第四章 基于深度强化学习的协同决策模型求解 HFPDPTW 问题

本章针对 HFPDPTW 这一复杂场景问题，提出了一种基于深度强化学习的求解模型。本章首先对 HFPDPTW 问题进行了分析和描述，并为其建立了数学模型。然后将问题的求解过程构建为马尔可夫决策过程，详细阐述了提出的求解模型，概述了模型的整体框架，重点描述了策略网络的结构设计，并介绍了模型的训练算法设计。最后，设计了对比实验和消融实验以验证所提出模型的有效性和先进性。

4.1 HFPDPTW 问题分析与数学建模

4.1.1 问题定义

对于 HFPDPTW 问题的定义如下：给定若干个取送货订单，其中每个订单包括一个取货客户节点和一个对应的送货客户节点，每个节点具有不同的地理位置和时间窗要求，完成一个订单需将货物从取货节点取出并配送到对应的送货节点；同时，给定从一个车场出发的若干车辆，车辆根据核载和使用成本可分为多种车型，合理为每辆车规划一条从车场出发、按一定顺序服务若干客户节点后返回车场的路线，使得整个过程的配送成本最小。

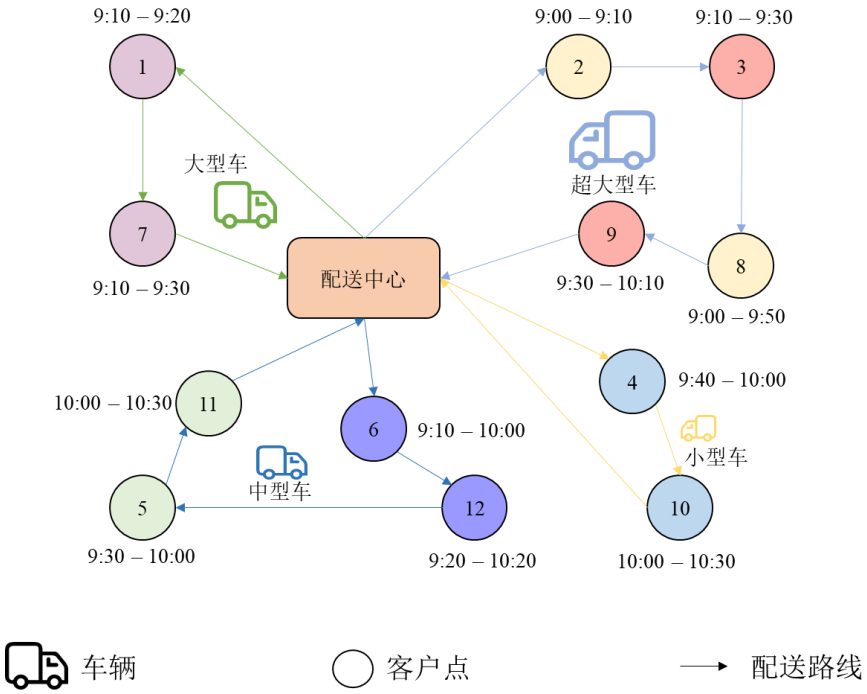


图 4-1 HFPDPTW 问题场景示意图

如图 4-1 所示，给出了 HFPDPTW 问题场景的示意图，图中不同取送货订单的客户点用不同颜色画出，一个订单包含一个取货点和对应的送货点，如（1，7）、（2，8）等。车队中的可用配送车辆共包含四种车型，分别是：超大型车、大车型、中型车以及小型车。

根据问题定义，HFPDPTW 问题的关键约束条件包括：

（1）同一个订单的取货任务和送货任务必须由同一个车辆来完成，且车辆必须从取货节点取到货物，然后送往对应的送货节点；

（2）车辆必须在客户节点指定的时间窗内进行取货或送货；

（3）车辆在配送过程中的负载量不允许出现超出核载的情况，同时，由于不同车辆具有不同的核载和使用成本，在进行车辆调度时，根据客户节点需求、优化目标等因素合理安排车辆。

4.1.2 数学模型构建

根据 HFPDPTW 问题的定义，本节为 HFPDPTW 问题建立数学模型。首先介绍涉及到的参数和变量如下：

（1）n：取送货订单数量；

（2）$P=\{1,2,...,n\}$ 和 $D=\{n+1,n+2,...,2n\}$：取货客户点的集合 P 以及送货客户点的集合 D；

（3）$V=0\bigcup P\bigcup D$：所有节点的集合；

（4）K：车队中车辆的总数量

（5）$C=\{C_1,C_2,...,C_K\}$：车辆核载的集合，其中 C_k 表示第 k 辆车的核载；

（6）$w=\{(w_0^1,w^1),(w_0^2,w^2),...,(w_0^K,w^K)\}$：车辆使用成本的集合，其中 w_0^k 表示第 k 辆车的单次调度成本，w^k 表示第 k 辆车的单位距离运输成本；

（7）q_i：客户点 i 的需求量；

（8）a_i 和 b_i：客户点 i 的最早开始服务时间和最晚开始服务时间；

（9）o_i：车辆在客户点 i 的服务花费时长；

（10）s_i^k：车辆 k 在客户点 i 的开始服务时间；

（11）C_i^k：车辆 k 离开客户点 i 时的载重量；

（12）$d_{i,j}$：从客户点 i 到客户点 j 的行驶距离；

（13）$t_{i,j}$：从客户点 i 行驶到客户点 j 所需时间；

（14）$x_{i,j}^k$：如果车辆 k 以先后顺序连续服务了客户点 i 与客户点 j，则 $x_{i,j}^k=1$；否则，$x_{i,j}^k=0$。

HFPDPTW 问题的数学模型可描述如下：

优化目标：

$$\min \quad \sum_{k=1}^{K} w_0^k + \sum_{k=1}^{K}\sum_{i\in V}\sum_{j\in V} d_{i,j} w^k x_{i,j}^k \tag{4-1}$$

约束条件：

$$\sum_{k=1}^{K}\sum_{j\in P\cup D, j\neq i} x_{i,j}^k = 1, \quad \forall i \in P \tag{4-2}$$

$$\sum_{j\in V} x_{i,j}^k = \sum_{j\in V} x_{j,n+i}^k, \quad \forall i \in P \tag{4-3}$$

$$\sum_{j\in V} x_{j,i}^k - \sum_{j\in V} x_{i,j}^k = 0, \quad \forall i \in P\cup D \tag{4-4}$$

$$0 \le C_i^k \le C^k, \quad \forall i \in P \tag{4-5}$$

$$C_j^k \ge (C_i^k + q_j) x_{i,j}^k, \quad \forall i,j \in V \tag{4-6}$$

$$(s_i^k + o_i + t_{i,j}) x_{i,j}^k \le s_j^k, \quad \forall i,j \in V \tag{4-7}$$

$$a_i \le s_i^k \le b_i, \quad \forall i \in V \tag{4-8}$$

$$s_i^k + o_i + t_{i,i+n} \le s_{i+n}^k, \quad \forall i \in P \tag{4-9}$$

其中，优化目标（4-1）是配送方案使用的车辆总成本，由每辆车的调度成本和运输成本构成。约束条件（4-2）保证每个取货任务恰好被服务一次；约束条件（4-3）保证若某订单的取货节点被服务，则其对应的送货节点也应该被服务，且两者由同一辆车完成；约束条件（4-4）确保车辆形成连续的行驶路线；约束条件（4-5）确保配送过程中始终不会违反最大载重量限制；约束条件（4-6）保证运输过程中车辆载重的变化符合实际；约束条件（4-7）保证了运输过程中到达各节点的时间是符合实际的；约束条件（4-8）保证了车辆的服务时间时符合时间窗规定；约束条件（4-9）保证满足同一订单的先取货后送货的要求。

4.2 基于深度强化学习的协同决策模型

4.2.1 基本思想和框架

本节对基于深度强化学习的 HFPDPTW 求解模型进行介绍。与第三章中所提出的模型类似，本章的模型同样也是一个改进式求解模型，通过不断迭代优化问题实例的初始解，来寻找最优解。

与第三章模型的不同之处在于，用于求解 HFPDPTW 的模型引入了多种启发式操作算子对解决方案进行迭代更新。这是出于对具体问题场景特性的考虑，由于 HFPDPTW 问题涉及异构车型，且存在成对取送货节点的配对关系，导致客户点之间形成了复杂多样的相互关系，现有的改进式模型无法高效的探索解的邻域空间。在现有的模型中，主要有两种基本的求解思想，一种是将深度学习模型作为节点选择控制器，由模型选择出节点位置后通过一种固定的操作算子执行解的更新；一种是将深度学习模型用作启发式算子选择控制器，由模型从若干算子中做出当前时刻的最佳选择，然后交给算子自行探索具体的操作位置。前者难以应对复杂场景问题，后者会导致模型的求解效率低下。

因此，本文将两种求解思想进行结合，同时选择出需要移动位置的节点和执行更新操作使用的启发式算子，高效的实现对当前解的更新。图 4-2 给出了模型求解 HFPDPTW 过程的一个例子。对于 HFPDPTW 问题，为保证不违反问题约束，节点的移动操作必须以成对的形式完成，即同时移动一个订单中的一对取送货节点，如图 4-2 中的（2，8）。在第一步迭代更新操作中，首先选择出需要移动的节点对（2，8），然后选择使用路线间交换算子将节点对（2，8）与另一对节点（3，9）的访问顺序进行交换；在第二步中，选择使用路线间重定位算子将（2，8）重定位至蓝色路线中；第三步使用路线内重定位算子将（6，12）重定位至原路线中的合适位置。

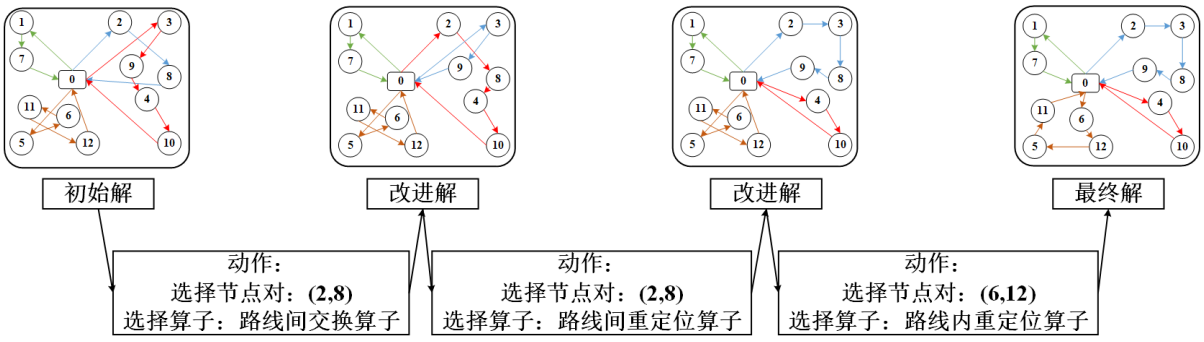


图 4-2 模型求解过程示意图

在启发式算子方面，本文引入文献[67]中为 PDP 问题精心设计的 4 种操作算子，这里对其做简要介绍如下：

（1）路线内交换算子：交换同一车辆路线内的两对取送货节点的访问顺序；

（2）路线内重定位算子：将一对取送货节点重定位至其所在车辆路线中的其他位置；

（3）路线间交换算子：将分别属于不同车辆路线的两对取送货节点进行交换；

（4）路线间重定位算子：将一对取送货节点重定位至其他车辆路线中。

在由模型确定好需要移动位置的取送货节点以及使用的算子后，由算子通过贪婪的方式确定重定位或交换的位置。

4.2.2 马尔可夫决策过程的构建

根据上述对模型求解基本原理和过程的描述，建立马尔可夫决策过程（MDP），MDP 的基本组成要素分别定义如下：

状态：状态是环境在某个决策时刻的动态表征，为智能体提供决策所需的信息。在决策时刻 t，状态 $S_t=\{\mathcal{W}_t,\mathcal{U}_t,\omega(\sigma_t)\}$ 从客户点和车辆配送路线两个维度描述了问题实例及当前解 σ_t 的信息。其中 $\mathcal{W}_t=\{\mathcal{W}_t^1,...,\mathcal{W}_t^{2n}\}$ 表示静态状态，静态状态是不会随着求解过程推进而改变的信息，如节点的坐标和时间窗等。$\mathcal{U}_t=\{\mathcal{U}_t^1,...,\mathcal{U}_t^{2n}\}$ 表示表示动态状态，动态状态是随着求解过程会发生改变的信息，如每个节点的前后相邻节点、车辆服务完节点后的剩余载重量等。$\omega(\sigma_t)=\{\omega(R_1),...,\omega(R_K)\}$ 则是从车辆路线维度描述了当前解决方案的状态。对 $\mathcal{W}_t^i$ 的完整的表述如表 4-2 所示，$\mathcal{U}_t^i$ 和 $\omega(R_k)$ 则分别如表 4-3 和表 4-4 所示。

表 4-2 HFPDPTW 问题的静态状态信息

特征	描述
(x_i,y_i)	节点 i 的二维位置坐标
q_i	节点 i 的取货需求
(a_i,b_i)	节点 i 的时间窗
o_i	车辆在节点 i 处的服务时间

表 4-3 HFPDPTW 问题的动态状态信息

特征	描述
C_i	车辆服务完节点 i 后的剩余载重量
i_{pre}	节点 i 的前序节点
i_{succ}	节点 i 的后继节点
$d_{i_{pre},i}$	节点 i 与其前序节点的距离
$d_{i,i_{succ}}$	节点 i 与其后继节点的距离
R_i	节点 i 所在路径的编号

表 4-4 HFPDPTW 问题的解维度状态信息

特征	描述
C_{R_k}	路线 R_k 所使用车辆的核载
$w_{R_k}^0$	路线 R_k 所使用车辆的单次调度成本
w_{R_k}	路线 R_k 所使用车辆的单位距离运输成本
$Cost_{R_k}$	路线 R_k 所使用车辆的总配送成本
$maxd_{R_k}$	路线 R_k 中的最大前后节点相邻距离

动作：$a_t=\{(i,i+n),opt_t\}$ 表示在当前时刻 t，为了更新当前解而选择的取送货节点对和启发式算子。

状态转移：$S_{t+1}=\tau(S_t,a_t)$ 表示采取动作 a_t 后相关状态信息的变化，主要是涉及节点维度的动态状态 $\mathcal{U}_t$ 以及车辆路线维度的状态 $\omega(\sigma_t)$ 中相关特征的更新。

奖励：与第三章的模型类似，鉴于优化目标是最大程度的改进初始解，本文使用相对于当前最优解的即时优化值作为奖励，如公式（4-10）所示。

$$r_t=f(\sigma_t^*)-\min\{f(\sigma_t^*),f(\sigma_{t+1})\} \tag{4-10}$$

4.2.3 策略网络

根据上文分析，为实现对 HFPDPTW 问题的高效求解，本文构建一种协同决策机制的深度强化学习算法，通过协同优化节点选择与启发式算子选择的动态匹配关系，突破

传统单一决策模型在解空间搜索效率上的局限性。策略网络使用双解码器结构，在任一决策时刻，首先由一个解码器识别出当前状态下对优化目标影响最大的取送货节点对，然后再由另一个解码器基于此选择出当前最适合的启发式算子，通过该算子调整被选择的节点对在解决方案中的位置，实现对解的一次更新。此外，受 Li 等人[48]研究的启发，为了充分提取原始输入的特征，捕获取送货问题中不同角色节点之间的关系，本文设计了异构车型场景下的异构注意力机制。同时，为了更好的应对异构车型场景问题，本文还引入了车辆路线维度的特征信息，用于辅助解码器进行决策。策略网络的结构图如图 4-3 所示。

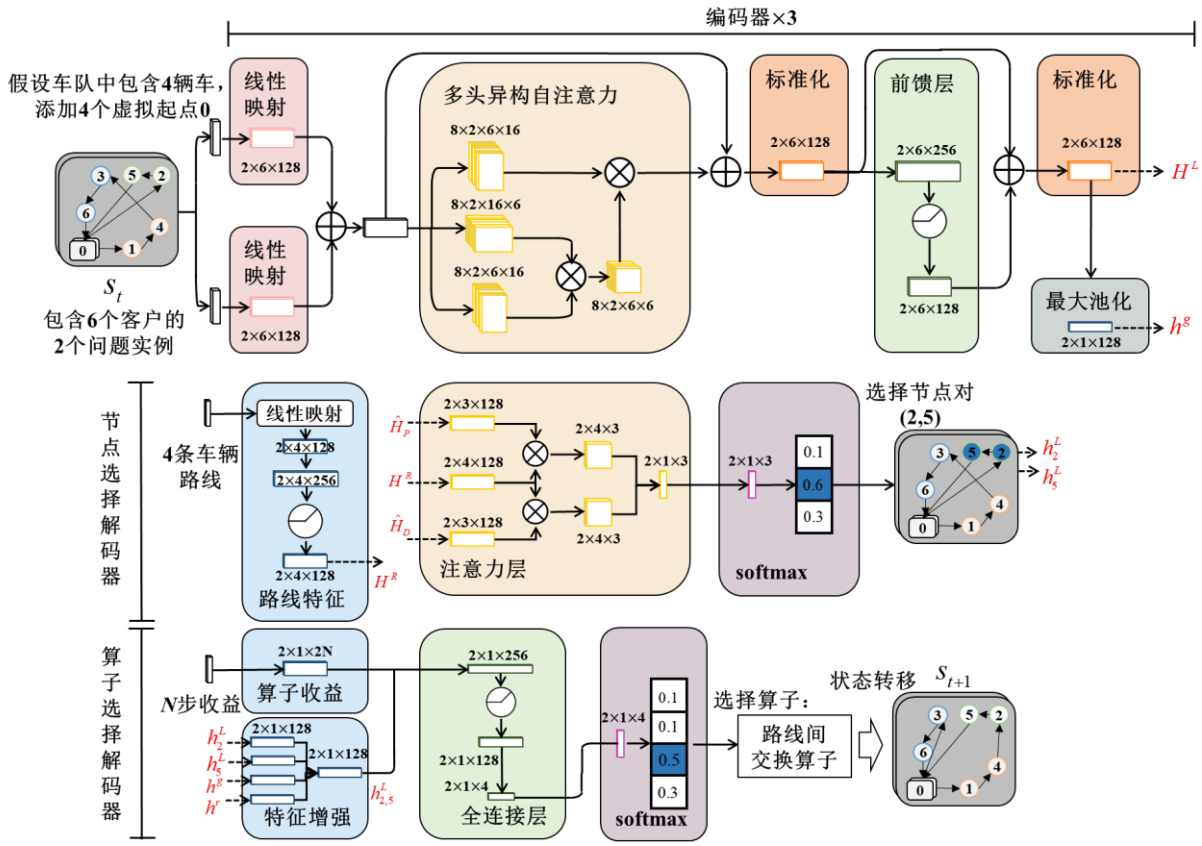


图 4-3 策略网络结构图

4.2.3.1 编码器

编码器的作用是挖掘当前状态的特征信息，以供模型决策使用。对编码器的工作流程介绍如下。

在嵌入层，针对静态特征和动态特征，分别通过线性映射转换到d_h=128 维空间中，并将两者进行融合，得到初始的嵌入特征 $\hat{h}_i^0$。

当策略网络选择在取货节点 i 处施加改进操作以更新当前解时，由于存在约束条件

（4-3）和（4-9）的限制，也必须要同时考虑节点 i 对应的送货节点 $i+n$，以确保在满足约束的情况下同时对节点 i 和 $i+n$ 的访问顺序进行更新。因此，取货节点 i 的完整表征应当包含其配对的送货节点 $i+n$ 的信息，即 $\hat{h}_i^0=[\hat{h}_i^0;\hat{h}_{i+n}^0]$。因此，将初始的嵌入特征经过线性映射，得到嵌入层的最终输出，可表示为公式（4-11）的形式：

$$h_i^0=\begin{cases}W_p^0\hat{h}_i^0+b_p^0, & 1\le i\le n\\ W_d^0[\hat{h}_i^0;\hat{h}_{i+n}^0]+b_d^0, & n+1\le i\le 2n\end{cases} \tag{4-11}$$

在注意力层，由于各节点功能属性不同、所属订单不同，节点之间形成了多种结构依赖关系，如同一订单内的取货与送货节点之间的直接关联、不同订单的取货节点之间的潜在交互、不同订单的取货与送货节点之间的间接耦合等。传统的注意力机制难以有效度量这种多样化的关系。因此，本文引入了异构注意力机制，以便更好的捕获节点之间的复杂关系，提升模型的表征能力。对于取货节点，计算它与以下各类节点之间的五种不同的注意力：配对的送货节点、由同一车辆服务的其他取货节点、由同一车辆服务的其他送货节点、由不同车辆服务的取货节点、由不同车辆服务的送货节点。同样的，对于送货节点，也计算类似的五种注意力。

用 h_i^{l-1} 表示输入到编码器第 l 层的节点 i 特征信息，对于取货节点 i，与其配对的送货节点的注意力核心要素为：

$$Q_i^{pd}=W_Q^{pd}h_i^{l-1},K_i^{pd}=W^K h_{i+n}^{l-1},V_i^{pd}=W^V h_{i+n}^{l-1} \tag{4-12}$$

取货节点到同一车辆服务的其他取货节点和送货节点的注意力查询向量分别为：

$$Q_i^{pp_{same}}=W_Q^{pp_{same}}h_i^{l-1} \tag{4-13}$$

$$Q_i^{pd_{same}}=W_Q^{pd_{same}}h_i^{l-1} \tag{4-14}$$

取货节点到不同车辆服务的其他取货节点和送货节点的注意力查询向量分别为：

$$Q_i^{pp_{diff}}=W_Q^{pp_{diff}}h_i^{l-1} \tag{4-15}$$

$$Q_i^{pd_{diff}}=W_Q^{pd_{diff}}h_i^{l-1} \tag{4-16}$$

类似的，对于送货节点，也有 Q_i^{dp}、$Q_i^{dp_{same}}$、$Q_i^{dd_{same}}$、$Q_i^{dp_{diff}}$、$Q_i^{dd_{diff}}$ 五种注意力查询向量。其中，参数矩阵均是可学习的，并且十种注意力的键向量及值向量的参数矩阵是共享的，只将异构注意力的查询向量进行了差异化设计，这样的好处在于减少了模型的参数量，在提高模型表征的同时尽可能的减少了模型训练的消耗时长。

基于异构注意力的核心要素，可得每种注意力的计算方式，如公式（4-17）所示：

$$\alpha_{ij}^{\partial}=\mathrm{softmax}\left(\frac{Q_i^{\partial T}K_j^{\partial}}{\sqrt{d_k}}\right),\quad \partial\in\{pd,dp,pp_{same},pd_{same},pp_{diff},pd_{diff},dp_{same},dd_{same},dp_{diff},dd_{diff}\} \tag{4-17}$$

同时，本文参考 Transformer 中的多头注意力机制，通过并行计算多个独立的注意力子空间，来增强编码器的特征提取能力。多头注意力的计算公式如下：

$$\mathrm{MHA}\left(Q_i^{\partial},K_j^{\partial},V_j^{\partial}\right)=\mathrm{Concat}\left(h_i^{head_1},h_i^{head_2},...,h_i^{head_M}\right)W_{Att}^{O} \tag{4-18}$$

其中W_{Att}^{O}是可学习参数，M=8 是注意力机制的头数。其中，每一个单头注意力是通过将各种异构的注意力相加得到的，如公式（4-19）所示：

$$h_i^{head_m}=\alpha_{ij}V_j+\sum_{\partial}\sum_{j}\alpha_{ij}^{\partial}V_j^{\partial},\quad m\in\{1,2,...,M\} \tag{4-19}$$

根据注意力的类型，将不同的注意力头加到不同的节点上去。例如，由取货节点构建出来的注意力头，只作用于取货节点上，对送货节点的影响设置为 0。

编码器是一个 L 层 Transformer 结构的网络，且各层之间参数不共享，第 l 层网络的计算公式如下：

$$\begin{aligned}h_i^{l}&=\mathrm{BN}^{l}(h_i^{l-1}+\mathrm{MHA}_i^{l}(Q_i^{\partial},K_j^{\partial},V_j^{\partial}))\\ h_i^{l}&=\mathrm{BN}^{l}(h_i^{l}+\mathrm{FF}^{l}(h_i^{l}))\end{aligned} \tag{4-20}$$

最后，通过最大池化层将节点特征进行聚合以获取全局上下文的表征 h^g 。

4.2.3.2 路线特征嵌入模块

鉴于异构车型问题的特性，在时刻 t，构成当前解决方案的若干条车辆配送路线之间，在车辆类型、配送总成本等方面存在差异，这些信息对于优化异构车型场景下的配送方案具有重要作用。因此，节点选择解码器在决策时，不应当仅根据从节点维度构建出来的特征信息，还应当结合当前解中各条配送路线的特征。本文设计了路线特征嵌入模块，作为解码器的输入之一。

针对配送方案中的每条路线，其原始的特征信息如公式（4-21）所示：

$$H^{R_k}=[C_{R_k},w_{R_k},V_{R_k},maxd_{R_k}] \tag{4-21}$$

其中，特征 C_{R_k} 表示路线 R_k 所使用车辆的核载， w_{R_k} 表示路线 R_k 所使用车辆的运输

成本价，V_{R_k} 表示路线 R_k 上所花费的总成本，$maxd_{R_k}$ 表示路线 R_k 上的最大前后节点相邻距离。在模型迭代求解的任意时刻，对于解中的没有实际配送安排的虚拟路线，其 C_{R_k} 和 w_{R_k} 两个特征的值为当前该路线所对应车型的相关参数，而 V_{R_k} 和 $maxd_{R_k}$ 两个特征的值均设置为 0。

将所有的 K 条路线的原始特征连接，通过参数 W^R 和 b^R 线性映射后，经一个前馈层处理，得到路线特征嵌入 H^R：

$$\begin{aligned} \hat{H}^R &= \left\{ H^{R_k} \right\}_{k=1}^{K} \\ H^R &= \mathrm{FF}\left(W^R \hat{H}^R + b^R \right) \end{aligned} \tag{4-22}$$

4.2.3.3 节点选择解码器

节点选择解码器的核心思想是通过注意力机制建立路线特征与取送货节点特征之间的关联，并据此生成节点选择的概率分布。

首先将全局上下文表征 h^g 融合到各个节点的特征中，对节点特征进行增强，得到 $\hat{h}_i^L$。将编码器输出的特征矩阵划分成取货节点特征矩阵 $\hat{H}_P = \{\hat{h}_i^L\}_{i=1}^{n}$ 和送货节点特征矩阵 $\hat{H}_D = \{\hat{h}_i^L\}_{i=n+1}^{2n}$，分别计算两种节点与路线特征的注意力分数。对于取货节点特征与路线特征嵌入的注意力计算如下：

$$\begin{aligned} Q_P &= W_P^Q \hat{H}^R, K_P = W_P^K \hat{H}_P \\ \alpha_{ij}^P &= \mathrm{softmax}\left(\frac{Q_P^i \cdot K_P^j}{\sqrt{d_k}} \right) \end{aligned} \tag{4-23}$$

α_{ij}^P 表示路线 R_i 与取货节点 j 之间的注意力分数。类似的，计算送货节点与路线特征嵌入的注意力：

$$\begin{aligned} Q_D &= W_D^Q \hat{H}^R, K_D = W_D^K \hat{H}_D \\ \alpha_{ij}^D &= \mathrm{softmax}\left(\frac{Q_D^i \cdot K_D^j}{\sqrt{d_k}} \right) \end{aligned} \tag{4-24}$$

对于取送货节点对 $(i, i+n)$，通过融合以上两方面的注意力分数，得到其联合注意力如公式（4-25）所示：

$$\alpha_{i,i+n}^{pd} = \sum_{r=1}^{n} W_{pd}^{R_r} \left(W_{pd}^p \alpha_{r,i}^P + W_{pd}^d \alpha_{r,i+n}^D \right) + b_{pd} \tag{4-25}$$

最后，通过 softmax 生成节点对选择的概率分布：

$$P_{node} = \text{softmax}(\alpha^{pd}) \tag{4-26}$$

4.2.3.4 算子选择解码器

算子选择解码器根据来自编码器的特征信息、来自节点选择解码器选择出的取送货节点对$(i,i+n)$，并结合过去几次迭代时选择的启发式算子及其收益A，输出选择各种启发式算子的概率分布，并据此选出一个合适的启发式算子，用于对$(i,i+n)$进行操作，实现对解的一次更新。

首先，对于算子及其收益这一输入信息，可表示为公式（4-27）的形式：

$$\begin{gathered} A = \{(opt_{t-N}, r_{t-N}), ..., (opt_{t-1}, r_{t-1})\} \\ opt_i \in \{1,2,3,4\}, r_i \in \{-1,1\} \end{gathered} \tag{4-27}$$

在时间步 t，opt_{t-N} 表示在 t-N 步时所使用的启发式算子编号，r_{t-N} 表示其对应的收益，若实现了对优化目标的正向优化则为 1，否则为-1。

其次，通过可学习的参数矩阵将取送货节点对$(i,i+n)$的特征h_i^L和h_{i+n}^L进行融合得到$\hat{h}_{i,i+n}^L$，并使用来自编码器的全局上下文表征h^g和来自路线特征嵌入模块的全局路线上下文信息h^r，对融合后的特征进行增强$h_{i,i+n}^L = [\hat{h}_{i,i+n}^L, h^g, h^r]$，通过可学习参数将$h_{i,i+n}^L$的维度映射到 128 维。

然后，将算子及其收益A与增强后的节点对特征$h_{i,i+n}^L$进行拼接，并将拼接后的向量经过两个全连接层，其中第一层为 128 维并以 ReLU 为激活函数，第二层输出 4 维的向量$\tilde{Y}^A$，并使用 softmax 生成选择四种算子的概率分布P_{opt}：

$$P_{opt} = \text{softmax}(\tilde{Y}^A) \tag{4-28}$$

4.3 实验及分析

4.3.1 实验环境和参数设置

本章的策略网络同样使用 PPO 算法的 Actor-Critic 变体进行训练，具体过程可参考 3.2.5 节中的描述，在此不再赘述。本章的模型实现以及相关实验所使用的计算机软硬件环境配置，与第三章 3.3.1 节中的一致。模型的超参数设置如表 4-5 所示。

表 4-5 模型的超参数设置

超参数	参数描述	值
dim	高维向量维度	128
L	编码器层数	3
N	过去 *N* 步使用过的算子	$\frac{1}{2}n$
E	训练轮数	200
B	训练的批次数	100
batch_size	训练的批次大小	128
κ	训练的小批次数量	4
T	模型对解的迭代改进次数	3000
n-step	PPO 中的累计奖励步数	5
ϵ	PPO 的裁剪阈值	0.1
β	学习率衰减系数	0.98
η_θ	策略网络参数的学习率	5×10^{-5}
η_ϕ	价值网络参数的学习率	2×10^{-5}

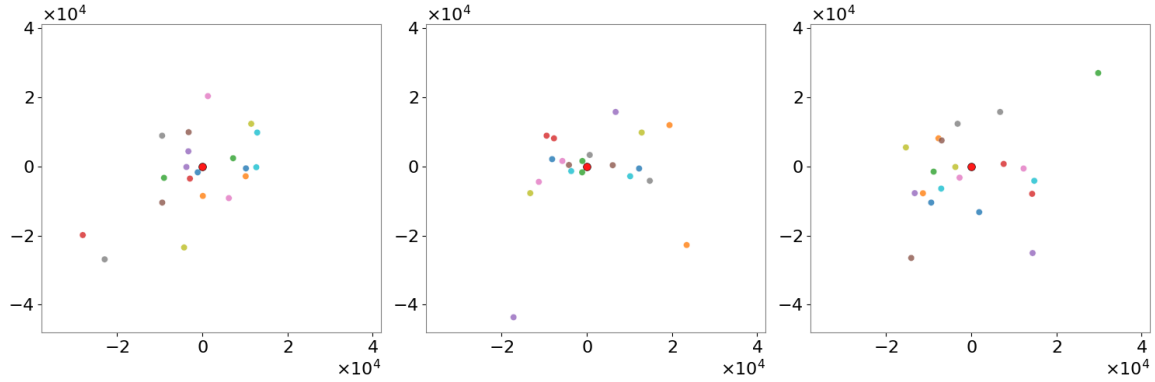
4.3.2 数据集

针对 HFPDPTW 问题的求解模型，本文分别在基于企业真实物流订单数据构建的 HFPDPTW 数据集、公开数据集共两种数据集上进行了实验验证与分析。

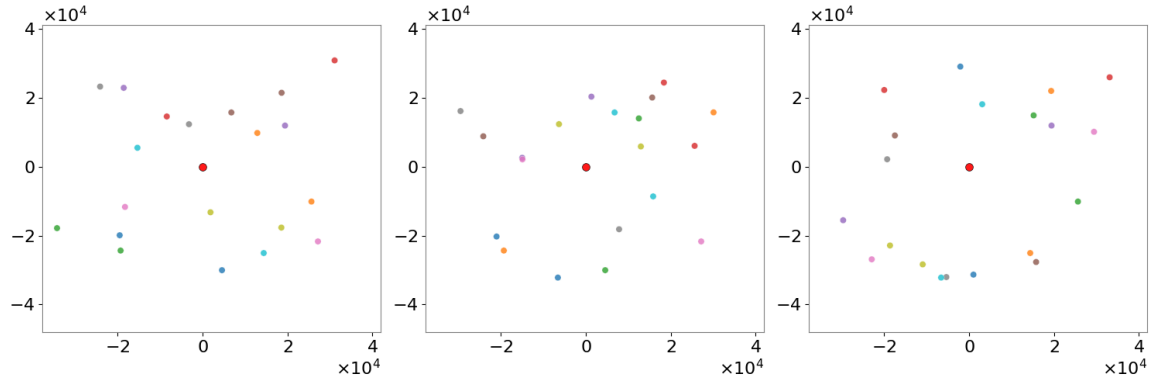
首先，与 VRPSPDTW 问题一样，由于运筹学领域的 HFPDPTW 数据实例在数量方面不能满足深度强化学习模型的训练所需，本节基于物流企业的真实订单数据，并结合领域内常用的数据集生成方法，构建 HFPDPTW 问题数据集。构建方式与 VRPSPDTW 数据集的类似，但对于 HFPDPTW 问题，本文不仅构建了不同客户规模的数据实例，还构建了不同客户分布的实例。根据客户点分布的聚集程度不同，分成聚集分布、分散分布以及混合分布三种。每种分布的数据集都有客户点数量为 20、50 和 80 三种规模的实例。

如图 4-4 所示，以 20 个客户点的规模为例，给出了三种分布的实例可视化图各三个。图中所有客户点分布于以配送中心为原点建立的二维直角坐标系内，横纵坐标值反

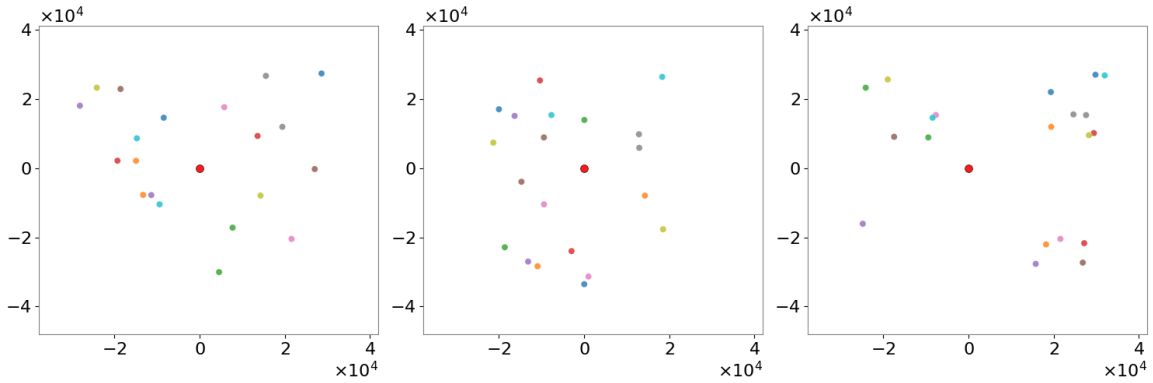
映客户点与配送中心的相对地理位置，坐标单位为米，图中用相同颜色标注出了属于同一订单的一对取送货节点。针对任意一种规模和分布，本文为每轮训练生成 12800 个实例，并划分成 100 个批次，同时生成 1280 个测试实例用于验证模型的推理效果。



a) 聚集分布的数据实例



b) 分散分布的数据实例



c) 混合分布的数据实例

图 4-4 HFPDPTW 问题 20 规模的数据实例可视化图

其次，在 Gasque 等人[40]的研究中，通过对 Ropke 等人[6]设计的 PDPTW 基准实例进行改造，为其构建异构车队，获得了若干 HFPDPTW 问题实例。该数据集共包含 60 个数据实例，虽不足以用于深度强化学习模型的训练，但本文遵循 Ropke 等人[6]构造

PDPTW 基准实例的方法来生成训练数据。

因此，在该公开数据集上，本文通过扩充数据集生成训练数据，并将模型用于 Gasque 等人[40]的测试实例的求解，与现有的算法进行对比。该数据集包含 AA、BB、CC 和 DD 共 4 类实例，每类实例 15 个，其中各个实例的客户规模各不相同。不同类型实例的区别在于客户时间窗宽窄、异构车型数量以及车辆核载不同。其中，AA 类和 BB 类实例的时间窗比其他两类的窄，BB 类和 DD 类的车型数量比其他两类多、车辆核载比其他两类大。

4.3.3 对比算法

在本节中，对各种对比算法做简要介绍：

（1）VNS[86]：一种可变邻域搜索算法；

（2）h_LAHC[84]：一种加强延迟爬山算法；

（3）ALNS[40]：在 Gasque 等人[40]的研究中，问题场景既包含 HFPDPTW，也有在此基础上增加客户需求的拆分配送的场景。该文献提出了一种基于自适应大邻域搜索的元启发式算法，为方便起见，这里记为 ALNS；

（4）N2S[66]：一种基于深度强化学习的邻域搜索算法，通过两个解码器自动学习取送货节点对的重定位启发式操作，在求解 PDP 问题上取得了较好的效果。

鉴于 N2S 原文中没有引入异构车型约束，在构造 N2S 的初始解时遵循与本文相同的方法，生成包含全部可用车辆的初始解，并在迭代改进过程中通过设置虚拟配送中心保持所有车辆的路线信息，使算法能够提取异构车型相关特征信息，有效处理 HFPDPTW 问题。

4.3.4 对比实验与分析

4.3.4.1 现实场景数据集上的对比实验

本文在不同客户点分布、不同客户数量规模的问题实例上进行了充分的对比实验。对比实验结果如表 4-6 至表 4-8 所示。

表中的 size 表示数据实例的客户数量规模；Cost 表示算法的优化目标值，即物流配送成本，单位是元；Time 表示算法的求解时间，单位是秒。所有数据均是 1280 个测试实例上的求解结果取平均得到的。

表 4-6 HFPDPTW 聚集分布数据实例上的对比结果

算法	size=20		size=50		size=80	
	Cost	Time	Cost	Time	Cost	Time
VNS	7.04×10^3	42.91	1.32×10^4	102.66	1.91×10^4	325.21
h_LAHC	6.84×10^3	19.93	1.22×10^4	77.52	1.85×10^4	127.16
ALNS	6.76×10^3	<0.01	1.20×10^4	6.37	1.76×10^4	224.47
N2S	6.87×10^3	1.16	1.23×10^4	3.13	1.83×10^4	5.85
Ours	6.73×10^3	2.65	1.20×10^4	6.71	1.78×10^4	45.88

表 4-7 HFPDPTW 分散分布数据实例上的对比结果

算法	size=20		size=50		size=80	
	Cost	Time	Cost	Time	Cost	Time
VNS	7.78×10^3	41.80	1.88×10^4	102.35	2.88×10^4	324.15
h_LAHC	7.64×10^3	19.35	1.84×10^4	78.21	2.85×10^4	128.35
ALNS	7.63×10^3	<0.01	1.84×10^4	4.94	2.71×10^4	259.31
N2S	7.75×10^3	1.14	1.86×10^4	3.09	2.82×10^4	5.79
Ours	7.64×10^3	2.63	1.81×10^4	6.65	2.74×10^4	45.35

表 4-8 HFPDPTW 混合分布数据实例上的对比结果

算法	size=20		size=50		size=80	
	Cost	Time	Cost	Time	Cost	Time
VNS	7.54×10^3	42.94	1.68×10^4	103.68	2.90×10^4	328.09
h_LAHC	7.48×10^3	18.87	1.63×10^4	78.35	2.80×10^4	129.69
ALNS	7.31×10^3	0.01	1.62×10^4	7.22	2.78×10^4	275.02
N2S	7.44×10^3	1.21	1.65×10^4	3.16	2.84×10^4	5.88
Ours	7.22×10^3	2.66	1.55×10^4	6.70	2.62×10^4	45.81

通过综合观察表 4-6 至表 4-8，可做以下几点分析：

（1）无论何种数据规模与分布类型，本文算法的求解质量均优于 VNS、h_LAHC 两种启发式算法和 N2S 算法。例如，在聚集分布下，当规模 size=80 时，本文算法的 1.78×10^4 较 VNS 的 1.91×10^4 提升 6.81%，较 N2S 的 1.83×10^4 提升 2.73%。在分散分布下，规模 size=50 时，本文算法的 1.81×10^4 较 h_LAHC 的 1.84×10^4 提升 1.63%。而在混合分布下，

规模 size=20 时，本文算法的 7.22×10^3 较 N2S 的 7.44×10^3 优化了 3.00%。

（2）相较于其中最有竞争力的 ALNS 算法，本文算法依然能取得优势。在聚集分布与分散分布数据集上，本文算法与 ALNS 在求解质量上的表现基本持平。例如，聚集分布下 size=50 时，两者基本持平；分散分布下 size=80 时，本文算法仅略差于 ALNS；而在分散分布下 size=50 时，本文算法略优于 ALNS。然而，在混合分布的数据集上，本文算法的优势全面凸显，在各种规模的数据实例上，求解质量均优于 ALNS，可见本文算法在处理复杂客户节点分布时具有较大的优势。

（3）在求解效率上，本文算法大幅领先启发式算法。由于本文算法采用固定的 3000 次迭代的优化策略，所以即使在小规模问题中也需消耗一定时间进行推理，虽高于 ALNS 和 N2S，但其绝对时间开销仍处于可接受范围，均在 3 秒以内完成求解。随着问题规模增大，本文算法的高效性逐步显现，在 size=80 的混合分布实例中，求解时间 45.81 秒较 VNS 的 328.09 秒缩短了 86.04%，较 h_LAHC 的 129.69 秒缩短了 64.68%，较 ALNS 的 275.02 秒缩短了 83.34%，可见传统启发式算法难以满足实时性需求。

（4）与同属于深度强化学习的 N2S 算法相比，本文算法的求解效率虽不及 N2S，但求解质量在所有数据集上均更优。这种优势源于本文提出的协同决策机制，其通过对客户点以及启发式算子的协作决策，能够更全面的探索复杂场景中的动作空间，充分发挥深度强化学习算法的潜力。

综上所述，本文算法在求解质量和求解效率方面均表现出显著优势，证明了算法的先进性和高效性。

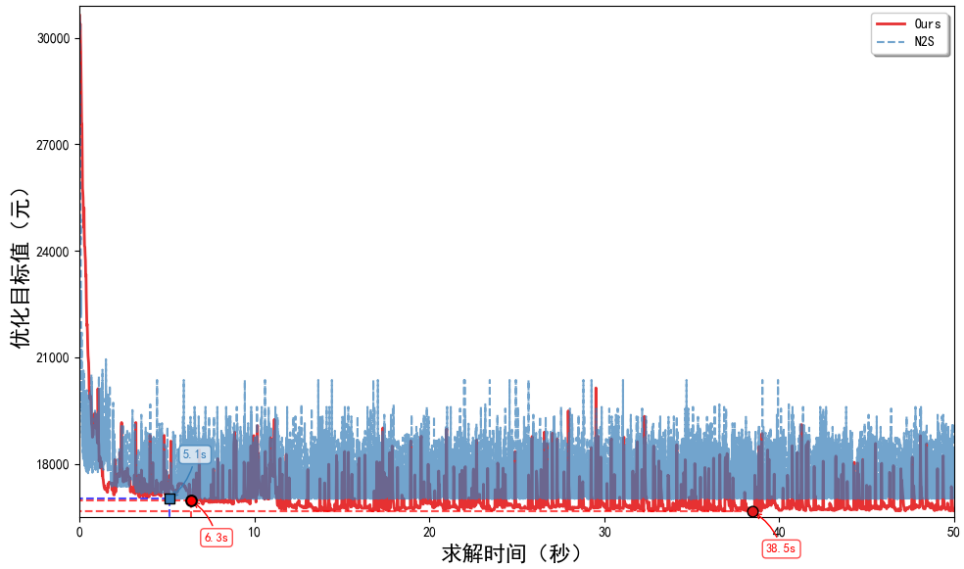


图 4-5 算法的推理曲线图

如图 4-5 所示，绘制出了 N2S 与本文算法在 size=50 规模的混合分布数据实例上的推理过程图，以直观的展示算法的求解效果。为了比较的公平性，将 N2S 的迭代次数调大，使其在求解时间上与本文算法在迭代次数 T=3000 时所需时间基本保持一致，图中展示了两种算法在相同的求解时间下的效果。图中标注出了两种算法第一次找到各自的最优解的时间，以及本文算法第一次找到优于 N2S 的解的时间。由于 N2S 迭代一次所需时间短，相同时间段内其可以迭代的次数远多于本文算法，因此折线图 4-5 中 N2S 的折线会较为密集。通过观察图 4-5，并结合表 4-8 中的对比数据可知，尽管在迭代次数相等时，本文算法求解所需时间比 N2S 长，但本文算法可在非常相近的时间内找到 N2S 所能求得的最优解。同时，即使延长 N2S 的求解时间，在两种算法保持相同求解时间的情况下，N2S 依然无法达到本文算法的求解效果，可见本文算法的先进性。

4.3.4.2 公开数据集上的对比实验

表 4-9 AA 类数据实例上的对比结果

数据	h_LAHC		ALNS		N2S		Ours	
实例	Cost	Time	Cost	Time	Cost	Time	Cost	Time
AA05	**10184.80**	7.42	**10184.80**	<0.01	**10184.80**	0.88	**10184.80**	1.32
AA10	**10383.60**	18.63	**10383.60**	<0.01	**10383.60**	1.14	**10383.60**	2.47
AA15	**20542.40**	25.77	**20542.40**	1.02	**20542.40**	1.83	**20542.40**	3.16
AA20	**20757.90**	58.45	**20757.90**	3.25	**20757.90**	2.72	**20757.90**	5.26
AA25	**21041.02**	76.10	**21041.02**	3.93	21041.47	3.08	**21041.02**	6.85
AA30	31116.86	85.21	**31116.00**	132.53	31117.04	3.81	31116.55	17.88
AA35	31278.83	115.55	**31277.10**	250.17	31279.15	4.74	**31277.10**	27.39
AA40	31450.75	130.61	31448.72	269.68	31450.31	5.81	**31447.38**	45.68
AA45	31670.14	155.09	31666.64	410.50	31669.35	6.57	**31663.84**	62.17
AA50	41742.48	210.05	**41733.77**	470.73	41742.59	7.53	**41733.77**	83.63
AA55	41950.23	265.69	**41930.94**	395.04	41948.80	9.32	41936.83	91.84
AA60	42139.29	284.34	42108.37	533.92	42136.17	11.57	**42102.88**	104.07
AA65	42252.33	395.83	42238.62	361.32	42254.10	14.08	**42228.79**	123.84
AA70	52481.26	507.35	**52448.47**	396.14	52478.37	18.44	**52448.47**	164.71
AA75	52621.32	574.50	**52575.14**	341.20	52633.69	25.20	52588.61	183.95

表 4-10 BB 类数据实例上的对比结果

数据	h_LAHC		ALNS		N2S		Ours	
实例	Cost	Time	Cost	Time	Cost	Time	Cost	Time
BB05	**10339.70**	7.73	**10339.70**	<0.01	**10339.70**	0.85	**10339.70**	1.27
BB10	**20512.50**	18.05	**20512.50**	<0.01	**20512.50**	1.20	**20512.50**	2.35
BB15	**20667.60**	32.87	**20667.60**	0.27	**20667.60**	1.78	**20667.60**	3.34
BB20	**20786.00**	62.83	**20786.00**	2.44	20786.90	2.63	**20786.00**	5.56
BB25	20947.06	77.42	**20946.10**	4.83	20949.35	3.15	**20946.10**	6.44
BB30	31040.63	92.59	**31036.74**	119.75	31043.54	3.93	**31036.74**	18.82
BB35	31234.15	112.05	31225.02	133.19	31248.79	4.78	**31224.82**	27.09
BB40	31454.86	128.65	**31447.33**	182.55	31461.58	5.77	31449.25	46.76
BB45	41532.11	179.25	**41522.38**	432.54	41541.35	6.58	41525.71	61.42
BB50	41683.47	224.50	41652.56	526.86	41685.16	7.48	**41651.25**	82.51
BB55	41864.31	245.35	41853.74	514.77	41892.82	9.33	**41848.93**	91.39
BB60	62369.75	282.49	**62344.23**	591.10	62399.05	11.71	62347.51	104.75
BB65	62665.61	410.33	62635.18	573.24	62722.17	14.69	**62628.27**	122.85
BB70	62994.34	526.28	**62920.75**	471.95	63032.11	19.36	62925.64	165.10
BB75	63135.68	571.14	**63031.06**	318.28	63215.04	24.97	63055.17	183.53

表 4-11 CC 类数据实例上的对比结果

数据	h_LAHC		ALNS		N2S		Ours	
实例	Cost	Time	Cost	Time	Cost	Time	Cost	Time
CC05	**10212.50**	8.36	**10212.50**	<0.01	**10212.50**	0.96	**10212.50**	1.44
CC10	**10394.30**	20.73	**10394.30**	<0.01	**10394.30**	1.34	**10394.30**	2.71
CC15	**20554.90**	28.04	**20554.90**	0.66	**20554.90**	2.17	**20554.90**	4.29
CC20	**20721.40**	55.27	**20721.40**	17.54	**20721.40**	2.86	**20721.40**	5.87
CC25	20908.87	80.61	**20908.84**	47.39	20910.35	3.13	**20908.84**	6.92
CC30	31021.46	89.19	**31010.22**	87.16	31028.43	3.88	**31010.29**	17.15
CC35	31124.21	118.54	**31105.03**	308.65	31122.82	4.81	31109.11	28.14
CC40	31284.34	127.50	31276.37	262.58	31282.72	5.74	**31274.28**	45.34

CC45	31425.75	165.94	31419.68	337.11	31439.37	6.55	**31415.85**	62.19
CC50	41655.15	205.40	**41628.81**	519.97	41653.56	7.62	41632.51	82.78
CC55	41771.37	243.69	41745.03	484.03	41782.94	9.51	**41743.68**	92.13
CC60	42014.49	284.13	41997.75	490.63	42031.45	12.32	**41988.25**	103.60
CC65	52182.74	374.66	52147.64	469.92	52215.83	14.58	**52144.57**	123.56
CC70	52377.18	513.08	**52353.58**	480.05	52392.52	19.13	52361.81	164.37
CC75	52561.69	581.45	**52519.82**	442.34	52582.50	25.85	52526.21	184.90

表 4-12 DD 类数据实例上的对比结果

数据	h_LAHC		ALNS		N2S		Ours	
实例	Cost	Time	Cost	Time	Cost	Time	Cost	Time
DD05	**10324.80**	8.61	**10324.80**	<0.01	**10324.80**	0.89	**10324.80**	1.35
DD10	**20567.80**	19.98	**20567.80**	<0.01	**20567.80**	1.17	**20567.80**	2.58
DD15	**20616.40**	35.37	**20616.40**	0.15	**20616.40**	1.89	**20616.40**	4.35
DD20	**20679.20**	65.30	**20679.20**	12.06	20680.05	2.94	**20679.20**	5.91
DD25	20902.93	80.55	**20902.34**	69.98	20904.78	3.34	**20902.34**	6.61
DD30	21017.05	93.67	**21015.07**	90.19	21017.93	4.05	**21015.15**	18.54
DD35	31089.52	121.18	31083.52	184.49	31093.65	4.73	**31080.82**	26.49
DD40	31212.08	126.46	31202.17	266.71	31214.47	6.21	**31197.56**	45.87
DD45	31326.14	184.45	31310.91	382.50	31332.61	6.58	31316.93	62.81
DD50	31452.49	236.72	31430.18	514.18	31463.57	7.51	**31425.46**	82.14
DD55	31613.85	265.53	**31581.62**	534.31	31634.16	9.19	31587.82	91.04
DD60	41794.36	286.32	**41762.33**	503.68	41815.05	11.68	41771.34	104.26
DD65	42146.30	427.88	42129.81	494.75	42153.43	14.13	**42127.22**	123.49
DD70	42276.29	554.02	42228.23	540.22	42280.52	18.65	**42223.79**	165.66
DD75	42484.24	578.84	42462.66	458.32	42528.95	25.42	**42450.25**	184.02

本文对公开数据集上的 60 个测试实例进行了求解。该数据集的实例以“所属类型+取送货订单数量”命名，如 AA05 表示一个 AA 类型的、包含 5 个取送货订单的实例，即该实例包含 10 个客户点，在客户规模上对应本文 size=10 的实例。

为了保证公平性，对于每一个实例，本文使用各个算法对其求解 10 次，记录 10 次

求解结果。如表 4-9 至 4-12 所示，Cost 为优化目标值，取各自的最优值进行呈现。Time 列表示平均每次求解所花费的时间，其单位为秒。表中加粗的数据表示几种对比算法中求得的最优解以及与最优解差距在 0.1 以内的求解结果。

通过观察表 4-9 中数据可知，本文算法在 12 个实例上取得了成本最低的解决方案，而 h_LAHC、ALNS 以及 N2S 三种算法分别在 5 个、11 个和 4 个实例上取得较优的解，均不敌本文算法。同时，即使在没有取得最优解的几个实例上，如 AA40、AA45、AA60、AA70 和 AA75，本文算法的求解质量也非常接近最优解，并且均优于 h_LAHC 和 N2S 两种算法。同理，通过观察表 4-10 至 4-12，本文算法分别在 10 个、11 个和 12 个实例上取得了最优解，而 ALNS 则分别为 11 个、10 个和 9 个，其余两种算法则均在 5 个以下。可见，本文算法在求解质量方面优于各种对比算法。

在求解效率方面，本文绘制出了在 AA 类数据实例上，算法的求解时间随问题规模的变化曲线图，如图 4-6 所示。从图中可以直观的得出，在面临大规模问题时，启发式算法的求解时间急剧增长，而深度强化学习算法则依然保持高效性。

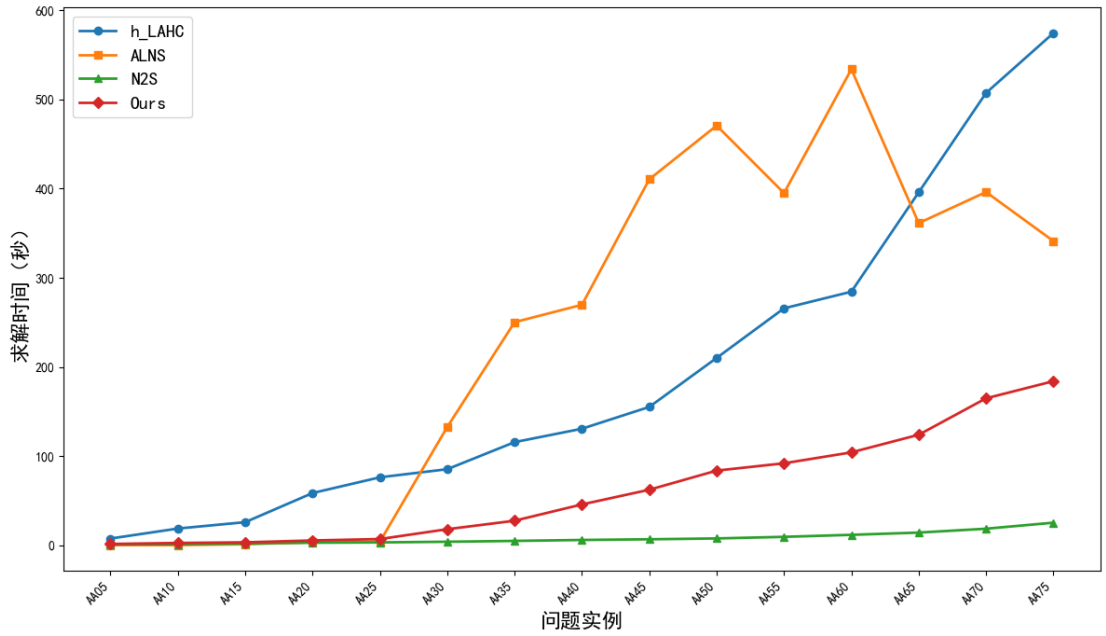


图 4-6 各种算法求解时间随问题规模增长的折线图

4.3.5 消融实验与分析

在本小节中，通过消融实验分别验证本文设计的异构注意力机制、路线特征嵌入模块以及协同决策解码器的有效性。首先，本文通过将编码器中的包含了十种类型的异构注意力，分别替换为普通的多头注意力、Li 等人[48]的具有七种类型的异构注意力，通过

对比来验证本文的异构注意力的表征能力。如表 4-13 所示，展示了在混合分布的各种规模的数据实例上的消融实验结果，表中的数据均是在生成的 1280 个测试实例上进行求解并取平均值得到。

分析表中数据可以得到，在各种规模的数据实例上，本文设计的 10-ATT 机制，相较于 7-ATT 机制分别提升了 0.82%、1.27%和 2.25%，相较于普通多头注意力分别提升了 1.50%、3.13%和 4.74%。可见，普通多头注意力虽然可以提高求解效率，其在 size=80 规模的实例求解时可以比本文的 10-ATT 平均节省 7.62 秒的求解时间，但其求解质量与两种异构注意力差距较大。对比两种异构注意力机制，本文的 10-ATT 优于 7-ATT，原因就在于 10-ATT 不仅能捕获取不同取送货节点之间的相互关系，还能识别不同车辆路线之间复杂的节点关系，这对考虑了异构车型的问题场景来说至关重要。由此可见，本文设计的异构注意力机制能够提高模型在异构车型取送货场景下的表征能力。

表 4-13 异构注意力机制的消融实验结果

算法	size=20		size=50		size=80	
	Cost	Time	Cost	Time	Cost	Time
ATT	7.35×10^3	2.19	1.60×10^4	5.91	2.74×10^4	38.24
7-ATT	7.30×10^3	2.53	1.57×10^4	6.44	2.67×10^4	43.05
Ours	7.24×10^3	2.66	1.55×10^4	6.68	2.61×10^4	45.86

表 4-14 路线特征嵌入模块的消融实验结果

算法	size=20	size=50	size=80
Ours w/o Route	7.31×10^3	1.62×10^4	2.77×10^4
Ours	7.22×10^3	1.55×10^4	2.62×10^4
提升	1.23%	4.32%	5.42%

其次，针对路线特征嵌入模块的消融实验，本文将该模块去除，直接使用编码器输出的全局上下文信息代替路线特征嵌入，用于两个解码器中的相关计算。表 4-14 中展示了在混合分布的数据实例上的实验结果，其中用 Ours w/o Route 表示去除了该模块后的模型。通过分析可得，引入路线特征嵌入模块后，算法在 size=20、size=50 和 size=80 规模实例上的求解质量分别提升了 1.23%、4.32%和 5.42%，由此可见该模块对于辅助解码器进行决策的重要作用。

然后，针对解码器，为了验证本文的协同决策解码器对于求解质量和求解效率的提

升，本文参考现有文献中的设计方法，对本文的解码器模型进行修改，开展消融实验。一方面，参考一些通过深度学习模型来学习单一启发式的文献，如 DACT 算法学习 2-opt 启发式，N2S 算法学习 Relocate 启发式等，将本文的算子选择解码器分别替换为单一的启发式算子 Swap 和 Relocate，保留节点选择解码器，从而使得模型整体上表现为学习单一启发式的形式。另一方面，参考一些通过深度学习模型作为算子选择器的文献，将本文的节点选择解码器分别替换为随机解码器（Random）和贪婪解码器（Greedy），以及去除该解码器只保留算子选择解码器，从而使得模型表现为专注于选择启发式算子的形式。解码器的消融实验结果如表 4-15 所示，展示了在 size=50 规模的混合分布数据实例上的实验结果，结果数据依然是在生成的 1280 个测试实例上的求解结果取平均得到的。

表 4-15 协同决策解码器的消融实验结果

节点选择解码器	算子选择解码器	T=3000		T=5000	
		Cost	Time	Cost	Time
Ours	Swap	1.65×10^4	7.05	1.64×10^4	12.69
Ours	Relocate	1.68×10^4	7.13	1.63×10^4	12.93
/	Ours	1.82×10^4	5.54	1.74×10^4	9.24
Random	Ours	1.84×10^4	5.51	1.74×10^4	9.23
Greedy	Ours	1.77×10^4	5.76	1.69×10^4	9.75
Ours	Ours	1.55×10^4	6.72	1.53×10^4	10.25

分析消融实验结果可知，在相同的迭代次数下，第二类模型的求解质量是最差的，相较于本文的模型分别下降了 17.42%、18.71%和 14.19%。第一类模型的求解质量虽不及本文的模型，但差距相对较小。若调大迭代次数至 5000 次，相较于迭代 3000 次时的求解质量，第一类模型的提升效果不明显，分别提升了 0.61%和 2.30%，这是因为对于 HFPDPTW 这样的复杂场景，单一的启发式算子难以实现对解邻域空间的高效探索。而第二类模型的求解质量提升相对较为明显，分别提升了 4.40%、5.43%和 4.52%，可见这类模型虽然缺乏精心设计的节点选择解码器，导致模型求解效率低下，但可以通过延长求解时间来发掘模型的潜力。当由本文设计的协同决策双解码器同时完成节点和启发式算子的选择时，模型的综合求解效果是最优的。

最后，通过泛化性实验，以此评估本文模型对不同规模问题的适应性。本文基于客

户点混合分布的数据集，将特定规模数据训练出来的模型用于求解其他规模的实例，实验结果如图 4-7 所示。通过观察可得，所有模型在规模 size=20 的实例上的求解结果均优于 VNS 算法的 7.54×10^3，在 size=50 实例上的最差求解结果 1.69×10^4 仅与 VNS 的 1.68×10^4 有 0.60%的微小差距，即使用 size=20 的数据训练出来的模型求解 size=80 的实例，其结果也仅与 VNS 的 2.90×10^4 相差 6.2%，可见本文的模型在不同规模的问题上的良好泛化性能。

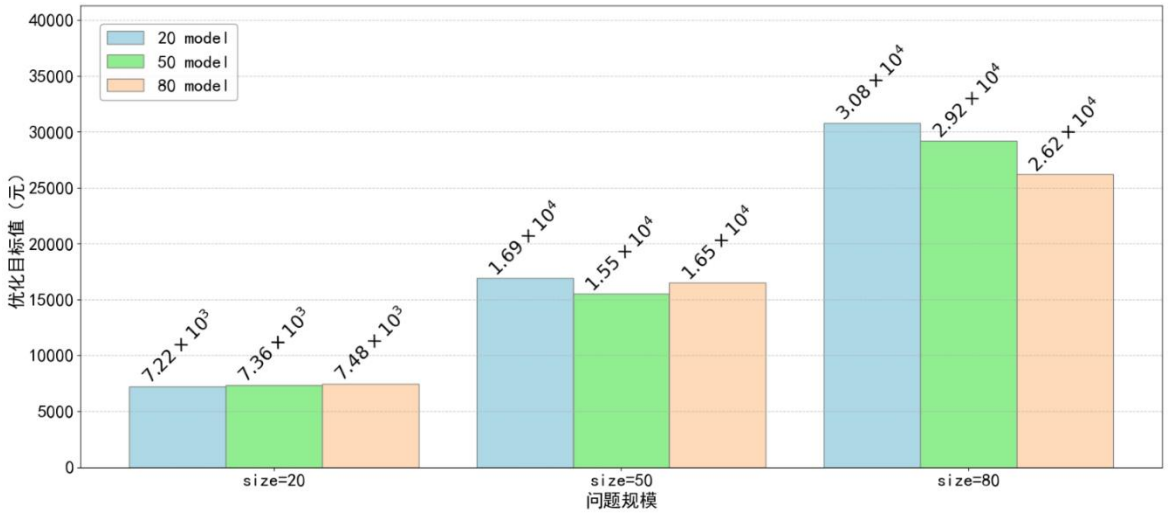


图 4-7 模型的泛化性实验结果

4.4 本章小结

本章在建立 HFPDPTW 问题数学模型的基础上，提出了一种基于深度强化学习的改进式求解模型。本章详细介绍了模型的策略网络通过一个异构注意力机制捕捉不同客户节点之间的多样化相互关系，并使用两个解码器协同决策客户点和启发式算子的选择，实现对解空间的高效探索。本章分别基于公开数据集和构建的现实场景数据集进行实验，结果表明了求解模型的先进性、高效性。