Deep Reinforcement Learning for Multiobjective Optimization

1 Deep Reinforcement Learning Based Multiobjective Optimization Algorithm (DRL-MOA)

这个研究基于两个关键背景：

- 多目标优化问题（Multiobjective Optimization Problem, MOP）的基本困境：传统方法（如 NSGA-II, MOEA/D）通过迭代更新种群寻找 Pareto 最优解，但面对大规模问题时（如 200 城市的多目标旅行商问题 MOTSP）时，迭代次数多、计算效率低，且问题稍有变化（如城市位置微调）就需要重新计算；
- 深度强化学习（Deep Reinforcement Learning, DRL）的优势：DRL 能通过试错学习训练一个“黑箱模型”，训练后只需要一次前向传播就能输出解，无需迭代，且泛化能力强（能够处理未见过的问题实例）。

文章[1]提出的 DRL-MOA 框架，本质是用“分解思想”（来自 MOEA/D[2]）拆解多目标优化问题，用“DRL + 神经网络”（来自 Pointer Network[3]、Actor-Critic[4-5]）求解每个子问题，再用“参数迁移”加速训练。

2 General Framework

通用框架是 DRL-MOA 的“骨架”，解决了“如何将多目标问题转化为 DRL 可处理的单目标问题”和“如何高效训练多个子问题的模型”这两个核心问题。分为分解策略和邻域参数迁移策略两部分。

2.1 Decomposition Strategy

Decomposition Strategy: 文章中采用 weighted sum approach[6] 进行多目标优化问题的分解，也可以采用其他 scalarizing methods，比如 Chebyshev 和 the penalty-based boundary intersection (PBI) method[7-8]。首先，生成一组均匀分布的权重向量（uniformly spread weight vector）$\lambda^1, \lambda^2, \ldots, \lambda^N$，其中 N 为子问题的数量，比如对于双目标问题（$M = 2$），可以取权重向量为 $(1,0), (0.9,0.1), \ldots, (0,1)$，每个向量表示对不同目标的“重视程度”。对第 j 个权重向量 $\lambda^j = (\lambda_1^j, \lambda_2^j, \ldots, \lambda_M^j)^{\mathrm{T}}$，$M$ 表示目标函数的个数，通过 weighted sum approach，可以将 MOTSP 分解为 N 个单目标优化子问题（scalar optimization subproblems）。第 j 个子问题的目标函数为：

$$\min g^{ws}(x \mid \lambda_i^j) = \sum_{i=1}^{M} \lambda_i^j f_i(x) \tag{1}$$

其中 $f_i(x)$ 是原 MOP 的第 i 个目标函数，$g^{ws}(x \mid \lambda_i^j)$ 是第 j 个子问题的“加权和成本”（单目标）。

分解后每个子问题的解都是原 MOP 的 Pareto 最优解，这是因为权重向量的不同权衡，使得每个子问题的最优解对应 PF（Pareto Front）上的一个“权衡点”。通过将 MOP 分解成子问题，可以将每个子问题的“加权和成本”作为 DRL 的“奖励信号（比如奖励 = 负的加权和成本，因为 DRL 通常最大化奖

励，而 MOP 需要最小化成本）。这样就通过将 MOP 拆解为多个标量子问题，每个子问题对应一个“权重向量”，求解所有子问题的解就可以组成 PF。

2.2 Neighborhood-Based Parameter-Transfer Strategy

Neighborhood-Based Parameter-Transfer Strategy: 采用邻域参数迁移的策略的核心在于，如果每个子问题都“从头训练”一个神经网络，计算量会非常大（N 个子问题需要 N 次独立训练）。但文章根据公式 (1) 和 Zhang 的研究[2] 发现，相邻权重向量对应的子问题，其最优解和最优模型参数非常相似，比如在双目标问题中，权重向量为 $(0.8, 0.2)$ 和 $(0.7, 0.3)$ 的子问题对于目标的权衡接近，最优路径和模型参数也接近。因此，借鉴 MOEA/D 的“邻域更新”思想[2]，提出了邻域参数迁移策略，即用前一个子问题的最优模型参数，作为当前子问题的初始参数，避免从头训练，减少计算成本。

其具体过程为：假设已经训练好第 $i-1$ 个子问题的最优模型参数 $[w^*_{\lambda^{i-1}}, b^*_{\lambda^{i-1}}]$（$w$ 为权重，b 为偏置），在训练第 i 个子问题时，使用 $[w^*_{\lambda^{i-1}}, b^*_{\lambda^{i-1}}]$ 作为初始参数（$[w_{\lambda^i}, b_{\lambda^i}] = [w^*_{\lambda^{i-1}}, b^*_{\lambda^{i-1}}]$）进行训练。然后在此基础上用 Actor-Critic 进行微调，快速收敛到第 i 个子问题的最优参数 $[w^*_{\lambda^i}, b^*_{\lambda^i}]$。重复该过程，直到所有 N 个子问题都训练完毕。

邻域参数迁移策略的优势在于无需为每个子问题初始化随机参数，从而减少了收敛时间，降低了训练复杂度；同时，由于相邻子问题的模型参数平滑过渡，避免 PF 上出现“跳跃”的解，保证了解的一致性和多样性。

2.3 Pseudo Code of General Framework of DRL-MOA

DRL-MOA 的通用框架伪代码如 algorithm 1所示。每个子问题的训练核心是 Actor-Critic 算法，负责将子问题的“加权和成本”转化为模型的优化信号。训练完成之后，对于新的 MOP 实例，只需要一次前向传播（forward propagation）就能得到对应的 Pareto 最优解，无需重新训练。

Algorithm 1: General Framework of DRL-MOA

Input: The model of the subproblem $\mathcal{M} = [\mathbf{w}, \mathbf{b}]$, weight vectors $\lambda^1, \ldots, \lambda^N$

Output: The optimal model $\mathcal{M}^* = [\mathbf{w}^*, \mathbf{b}^*]$

$[\omega_{\lambda^1}, \mathbf{b}_{\lambda^1}] \leftarrow$ Random_Initialize
for $i \leftarrow 1$ **to** N **do**
  if $i == 1$ **then**
    $[\omega^*_{\lambda^1}, \mathbf{b}^*_{\lambda^1}] \leftarrow$ Actor_Critic$([\omega_{\lambda^1}, \mathbf{b}_{\lambda^1}], g^{\text{ws}}(\lambda^1))$
  else
    $[\omega_{\lambda^i}, \mathbf{b}_{\lambda^i}] \leftarrow [\omega^*_{\lambda^{i-1}}, \mathbf{b}^*_{\lambda^{i-1}}]$
    $[\omega^*_{\lambda^i}, \mathbf{b}^*_{\lambda^i}] \leftarrow$ Actor_Critic$([\omega_{\lambda^i}, \mathbf{b}_{\lambda^i}], g^{\text{ws}}(\lambda^i))$
  end if
end for
return $[\mathbf{w}^*, \mathbf{b}^*]$
/ Given inputs of the MOP, the PF can be directly calculated by* $[\mathbf{w}^*, \mathbf{b}^*]$*. */*

3 Modeling the Subproblem of MOTSP

文章通过 MOTSP 实例，展示了如何将具体的 MOP 转化为 DRL-MOA 框架中的子问题，这部分分为 MOTSP 问题的定义、模型结构（经过修改后的 Pointer Network[4]）和训练方法（Actor-Critic[4-5]）

三部分。

文章的实验实例是多目标旅行商问题 MOTSP: The multiobjective traveling salesman problem (MOTSP), where given n cities and M cost functions to travel from city i to j, one needs to find a cyclic tour of the n cities, minimizing the M cost functions.

文章为了将输入 X 映射到输出 Y，用概率链式法则（probability chain rule）将排列 Y 的概率分解为条件概率的乘积：

$$P(Y \mid X) = \prod_{t=1}^{n} P(\rho_{t+1} \mid \rho_1, \rho_2, \ldots, \rho_t, X_t) \tag{2}$$

首先，随机选择一个城市作为起点 ρ_1，然后在每一步 $t = 1, 2, \ldots$ 过程中，基于当前已选择的城市 $\{\rho_1, \rho_2, \ldots, \rho_t\}$ 在剩余未选择的城市 X_t 中选择下一个城市 ρ_{t+1}，在选择完一个城市后，将其从 X_t 中移除，直到所有城市都被选择完毕。

文章中采用的类似于 Nazari 等人[4] 的 Pointer Network 的基本结构是 Sequence-to-Sequence 模型[9]。Sequence-to-Sequence 模型由两层 RNN 组成，分别是 Encoder 和 Decoder。Encoder RNN 负责将输入序列编码为隐藏状态（hidden state），Decoder RNN 则基于隐藏状态生成输出序列。即 Encoder RNN 将输入序列转化成一个上下文向量（code vector），然后 Decoder RNN 基于该上下文向量逐步生成输出序列。

3.1 Formulation of MOTSP

One needs to find a tour of n cities, that is, a cyclic permutation ρ, to minimize M different cost functions simultaneously.

$$\min z_k(\rho) = \sum_{i=1}^{n-1} c^k_{\rho(i),\rho(i+1)} + c^k_{\rho(n),\rho(1)}, \quad k = 1, 2, \ldots, M \tag{3}$$

where $c^k_{\rho(i),\rho(i+1)}$ is the k-th cost of traveling from city $\rho(i)$ to $\rho(i+1)$. The cost functions may, for example, correspond to tour length, safety index, or tourist attractiveness in practical applications.

3.2 Model

文章采用修改后的 Pointer Network[3] 作为解决 MOTSP 子问题的神经网络模型。模型可以分为输入和输出结构、编码器（Encoder）、解码器（Decoder）和注意力机制（Attention Mechanism）四部分。

3.2.1 Input and Output Structure

模型的输入为 $X = \{s^i, i = 1, 2, \ldots, n\}$，其中 n 为城市的数量。每个 s^i 由一个元组（tuple）$\{s^i = (s^i_1, s^i_2, \ldots, s^i_M)\}$ 表示，s^i_j 为城市 i 在第 j 个目标下用于计算成本的特征。比如 $s^i_1 = (x_i, y_i)$ 表示城市 i 的二维坐标（用于计算距离），s^i_2 可以表示城市 i 的安全指数等。模型的输出为一个城市的排列 $Y = \{\rho_1, \rho_2, \ldots, \rho_n\}$。

3.2.2 Encoder

Encoder: Encoder 的作用是将输入的城市特征 X 编码为一个高维向量，方便 Decoder 使用。由于城市的位置是无序的，在输入结构中城市之间的顺序是没有任何意义的，因此文章中没有采用 RNN 作为 Encoder（比如 long short-term memory (LSTM)，因为 RNN 会引入不必要的"顺序偏见"），并且由于 1-D 卷积层参数是共享的，计算更快，且对城市数量的泛化性更强，因此使用了 a simple embedding layer (the 1-D convolution layer) 来将输入 X 编码到一个高维向量（high-dimensional vector）空间中[4]（文章中的超参数 $d_h = 128$）。

3.2.3 Decoder

Decoder: Decoder 的作用是按顺序生成城市排列 Y（或者说一个城市的选择序列索引）。由于在 Decoder 中需要总结之前选择的城市信息 $\rho_1, \ldots, \rho_t$，然后进行下一步的城市选择 ρ_{t+1}，因此 Decoder 中需要使用 RNN，文章中使用 the gated recurrent unit (GRU)[10]，这种 RNN 结构比 LSTM 的参数更少但是性能相似（原始的 Pointer Network 使用的是 LSTM[4]）。文章中的 RNN 并不是直接用于生成下一个城市的选择，而是用 RNN Decoder 在解码到 t 时的隐藏状态（hidden state）d_t 来存储之前选择的城市信息 $\rho_1, \ldots, \rho_t$。通过结合 d_t 和 Encoder 的输出 $e_1, \ldots, e_n$ 用注意力机制（Attention Mechanism）来计算下一个城市选择的条件概率 $P(\rho_{t+1} \mid \rho_1, \ldots, \rho_t, X_t)$。

3.2.4 Attention Mechanism

Attention Mechanism: 直观上来看，注意力机制计算每个输入 e_i 对下一次解码步骤 t 的相关性（relevance），最相关的输入 e_i 会被赋予更高的权重（the most relevant one is given more attention）并且更有可能被选择为下一个访问的城市。计算的公式如公式 (4) 所示。

$$u_j^t = v^{\mathrm{T}} \tanh(W_1 e_j + W_2 d_t), \quad j \in (1, 2, \ldots, n) \tag{4}$$

其中 W_1, W_2 和 v 是需要学习的参数，e_j 是 Encoder 的输出，d_t 是 Decoder 在时间步 t 的隐藏状态。通过对每个未选择的城市 j 用 GRU 的隐藏状态 d_t 和 Encoder 的输出 e_j 计算相关性得分 u_j^t，u_j^t 越大，说明城市 j 越适合作为下一步选择的城市。然后通过 softmax 函数将相关性得分转化为概率分布，即将 $u_1^t, u_2^t, \ldots, u_n^t$ 归一化（normalize）为概率。

$$P(\rho_{t+1} \mid \rho_1, \ldots, \rho_t, X_t) = \mathrm{softmax}(u^t) \tag{5}$$

在选择下一个城市 ρ_{t+1} 时，文章采用贪心策略（greedy）选择概率最大的城市作为下一个访问的城市。在训练过程中，文章采用采样策略（sampling），从概率分布中采样下一个城市 ρ_{t+1}，以增加探索性。

3.3 Training Method

文章中采用 Actor-Critic 算法训练子问题的模型[4-5]。Actor-Critic 算法的伪代码如 algorithm 2所示。

Algorithm 2: Actor–Critic Training Algorithm

Input: $\theta, \phi \leftarrow$ Initialized parameters given in algorithm 1

Output: The optimal parameters θ, ϕ

for *iteration* $\leftarrow 1, 2, \ldots$ **do**
  generate T problem instances from $\{\Phi_{\mathcal{M}_1}, \ldots, \Phi_{\mathcal{M}_M}\}$ for the MOTSP
  for $k \leftarrow 1, \ldots, T$ **do**
    $t \leftarrow 0$
    while *not terminated* **do**
      select the next city ρ_{t+1}^k according to $P(\rho_{t+1}^k | \rho_1^k, \ldots, \rho_t^k, X_t^k)$
      Update X_t^k to X_{t+1}^k by leaving out the visited cities
    end while
    compute the reward R^k
  end for
  $d\theta \leftarrow \frac{1}{N} \sum_{k=1}^{N} \left(R^k - V(X_0^k; \phi)\right) \nabla_\theta \log P(Y^k | X_0^k)$
  $d\phi \leftarrow \frac{1}{N} \sum_{k=1}^{N} \nabla_\phi \left(R^k - V(X_0^k; \phi)\right)^2$
  $\theta \leftarrow \theta + \eta d\theta$
  $\phi \leftarrow \phi + \eta d\phi$
end for

Actor-Critic 算法分为两个主要部分：Actor 网络和 Critic 网络。Actor 网络就是前面介绍的修改后的 Pointer Network，负责给出选择下一个城市的概率分布 $P(\rho_{t+1} \mid \rho_1, \ldots, \rho_t, X_t)$，参数记为 θ。Critic 网络用于评估当前问题实例的价值（evaluates the expected reward given a specific problem state），参数记为 ϕ，结构和 Encoder 一致（1-D 卷积层），输入是问题实例，输出是一个标量（价值估计）。

训练是一个无监督的过程（unsupervised），目标是最大化期望奖励，即最小化加权和成本。训练的核心是策略梯度下降，首先通过从分布 $\{\Phi_{\mathcal{M}_1}, \ldots, \Phi_{\mathcal{M}_M}\}$ 中采样 T 个 MOTSP 实例，其中，$\mathcal{M}$ 表示城市的不同输入特征（比如城市坐标、安全指数等）。然后对于每个实例，通过当前参数为 θ 的 Actor 网络生成一个城市排列 ρ^k，计算对应的奖励 R^k。然后用策略梯度（policy gradient）更新 Actor 网络的参数 θ[11]。其中，$V(X_0^n; \phi)$ 是 Critic 网络对问题实例 n 的价值估计（reward approximation）。然后通过最小化“真实奖励”和 Critic 估计价值之间的平方差来更新 Critic 网络的参数 ϕ。重复该过程，直到参数收敛。

参考文献

[1] LI K, ZHANG T, WANG R. Deep Reinforcement Learning for Multiobjective Optimization[J/OL]. IEEE Transactions on Cybernetics, 2021, 51(6): 3103-3114. DOI: 10.1109/TCYB.2020.2977661.

[2] ZHANG Q, LI H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition [J/OL]. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712-731. DOI: 10.1109/TEVC.2007.892759.

[3] VINYALS O, FORTUNATO M, JAITLY N. Pointer Networks[C]//Advances in Neural Information Processing Systems: Vol. 28. Curran Associates, Inc., 2015.

[4] NAZARI M, OROOJLOOY A, SNYDER L, et al. Reinforcement Learning for Solving the Vehicle

Routing Problem[C]//Advances in Neural Information Processing Systems: Vol. 31. Curran Associates, Inc., 2018.

[5] BELLO I, PHAM H, LE Q V, et al. Neural Combinatorial Optimization with Reinforcement Learning: arXiv:1611.09940[A/OL]. 2017. arXiv: 1611.09940.

[6] MIETTINEN K. Nonlinear multiobjective optimization: Vol. 12[M]. Springer Science & Business Media, 1999.

[7] WANG R, ZHOU Z, ISHIBUCHI H, et al. Localized weighted sum method for many-objective optimization[J/OL]. IEEE Transactions on Evolutionary Computation, 2018, 22(1): 3-18. DOI: 10.1109/TEVC.2016.2611642.

[8] WANG R, ZHANG Q, ZHANG T. Decomposition-based algorithms using pareto adaptive scalarizing methods[J/OL]. IEEE Transactions on Evolutionary Computation, 2016, 20(6): 821-837. DOI: 10.1109/TEVC.2016.2521175.

[9] SUTSKEVER I, VINYALS O, LE Q V. Sequence to Sequence Learning with Neural Networks[C]// Advances in Neural Information Processing Systems: Vol. 27. Curran Associates, Inc., 2014.

[10] CHO K, VAN MERRIENBOER B, GULCEHRE C, et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation: arXiv:1406.1078[A/OL]. 2014. arXiv: 1406.1078.

[11] KONDA V, TSITSIKLIS J. Actor-Critic Algorithms[C]//Advances in Neural Information Processing Systems: Vol. 12. MIT Press, 1999.