, 2026.

Sandra Zajac and Sandra Huber. Objectives and methods in multi-objective routing problems: a survey and classification scheme. *European Journal of Operational Research*, 290(1):1–25, 2021.

Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

Yongxin Zhang, Jiahai Wang, Zizhen Zhang, and Yalan Zhou. Modrl/d-el: Multiobjective deep reinforcement learning with evolutionary learning for multiobjective optimization. In *2021 International Joint Conference on Neural Networks*, pp. 1–8. IEEE, 2021.

Zizhen Zhang, Zhiyuan Wu, Hang Zhang, and Jiahai Wang. Meta-learning-based deep reinforcement learning for multiobjective optimization problems. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10):7978–7991, 2022.

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*, 27(7):8270–8284, 2026.

Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Xu Chi. MVMoe: Multi-task vehicle routing solver with mixture-of-experts. In *Forty-first International Conference on Machine Learning*, 2024.

Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK report*, 103, 2001.

A Problem Definition

MOTSP. We consider the MOTSP with n nodes, where each node carries m pairs of 2D coordinates, one per objective. Let $L_i(x)$ denote the total length of tour x measured under the i-th coordinate system. Given a preference $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m) \in \Delta^{m-1}$, MOTSP is scalarized as minimizing the preference-weighted sum of tour lengths across all coordinate systems:

$$\min_x \sum_{i=1}^{m} \lambda_i L_i(x). \tag{14}$$

MOCVRP. We consider the Bi-CVRP with n customer nodes and one depot, served by a fleet of vehicles, each with capacity Q. Let $\mathrm{TD}(x)$ denote the total routing distance of solution x and $\mathrm{MS}(x)$ denote its makespan, i.e., the length of the single longest route. Given a preference $\boldsymbol{\lambda} = (\lambda_1, \lambda_2) \in \Delta^1$, MOCVRP is scalarized as:

$$\min_x \ \lambda_1 \mathrm{TD}(x) + \lambda_2 \mathrm{MS}(x). \tag{15}$$

MOKP. We consider the Bi-KP with n items and a shared capacity B, where a solution x selects a feasible subset of items with total weight at most B. Let $V_i(x)$ denote the total value of x under the i-th value assignment, $i = 1, 2$. Given a preference $\boldsymbol{\lambda} = (\lambda_1, \lambda_2) \in \Delta^1$, MOKP is scalarized as:

$$\max_x \ \lambda_1 V_1(x) + \lambda_2 V_2(x). \tag{16}$$

B Hypervolume Indicator

HV is a standard indicator for evaluating a set of Pareto-optimal solutions, since it jointly measures both convergence to the true Pareto front and diversity along it. Since HV is computed from a finite approximation $\mathcal{F}$ of the Pareto front, we further need a reference point $\mathbf{r} \in \mathbb{R}^m$, dominated by every point in $\mathcal{F}$, and normalization point $\mathbf{z}_0 \in \mathbb{R}^m$ satisfying $z_{0,i} < \min\{f_i(x) : \mathbf{f}(x) \in \mathcal{F}\}$ for every i. The hypervolume of $\mathcal{F}$ with respect to $\mathbf{r}$ is:

$$\mathrm{HV}_{\mathbf{r}}(\mathcal{F}) = \mu\Big(\bigcup_{\mathbf{f}(x) \in \mathcal{F}} [\mathbf{f}(x), \mathbf{r}] \Big), \qquad [\mathbf{f}(x), \mathbf{r}] = \prod_{i=1}^{m} [f_i(x), r_i], \tag{17}$$

where μ is the Lebesgue measure and $[\mathbf{f}(x), \mathbf{r}]$ is the axis-aligned box between $\mathbf{f}(x)$ and $\mathbf{r}$: a larger union of these boxes means the solutions are both closer to the ideal region and more spread out along the front. We report the normalized hypervolume:

$$\mathrm{HV}'_{\mathbf{r}}(\mathcal{F}) = \mathrm{HV}_{\mathbf{r}}(\mathcal{F}) \Big/ \prod_{i=1}^{m} |r_i - z_{0,i}|, \tag{18}$$

so that a larger value indicates a Pareto front that is both closer to optimal and more diverse. For the maximization problem MOKP, the inequalities are reversed accordingly All methods share the same $\mathbf{r}$ and $\mathbf{z}_0$ for a given problem and size, listed in Table 3.

Table 3: Reference points $\mathbf{r}$ and normalization points $\mathbf{z}_0$ used for hypervolume.

Problem	n	$\mathbf{r}$	$\mathbf{z}_0$
Bi-TSP	20	(20,20)	(0,0)
	50	(35,35)	(0,0)
	100	(65,65)	(0,0)
Bi-CVRP	20	(30,4)	(0,0)
	50	(45,4)	(0,0)
	100	(80,4)	(0,0)
Bi-KP	50	(5,5)	(30,30)
	100	(20,20)	(50,50)
	200	(30,30)	(75,75)
Tri-TSP	20	(20,20,20)	(0,0,0)
	50	(35,35,35)	(0,0,0)
	100	(65,65,65)	(0,0,0)

C INSTANCE AUGMENTATION

At inference time, we apply instance augmentation to improve solution quality: an instance is transformed into a set of equivalent instances that share the same optimal solution, each is solved independently, and the best solution across all transformed instances is retained. For MOCVRP, an instance admits 8 transformations of its 2-dimensional coordinates, applied identically to the depot and every customer node, $\{(x, y),\ (y, x),\ (x, 1-y),\ (y, 1-x),\ (1-x, y),\ (1-y, x),\ (1-x, 1-y),\ (1-y, 1-x)\}$.

For an m-objective TSP instance, each of the m groups of 2-dimensional coordinates independently admits the same 8 transformations, giving 8^m transformed instances in total. This augmentation relies on the 2-dimensional Euclidean coordinates of MOTSP and MOCVRP, and does not apply to MOKP, which has no spatial coordinates.

D ANCHOR PLACEMENT

D.1 IMPLEMENTATION DETAILS OF LOCATING KNEE POINT

For every problem, the extreme points required by the payoff matrix Φ (Section 4.2) are estimated by reducing the problem to the corresponding single objective, solving it on $N_0 = 10$ sampled instances of the target size with an existing, non-neural method appropriate to that problem, and averaging the resulting objective vectors into one representative point per extreme.

For MOTSP (Bi-TSP and Tri-TSP), each node carries two or three sets of 2D coordinates, one per objective; at extreme i, the reduced single-objective TSP uses only that objective's coordinate subspace, and is solved with LKH-3.

For MOCVRP, the two extremes reduce to (i) minimizing total route distance with no per-route length limit, and (ii) minimizing total distance subject to a per-route length cap set at the theoretical lower bound on the makespan, $\mathrm{MS}_0 = 2\max_c \mathrm{dist}(\mathrm{depot}, c)$, which turns the second extreme into a makespan-oriented solution. Both are solved with a hybrid genetic search CVRP solver under a size-dependent wall-clock budget per instance ($5\mathrm{s}/15\mathrm{s}/30\mathrm{s}$ at $n = 20/50/100$).

For MOKP, the two extremes use only one of the two item values as the knapsack objective and are solved exactly with OR-Tools' knapsack solver.

Table 4 reports the knee-point preference $\boldsymbol{\lambda}^\dagger$ estimated by our method for each problem and problem size. For Bi-TSP, Bi-KP, and Tri-TSP, the objectives are symmetric by construction, so the estimated knee point stays close to an even split. Bi-CVRP behaves differently: its two objectives, the total travel distance and the single-vehicle makespan, differ substantially in raw magnitude, so the estimated knee point is consistently skewed toward one objective rather than centered, and this skew grows more pronounced as the magnitude gap between the two objectives widens with problem size.

Table 4: Knee point preference $\boldsymbol{\lambda}^\dagger$ for each problem and problem size.

Problem	$n = 20$	$n = 50$	$n = 100$	$n = 200$
Bi-TSP	(0.503,0.497)	(0.500,0.500)	(0.501,0.499)	–
Bi-CVRP	(0.157,0.843)	(0.133,0.867)	(0.107,0.893)	–
Bi-KP	–	(0.503,0.497)	(0.492,0.508)	(0.501,0.499)
Tri-TSP	(0.335,0.329,0.336)	(0.324,0.338,0.338)	(0.330,0.338,0.332)	–

D.2 ANCHOR PLACEMENT FOR THREE OBJECTIVES

Locating the knee point requires no change when $m = 3$. Each extreme preference reduces the problem to a single-objective TSP on that objective's own coordinates, which is again solved with LKH-3, and the three resulting extreme points fill the 3×3 payoff matrix Φ.

Placing the remaining anchors, however, has to account for the preference space being a triangle rather than a segment. Its vertices are the three extreme preferences, and each of its edges is itself

a bi-objective preference space, so it has its own knee: restricting $\mathbf{f}(x^{(i)})$ and $\mathbf{f}(x^{(j)})$ to their i-th and j-th coordinates gives a 2×2 payoff matrix, to which the same system applies, and we call its solution $\boldsymbol{\lambda}^{\dagger}_{ij}$ an *edge knee*. The extreme points are already available, so the three edge knees cost no additional solving.

These four knees induce three groups of segments, and each segment is divided into equal parts with one anchor placed at every interior division point, which is the bi-objective rule of Section 4.2 applied along one direction at a time. The segments from $\boldsymbol{\lambda}^{\dagger}$ to the three extreme preferences are divided into R_1 parts, those from $\boldsymbol{\lambda}^{\dagger}$ to the three edge knees into R_2 parts, and, within each edge, the two segments that its edge knee cuts the edge into are divided into R_3 parts. Together with the global knee and the three edge knees themselves, this gives $M = 4 + 3(R_1 - 1) + 3(R_2 - 1) + 6(R_3 - 1)$ anchors, all of them positioned relative to a knee rather than at a fixed geometric location. The three resolutions are set independently because the triangle is two-dimensional, so a single one cannot control the spacing towards the extreme preferences, towards the edge knees and along the edges at once. We set $R_1 = R_2 = R_3 = 2$ for Tri-TSP20 and Tri-TSP50, which gives $M = 16$, and $R_1 = R_2 = R_3 = 3$ for Tri-TSP100, which gives $M = 28$. Figure 3 shows the resulting layout for $M = 16$.

Transfer training then proceeds outwards from the global knee, the only anchor trained from a random initialization, along two chains that never meet. On one chain, the anchors on a segment towards an extreme preference are trained in order of increasing distance from the knee, and the last of them passes its parameters on to the anchors lying on the two edge segments that end at the same vertex. On the other, the anchors on a segment towards an edge knee are trained in the same order, and the last of them passes its parameters on to that edge knee. Every remaining anchor is therefore adapted for E_1 epochs from its nearest already-trained neighbor, as in the bi-objective case, and anchors at the same distance from the knee are trained in parallel.

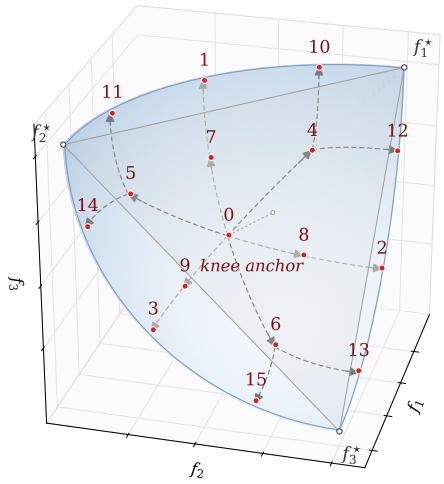


Figure 3: Anchor placement on the three-objective preference simplex with $R_1 = R_2 = R_3 = 2$, giving $M = 16$. The global knee, the three edge knees and the anchors placed on the segments they induce are shown together with the transfer chains along which the anchors are trained.

E GENERALIZATION EXPERIMENTS

We further examine how well the solvers of ENS transfer beyond the training setting through two experiments. The first tests generalization to larger sizes: all methods are trained on Bi-TSP100 and evaluated, without any retraining, on Bi-TSP150 and Bi-TSP200. The second replaces the synthetic instances by the real-world KroAB100/150/200 benchmark instances of TSPLIB, whose node coordinates follow a different distribution from the uniformly sampled ones used for training. In both experiments, ENS reuses the anchors and the evolved solvers obtained on Bi-TSP100, so nothing about the model is adapted to the new instances.

Table 5 reports the results on the unseen larger sizes. Although its solver set is reused without any adaptation, ENS leads every multi-model baseline at both sizes, with and without augmentation, and stays within 0.07% of the best neural method on Bi-TSP150 and within 0.22% on Bi-TSP200, where the closest competitors are the augmented single-model methods WE-CA and CNH. A set of solvers evolved at $n = 100$ therefore remains competitive with the strongest neural methods at sizes it has never seen.

Table 6 reports the results on the KroAB benchmark, whose node coordinates follow a different distribution from the training instances. On KroAB100, ENS-Aug attains the best hypervolume among all learning-based methods, indicating that ENS is robust to a shift in instance distribution at the size it was evolved for. On KroAB150 and KroAB200, where a size shift is added, ENS-Aug stays within 0.2% of the best neural method and ahead of every multi-model baseline, so the evolved solvers remain competitive even under both kinds of shift at once.

Table 5: Out-of-distribution generalization results on unseen larger sizes (Bi-TSP150 and Bi-TSP200), with all methods trained on Bi-TSP100.

		Bi-TSP150			Bi-TSP200		
	Method	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
Non-learnable	WS-LKH	**0.7148**	**-1.77%**	13h	**0.7490**	**-2.23%**	22h
	MOEA/D	0.6809	3.06%	2.4h	0.7139	2.57%	2.7h
	NSGA-II	0.6659	5.20%	6.8h	0.7045	3.85%	6.9h
	MOGLS	0.6768	3.64%	22h	0.7114	2.91%	38h
	PPLS/D-C	0.6784	3.42%	21h	0.7106	3.02%	32h
Single-model	PMOCO	0.6910	1.62%	50s	0.7231	1.31%	1.5m
	MORAM	0.6711	4.46%	4s	0.7024	4.14%	7s
	CNH	0.6985	0.56%	1.1m	0.7292	0.48%	1.9m
	WE-CA	0.6991	0.47%	56s	0.7264	0.86%	1.6m
Multi-model	DRL-MOA	0.6901	1.75%	45s	0.7219	1.47%	1.5m
	MDRL	0.6922	1.45%	40s	0.7251	1.04%	1.4m
	EMNH	0.6930	1.34%	40s	0.7260	0.91%	1.4m
	ENS	0.6968	0.80%	1.8m	0.7271	0.76%	4m
Single-model-Aug	PMOCO-Aug	0.6967	0.81%	47m	0.7283	0.60%	1.6h
	CNH-Aug	0.7025	-0.01%	52m	0.7343	-0.22%	1.7h
	WE-CA-Aug	0.7029	-0.07%	51m	0.7307	0.27%	1.7h
Multi-model-Aug	MDRL-Aug	0.6976	0.68%	47m	0.7299	0.38%	1.6h
	EMNH-Aug	0.6983	0.58%	47m	0.7307	0.27%	1.6h
	ENS-Aug	0.7024	0.00%	2h	0.7327	0.00%	4.2h

Table 6: Results on real-world benchmark instances (KroAB100/150/200).

		KroAB100			KroAB150			KroAB200		
	Method	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
Non-learnable	WS-LKH	**0.7022**	**-0.21%**	2.3m	**0.7017**	**-1.20%**	4.0m	**0.7430**	**-1.91%**	5.6m
	MOEA/D	0.6836	2.44%	5.8m	0.6710	3.23%	7.1m	0.7106	2.54%	7.3m
	NSGA-II	0.6676	4.72%	7.0m	0.6552	5.51%	7.9m	0.7011	3.84%	8.4m
	MOGLS	0.6817	2.71%	52m	0.6671	3.79%	1.3h	0.7083	2.85%	1.6h
	PPLS/D-C	0.6785	3.17%	38m	0.6659	3.97%	1.4h	0.7100	2.62%	3.8h
Single-model	PMOCO	0.6878	1.84%	9s	0.6819	1.66%	17s	0.7193	1.34%	23s
	MORAM	0.6723	4.05%	2s	0.6603	4.77%	2s	0.6955	4.61%	2s
	CNH	0.6947	0.86%	25s	0.6892	0.61%	35s	0.7250	0.56%	41s
	WE-CA	0.6959	0.69%	9s	0.6898	0.52%	19s	0.7232	0.81%	25s
Multi-model	DRL-MOA	0.6903	1.48%	10s	0.6794	2.02%	18s	0.7185	1.45%	23s
	MDRL	0.6881	1.80%	10s	0.6831	1.49%	17s	0.7209	1.12%	23s
	EMNH	0.6900	1.53%	9s	0.6832	1.47%	16s	0.7217	1.02%	23s
	ENS	0.6951	0.80%	1s	0.6871	0.91%	1.2s	0.7232	0.81%	2s
Single-model-Aug	PMOCO-Aug	0.6937	1.00%	12s	0.6886	0.69%	19s	0.7251	0.55%	32s
	CNH-Aug	0.6980	0.39%	30s	0.6938	-0.06%	37s	0.7303	-0.16%	54s
	WE-CA-Aug	0.6997	0.14%	14s	0.6944	-0.14%	23s	0.7279	0.16%	38s
Multi-model-Aug	MDRL-Aug	0.6950	0.81%	13s	0.6890	0.63%	19s	0.7261	0.41%	28s
	EMNH-Aug	0.6958	0.70%	12s	0.6892	0.61%	18s	0.7270	0.29%	27s
	ENS-Aug	0.7007	0.00%	5s	0.6934	0.00%	35s	0.7291	0.00%	76s

F Complete Algorithm

Algorithm 1 summarizes the complete procedure.

Algorithm 1 Evolving Neural Solver (ENS)

Require: number of anchors M; epochs E_0, E_1; target preferences $\{\Lambda_c\}_{c=1}^{M}$; generations G; offspring size K; reseed ratio r; mutation scale σ; step size α; EMA rate β; evaluation instances U

Ensure: anchors $\{\mathbf{w}_c\}_{c=1}^{M}$ and their evolved solver sets $\{S_c\}_{c=1}^{M}$

Stage one: knee point initialization and transfer

estimate the extreme points $\{\mathbf{f}(x^{(i)})\}_{i=1}^{m}$ and obtain $\boldsymbol{\lambda}^{\dagger}$ from the payoff matrix Φ

place the M anchor preferences $\{\boldsymbol{\lambda}^{(c)}\}$ relative to $\boldsymbol{\lambda}^{\dagger}$

train the knee anchor from a random initialization for E_0 epochs; $\mathcal{A} \leftarrow \{c^{\dagger}\}$

while $|\mathcal{A}| < M$ **do**

  warm-start the untrained anchor c closest to $\mathcal{A}$ from its nearest trained anchor, and train it for E_1 epochs under $\boldsymbol{\lambda}^{(c)}$; $\mathcal{A} \leftarrow \mathcal{A} \cup \{c\}$

end while

$\mathbf{w}_c \leftarrow$ flattened W_q of anchor c for $c = 1, \ldots, M$, with all remaining weights frozen

Stage two: evolutionary extension

for $c = 1$ **to** M **do**

  $S_c \leftarrow \emptyset$; $\boldsymbol{\delta}_i^c \leftarrow \mathbf{0}$ for $i = 1, \ldots, m$

end for

for $g = 1$ **to** G **do**

  Solution generation

  for $c = 1$ **to** M **do**

    $K_r \leftarrow K$ and $K_e \leftarrow 0$ if $g = 1$, otherwise $K_r \leftarrow \mathrm{round}(rK)$ and $K_e \leftarrow K - K_r$

    generate K_r reseed offspring $\mathbf{w}' = \mathbf{w}_c + \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$

    generate K_e elite offspring $\mathbf{w}' = \mathbf{w}_{\text{parent}} + \boldsymbol{\epsilon} + \alpha\, \boldsymbol{\delta}_i^c$, split evenly among $i = 1, \ldots, m$, with parents drawn cyclically from S_c

    $P_c \leftarrow S_c \cup \{\text{offspring}\}$

  end for

  Solution evaluation

  sample a batch of U instances

  for $c = 1$ **to** M, **in parallel do**

    for each candidate $\mathbf{w}' \in P_c$ **do**

      reshape $\mathbf{w}'$ into W_q, substitute it into the frozen backbone, and roll out N trajectories on each of the U instances

      for each $\boldsymbol{\lambda} \in \Lambda_c$, apply the internal selection to obtain the representative point $\bar{\mathbf{f}}(\mathbf{w}' \mid \boldsymbol{\lambda})$

    end for

  end for

  pool the representative points of all candidates across the M anchors and mark those that are non-dominated

  for $c = 1$ **to** M **do**

    $S_c \leftarrow \{\mathbf{w}' \in P_c$: at least one representative point of $\mathbf{w}'$ is non-dominated$\}$

  end for

  Searching direction update

  for $c = 1$ **to** M **do**

    assign the survivors of anchor c to the groups $\mathcal{B}_1^c, \ldots, \mathcal{B}_m^c$

    update $\boldsymbol{\delta}_i^c$ by the EMA of the mean perturbation over $\mathcal{B}_i^c$, for $i = 1, \ldots, m$

  end for

end for

return $\{S_c \cup \{\mathbf{w}_c\}\}_{c=1}^{M}$