通过自动启发式设计中的稳健求解器演化加速动态路线重新求解

Yizhuo Ma[1], Haoran Sun[1],Shuang Liang[1],Muquan Li[1],Qizhi Chen[1], Renzhong Wang[1],Nan Chen[2],Lunxu Liu[2],Ke Qin[1*]中国电子科技大学，中国成都[2]四川大学华西医院胸外科{myz, 2023091203011, muquanli2023, chenqizhi, 202611090902}@std.uestc.edu.cn；{shuangliang, qinke}@uestc.edu.cn； {dr.chennan,liulunxu}@wchscu.cn

arXiv:2611.32576v1 [cs.NE] 24 Nov 2026

摘要——多目标路线规划利用服务位置的空间数据和两两旅行成本，确定能够最小化总旅行成本的访问顺序。该任务可形式化为旅行商问题（TSP），并受益于基于大语言模型的自动启发式设计（AHD），以构建有效的求解器。尽管现有方法在静态场景中表现有效，但当位置变化、新请求或请求取消在执行过程中改变底层请求数据时，现有方法会表现出显著的重新求解低效。我们的实证分析表明，“静态过拟合”是导致这一低效的因素之一，因为演化出的求解器在动态变化后可能反复将搜索引向重叠的局部邻域。我们进一步发现，这种“过拟合”与现有演化过程的局限性有关：在某些实例上较低的目标值可能伴随着更长的求解时间。因此，本文提出 ERDR，这是一种即插即用框架，将两种策略整合到现有自动启发式设计流程中，引导演化过程生成能够更快地重新求解动态变化实例的求解器。具体而言：1）ERDR 使用验证引导的重新选择，在常规演化中未纳入的实例上重新评估候选解，从而减少对固定训练实例的专门化。2）ERDR 应用运行时鲁棒性增强评估，在 AHD 方法原有评估目标的基础上，加入求解时间的组间变异性和尾部风险，以抑制代价高昂且不均衡的搜索。在多个 TSP 规模和动态场景下开展的实验表明，ERDR 在引导式局部搜索下实现了平均重新求解加速比 2.2×，在迭代局部搜索下实现了平均重新求解加速比 4.4×。†

一、引言

多目标路线规划是基于位置的服务中的核心计算任务，例如配送和现场作业。它使用描述目标位置和旅行成本的空间数据来确定能够最小化旅行成本的访问顺序，这一任务可表述为旅行商问题（TSP） [1],[2]。鉴于TSP的NP难性 [3], , 在实际计算预算内求解此类问题通常依赖启发式求解器来获得高质量路径。然而，为不同问题设置开发有效的启发式算法并进行配置，需要大量专业知识和工作量 [4]。基于Transformer架构 [5], , 现代大语言模型（LLM）展现出广泛的少样本能力 [6] 和更强的指令遵循能力 [7], , 包括从自然语言规范 [8], [9]。这些进展促成了自动启发式设计(AHD)方法 [10], [11]，该方法在进化框架内通过迭代生成、修订和评估来发现有效的求解器。

现有基于大语言模型的 AHD 方法通常在节点集和位置固定的静态数据上演化启发式算法。如图1(a)所示，演化出的求解器会为已知服务节点规划一条巡回路线，随后车辆按顺序沿路线行驶并为这些节点提供服务。在路径执行过程中，服务请求的变化会通过未访问服务位置的变动、新请求以及请求取消来更新底层空间数据

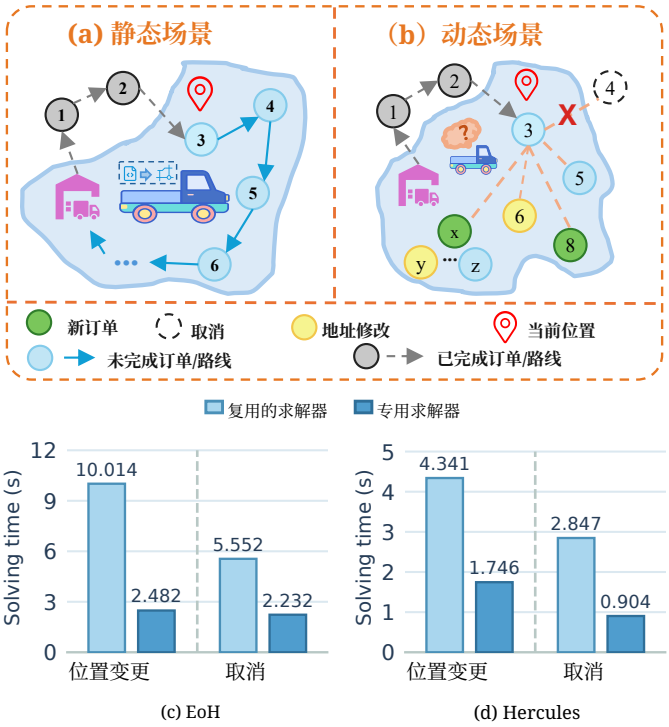

图1。（a）静态执行，此时AHD进化求解器规划一条巡回路线，并在整个过程中执行，不发生实例变化。（b）动态执行，此时位置变化、新请求或请求取消会改变剩余的路径规划问题，并需要重新使用进化得到的求解器进行重新求解。（c）和（d）分别比较了在路径进度达到20%时、面对位置变化和请求取消的情况下，在50个TSP500实例上，GLS中由EoH和Hercules生成的复用求解器与专用求解器的平均路径重新求解时间。

* Ke Qin是通讯作者。†代码可在 https://github.com/myz12138/ERDR获取。

。 [12]如图1(b)所示，此时必须利用更新后的信息，从车辆当前位置重新优化剩余路线。此类路线更新是动态路径规划的核心，在现实世界的运营中十分普遍，包括快递配送和现场维护。 [13]例如，禁忌搜索 [14] 会在新的客户请求到达且车辆正在途中时重新优化快递路线。RpALNS [15] 采用并行自适应大邻域搜索，在新的服务请求变得已知时更新技术人员的路线。然而，我们通过引导式局部搜索（GLS，详情见第VI-E节）进行的研究表明，与在变化后的数据上演化出的对应求解器相比，直接复用的求解器需要显著更多的求解时间。这一额外的求解成本会延迟对请求数据变化的路线更新。如图1(c)–(d)所示，在 TSP500 上，当巡回路线进度达到20%时发生位置变化和请求取消，使用 EoH/Hercules进化求解器直接重新求解该巡回路线，平均耗时是使用其专门化对应求解器的3.3/2.8倍。

我们在第IV-A节的分析表明，静态过拟合是上述重新求解低效的成因之一：进化过程倾向于选择针对静态源实例结构量身定制的启发式算法，而这些算法在实例变化后可能难以维持稳定的搜索行为。在重新求解过程中，复用的求解器反复将搜索引向重叠的局部邻域，因此需要付出额外的搜索成本才能获得高质量解。为了解这种“过拟合”如何在进化过程中形成，我们指出了两个关键发现：1）解质量的提升越来越集中在少数源实例上，因此即使大多数实例获益甚微，整体质量仍能提升。2）这些提升主要出现在求解时间较长的实例上，表明质量提升伴随着日益高昂且不均衡的搜索。这些发现表明，现有演化过程倾向于保留那些收益局限于少数实例、且需要越来越高搜索成本的求解器，从而导致重新求解低效。我们将在第IV-B节详述具体内容。

因此，我们提出了 ERDR（通过稳健求解器演化实现动态场景中的高效重求解），这是一种可与现有 AHD 方法集成的即插即用框架。鉴于验证集通常用于机器学习，以减轻对训练数据的“过拟合”，ERDR 首先引入验证引导的重新选择，以减少候选解对特定实例的专门化。在该机制下，实例被划分为互不相交的训练组和验证组，在训练组上进行常规演化，并在验证组上重新评估候选解，以更新种群。ERDR 进一步引入运行时鲁棒性增强评估，在 AHD 方法原有评估目标的基础上，加入组间变异性和求解时间中的尾部风险。这些标准将相关计算成本纳入进化选择，从而促进跨实例的稳定且高效的搜索。这两种策略共同构成了一种稳健的评估与选择机制，旨在引导动态场景中的巡回路线重新求解时，提高求解器可迁移性和搜索稳定性。

我们的贡献总结如下：

1）我们发现，在动态旅行商问题场景中重新求解巡回路线会产生显著开销：在静态条件下进化得到的求解器，在路径执行期间发生实例变化后被复用时，求解时间会显著延长。2）我们的分析表明，“静态过拟合”是导致这一问题的因素之一，并显示基于均值的评估使得那些质量提升集中于少数实例、且搜索成本不断增加的求解器仍能在进化过程中保持竞争力。3）我们提出 ERDR，这是一种即插即用的评估框架，结合验证引导的重新选择，以减少对固定训练实例的专门化，并采用运行时鲁棒性增强评估来加速动态重新求解。4）在多个动态旅行商问题场景中的实验表明，ERDR 可集成到不同的 AHD 方法中，在 GLS 下实现平均巡回路线重新求解加速比2.2×，在 ILS 下实现 4.4×。

II. 相关工作

a) 自动启发式设计：经典旅行商问题启发式算法（如引导式局部搜索（GLS） [16] 和 Lin–Kernighan 搜索 [17] ）依赖专家设计的 [18] 邻域和更新规则 [19]。超启发式 [20] 在启发式算法空间中进行搜索，而 FunSearch [21] 则将大语言模型生成的代码 [22]与系统评估 [23]相结合。近期的 AHD 方法采用了不同的进化机制 [24]。EoH [10] 协同进化想法与代码，ReEvo [11] 利用反思反馈，Hercules [25]从精英启发式算法中推导搜索方向，而 MEoH [26]采用多目标选择。其他研究则通过蒙特卡洛树搜索、联合算法-模型进化，以及后见信息和前瞻信息提示 [27]–[29]来拓展探索范围。这些方法在固定实例上评估候选解，但未考察实例变化后的求解器复用。

b) 旅行商问题中的动态场景：动态路径规划方法会在执行过程中获取新信息后更新路径 [13]。现有路径可以根据更新后的请求进行调整，并作为进一步优化的初始解。旅行商问题重新优化研究了节点插入与删除场景，并使用原始实例的最优巡回路线 [30]。基于插入的路径规划将新到达的请求纳入现有路径 [31]。其他方法则在变化环境中保留信息。显式记忆存储先前的解以供复用 [32]，，而基于蚁群优化的方法将搜索信息与局部改进相结合 [33]。基于方案库的蚁群优化方法会为反复出现或相似的节点位置配置维护预先计算的路径方案，并将其复用机制与 2-opt 局部搜索相结合 [34]。这些研究通过人工设计的启发式算法和复用机制来调整路径。相比之下，我们的工作重点是使大语言模型自动设计的启发式算法能够在路径执行过程中位置或请求发生变化时得到高效复用。

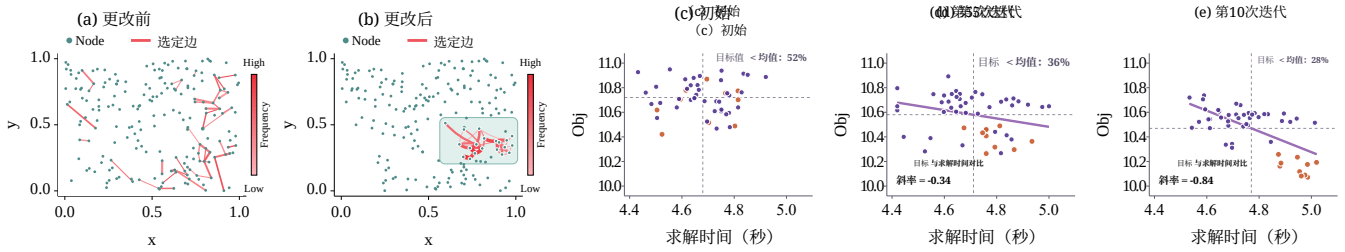

图2。GLS中重新求解低效的实验观察。(a)–(b)分别展示动态变化前后的边选择，这些数据是在静态TSP200上使用通过EoH进化得到的EoH求解器，经过至多500次GLS迭代记录的。(a)展示原始的200节点实例，而(b)展示路径进度达到20%后的剩余任务，其中新增节点数量为未访问节点数的20%。点表示TSP节点，边的深浅和宽度表示求解过程中的选择频率。(c)–(e)展示50个源TSP200实例在初始阶段以及第5次和第10次进化迭代时，目标值与求解时间的实例级联合分布。每个点表示一个实例。在(d)和(e)中，实线表示目标值与求解时间之间的拟合关系，旁边的标签报告相应的斜率。灰色虚线标记各阶段的平均目标值和平均求解时间，标签报告低于平均目标值的实例比例。橙色点标识第10次迭代中低目标值高耗时子集中的实例，并在全部三个检查点中追踪这些相同的实例。

III. 预备知识

a) 基于大语言模型的自动启发式设计：给定一个规定任务和所需函数接口的提示，大语言模型会利用选定父代及其评估反馈，迭代生成或修订候选启发式 [10], [11]。每个候选启发式都嵌入固定的搜索框架中，从而形成用于巡回优化的启发式求解器 [16]。生成的求解器会在一组实例上进行评估，其性能将指导候选项选择以及后续生成过程，从而逐步演化出有效的启发式算法。

形式化地，令 $D_s = \{x_i\}_{i=1}^{N}$ 表示源实例集，其中 $N = 1$ 对应单实例评估。给定一个候选启发式 h，搜索框架使用 h为 D_s 中的每个实例获取一条巡回路线，而任务特定评估$\mathbf{E}(h; D_s)$则汇总其性能。在第 t 次迭代中，令 $\mathcal{P}_t$ 表示当前种群，$\mathcal{Q}_t$ 表示新生成或修订后的候选启发式集合。种群更新为

$$\mathcal{P}_{t+1} = \mathrm{Select}_M\left(\mathcal{P}_t \cup \mathcal{Q}_t; \mathbf{E}(\cdot; D_s)\right), \tag{1}$$

其中，Select_M 根据评估结果保留 M 候选启发式。所选种群及其评估结果共同指导下一次迭代中的候选启发式生成。

b) 动态重新求解：当位置变化、新请求或请求取消在路径执行期间改变未服务请求集合或其位置时，必须对剩余任务进行重新优化。已执行前缀保持不变，而剩余任务需要从当前位置出发，经过更新后的未访问节点，前往原始起点。搜索框架从一条热启动路径开始，初始化其搜索状态，并复用此前在静态实例上演化得到的启发式算法，以优化剩余任务。

形式化地，令 $h^\star$ 表示在源实例集 D_s 上经过进化后选出的启发式算法，π_i 表示源实例 x_i 的执行路径，Δ_i 表示路径进度为 ρ 时发生的变化事件。相应的残余实例为

$$x_{i,\rho}^{\mathrm{res}} = R(x_i, \pi_i, \rho, \Delta_i), \tag{2}$$

其中，R 根据执行状态和更新后的节点信息构建剩余任务。集合 $D_\rho = \{x_{i,\rho}^{\mathrm{res}}\}_{i=1}^{N}$ 形成由路径执行诱导的目标实例集。演化后的启发式算法 $h^\star$ 用于求解这些残余实例，以评估其在动态变化后的性能。

IV. 观察

以下分析首先探究为什么重新求解更新后的实例会变得低效，随后考察求解器进化中导致这一现象的关键因素。

A. 观察*1*：为何低效?

为探究图1中重新求解更新后的实例时求解时间为何会增加，我们分析这一过程中的求解器搜索行为。我们选取在静态TSP200实例上进化得到的最佳EoH启发式算法，并在GLS中重复使用它，以便在路径进度达到20%后重新求解剩余任务；此时新增节点数为未访问节点数的20%。图2（a）展示了原始200节点实例上的边选择，图2（b）展示了变化后的实例上的边选择，线条越粗表示选择频率越高。这些边集中在一个小区域内，表明搜索反复访问重叠的局部邻域。这种行为说明，“静态过拟合”可能导致重新求解低效，因为静态进化会使启发式算法偏向源实例中的结构模式，从而可能增加实例变化后获得高质量解所需的搜索开销。因此，进化得到的求解器可能会专门适应静态场景，并在实例变化后难以维持稳定的搜索行为。

B. 观察*2*。“过拟合”是如何出现的?

我们通过追踪 EoH 在 TSP200 进化不同阶段所保留的求解器，研究“静态过拟合”是如何出现的。在不同迭代中，我们在相同的 50 个训练实例上评估每个检查点得到的进化求解器。对于每个实例，我们记录求解时间和目标值。图2（c）–（e）可视化了这三个检查点上这两项指标的联合分布，并揭示了两点观察：1）在三个检查点中，明显的质量提升出现在越来越小的训练实例子集上，而大多数实例则表现出

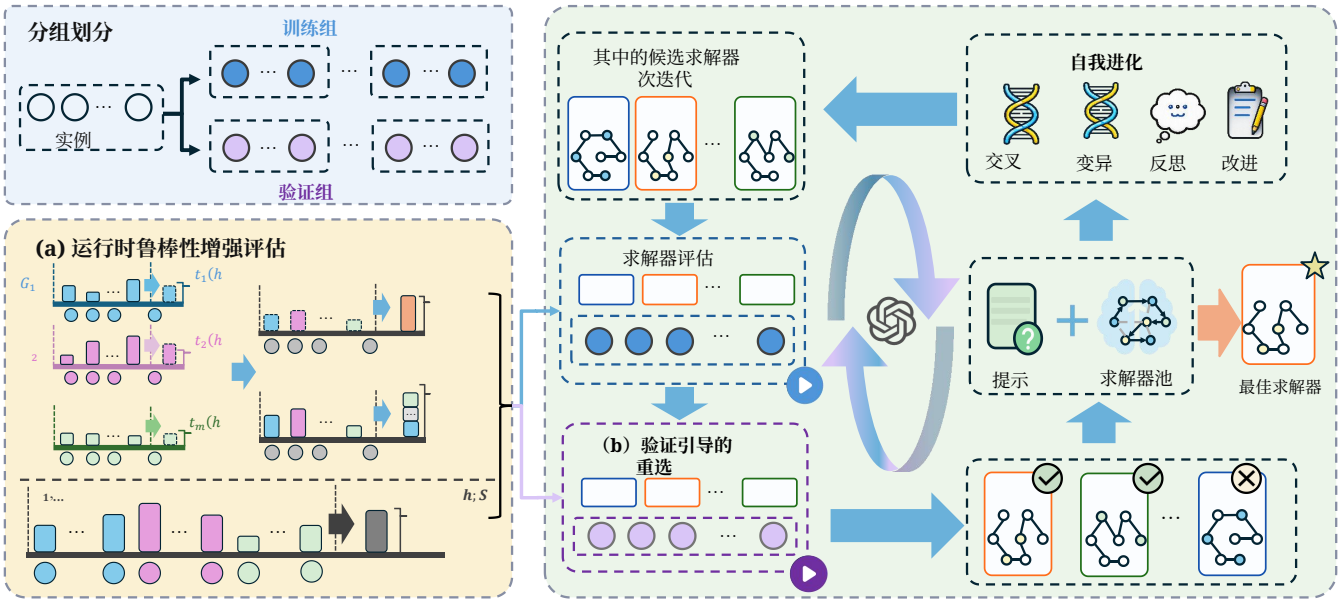

图 3. ERDR 框架概览。训练组和验证组分别用于求解器评估和验证引导的重新选择。(a) 求解器评估和重新选择中使用的原始目标，加入了组间变异性和求解时间尾部风险。(b) 验证引导的重新选择会重新评估候选解，以便在下一轮进化前更新求解器池。

仅有微小变化，甚至出现质量下降；2）较大的质量提升主要出现在需要更长求解时间的实例上，并且这一模式在各迭代中逐渐变得更加清晰。这些结果表明，现有进化模式对那些收益集中于狭窄实例子集、并伴随搜索成本增加的求解器惩罚较弱，从而使它们在进化过程中仍能保持竞争力。

V. 方法论

受上述两项缺陷的启发，ERDR 在现有 AHD 框架中加入验证引导的重新选择和运行时鲁棒性增强评估，以减少求解器对固定训练实例的“过拟合”。验证引导的重新选择会在一组单独的数据上重新评估候选求解器；这组数据不参与常规演化，从而减少求解器对演化过程中反复使用的实例产生专门化。评估目标还纳入了组间变异性和尾部风险，以鼓励候选项选择兼顾搜索稳定性。

A. 验证引导的重新选择

借鉴模型选择中用于抑制过拟合的训练-验证划分 [35], ，我们将实例划分为互不相交的训练组和验证组，分别用于常规演化和重新选择。给定一个源实例集D_s，我们将其随机划分为 K 组，记为$\mathcal{B}=\{\mathcal{G}_1,\ldots,\mathcal{G}_K\}$，并将它们分配到训练划分 $\mathcal{B}_{\text{train}}$和验证划分 $\mathcal{B}_{\text{val}}$ ，使得

$$\mathcal{B}=\mathcal{B}_{\text{train}}\cup\mathcal{B}_{\text{val}}. \tag{3}$$

$$\mathcal{B}_{\text{train}}\cap\mathcal{B}_{\text{val}}=\emptyset. \tag{4}$$

由当前种群生成的新候选求解器最初在 $\mathcal{B}_{\text{train}}$上进行评估。每当底层 AHD 方法更新其种群时，我们就根据相同的标准，在 $\mathcal{B}_{\text{val}}$ 上评估进入幸存者选择阶段的候选解，并使用其验证结果选择幸存者。演化结束后，我们从最终种群中选择在 $\mathcal{B}_{\text{val}}$ 上具有最高 DDS 分数的启发式算法，用于动态测试。

B. 运行时鲁棒性增强评估 *on*

求解器在不同问题实例上的表现可能存在显著差异，这是算法选择研究的核心前提 [36]。因此，我们引入两个标准——组间变异性和求解时间尾部风险——以惩罚那些在不同实例组中导致搜索开销不均或过高的候选解。这些标准扩展了每种集成 AHD 方法原有的评估目标，使运行时鲁棒性能够影响候选项选择。形式化地，设 $\mathcal{S}=\{\mathcal{G}_1,\ldots,\mathcal{G}_m\}$表示待评估的各组， $t_q(h)$表示求解器使用启发式算法 h在 $\mathcal{G}_q$中的实例上求解一条巡回路线所需的平均时间。我们将组间变异性定义为

$$\sigma_t(h\mid\mathcal{S})=\left(\frac{1}{m}\sum_{q=1}^{m}\left(t_q(h)-\frac{1}{m}\sum_{r=1}^{m}t_r(h)\right)^2\right)^{1/2}, \tag{5}$$

该指标会惩罚巡回路线求解时间在不同组之间波动剧烈的候选解，从而优先选择在不同实例子集中搜索效率稳定的候选解。

受条件风险价值背后的上尾聚合原理 [37],启发，我们通过对组求解时间进行排序来定义尾部风险

将组求解时间设为 $t_{(1)}(h) \geq \cdots \geq t_{(m)}(h)$，并设置 $r = \lceil m/2 \rceil$:

$$w_t(h \mid \mathcal{S}) = \frac{1}{r} \sum_{q=1}^{r} t_{(q)}(h), \tag{6}$$

该方法会对较慢组中的高求解时间施加惩罚。

令$\mathbf{E}^{\text{orig}}(h;\mathcal{S})$表示集成 AHD 方法的原始评估目标，包括其传统的解质量目标和任何方法特定目标。ERDR 按如下方式将组间变异性和尾部风险纳入这些目标。

$$\mathbf{z}(h \mid \mathcal{S}) = \left[\mathbf{E}_{\text{orig}}(h;\mathcal{S})^{\top}, \sigma_t(h \mid \mathcal{S}), w_t(h \mid \mathcal{S})\right]^{\top}. \tag{7}$$

增强后的目标保留了原始标准，同时在候选解比较过程中明确考虑组间变异性以及较慢组中的求解时间。这两个新增标准将相关计算成本纳入进化选择。

假设所有目标均采用最小化形式，候选解根据增强目标 [38], [39] 之间的帕累托支配关系进行排序。对于两个候选解 h_i 和 h_j，定义

$$\begin{aligned} h_i \prec_{\mathcal{S}} h_j \iff & \mathbf{z}(h_i \mid \mathcal{S}) \leq \mathbf{z}(h_j \mid \mathcal{S}) \\ & \wedge\ \mathbf{z}(h_i \mid \mathcal{S}) \neq \mathbf{z}(h_j \mid \mathcal{S}), \end{aligned} \tag{8}$$

其中，不等式按分量解释。关系 $h_i \prec_{\mathcal{S}} h_j$ 表示 h_i 帕累托支配 h_j: h_i 在任何标准上都不劣，并且至少在一个标准上严格更优。

幸存者选择优先保留非支配候选解，并按照降序支配–差异度评分（DDS），遵循 MEoH [26], 对每个类别中的候选解进行排序。DDS 相同时，依次采用集成 AHD 方法的原始排序规则、尾部风险和组间变异性进行裁决。候选解按照此顺序保留，直至达到种群预算。

C. 框架概述

ERDR 被设计为用于求解器进化的插件式框架，其整体工作流程如图3所示。从初始种群出发，底层 AHD 方法通过其原有进化算子生成新的候选求解器。ERDR 依据所提出的选择策略评估扩展后的候选池，并保留有潜力的求解器进入下一代。在每次种群更新时，进入幸存者选择的候选解会在验证实例上接受评估，其验证结果决定幸存者，随后进化继续进行。

VI. 实验

A. 数据集

对于每种规模 $n \in \{200, 500\}$，我们生成 50 个合成欧几里得旅行商问题实例，节点坐标从 $[0,1]^2$，中独立均匀采样，之后按照 EoH [10] 和 ReEvo [11] 进行演化。对于每种规模，我们将实例随机划分为 10 个大小相等的组，常规演化与验证引导的重新选择默认按 5:5 分配（每次运行时固定）。我们还在 TSPLIB [40], 上进行动态测试。TSPLIB 是一个标准的旅行商问题基准库，包含真实应用实例（第 VI-N 节）。

B. 动态场景

我们在全部 50 个源实例上评估九种动态旅行商问题场景，将表 I 中的三种变化类型与 20%、50% 和 80% 的路径进度相结合。位置变化会修改现有请求的坐标数据，而新请求和请求取消则会在未服务集合中添加或移除请求。我们默认使用 $\sigma = 0.02$ 和 $r_1 = r_2 = 20\%$。对于原始巡回 $\pi = (v_1, \ldots, v_n)$，我们在访问节点的比例达到 $\rho \in (0,1)$ 之后触发动态变化。计入起始节点后，已执行前缀包含 $k_\rho = \lceil \rho n \rceil$个已访问节点，且 $1 \leq k_\rho \leq n-1$。我们仅对未访问集合 $U_\rho = \{v_{k_\rho+1}, \ldots, v_n\}$应用表 I 中的变化，并保持已执行前缀固定。我们重新优化从当前位置 $s = v_{k_\rho}$ 经过变化后的集合 U'_ρ 到原始起点 $t = v_1$的路径，并添加一个虚拟节点 d，以形成具有边成本的闭合巡回旅行商问题

$$c(d,u) = c(u,d) = \begin{cases} 0, & u \in \{s,t\}, \\ M, & u \in U'_\rho, \end{cases} \tag{9}$$

其中 M 是一个较大的有限惩罚值，超过经过所有剩余节点的可行 s-到-t 路径的成本。我们保留真实节点距离，并要求 s 和 t 作为返回的巡回路线中 d 的两个相邻节点。我们在距离统计和启发式更新中排除虚拟边，并在整个搜索过程中保留其规定成本。零成本连接 $t \to d \to s$ 闭合路径，移除 d 后即可恢复成本相同的 s-到-t路线。

表 I 三种动态变化的构造。每项操作均应用于巡回进度截断点之后的未访问集合。

变化	操作
位置变化	向每个节点的两个坐标添加标准差为 σ 的独立零均值高斯噪声，然后将扰动后的坐标裁剪到 $[0,1]^2$。节点数保持不变。
新请求	添加最多$\{1,\text{round}(r_1\lvert U_\rho\rvert)\}$ 个节点，这些节点独立地从并均匀地从 $[0,1]^2$ 采样到未访问集合中，其中 r_1 表示节点添加比例。
请求取消	移除从未访问集合中均匀无放回抽取的最多$\{1,\text{round}(r_2\lvert U_\rho\rvert)\}$ 个节点，地从未访问集合中无放回抽取，其中 r_2 表示节点移除比例。移除数量为上限为 $\lvert U_\rho\rvert - 2$，且保留节点的坐标保持不变。

对于每个源实例，我们在执行由 Concorde 生成的静态巡回路线 [41] 期间采样事件，以构建共享的动态实例。我们使用相同的调整后剩余巡回路线对所有方法进行热启动：对于位置变化保留访问顺序，移除已取消的节点，并在新请求使路线长度增幅最小的位置依次插入节点。

C. 评估指标

我们测量目标值（Obj.）和求解时间。目标值是返回路线的总长度，使用

表 II 在 GLS 和 ILS 下处理动态场景时，TSP200 和 TSP500 的平均求解时间（秒）。20%、50% 和 80% 这几组列表示发生动态变化时已访问的节点比例。位置变更、新请求和取消请求分别表示位置变化、新请求和请求取消。对于每种场景，首先对每次运行中配对成功实例的求解时间取平均。我们报告三次运行中这些实例平均求解时间的均值 ± 标准差。在每组基线对中，对于每种规模和场景，均以粗体显示均值较低的结果及其标准差。

	方法	20%			50%			80%		
		位置变更↓	新请求↓	取消 ↓	位置变更↓	新请求↓	取消 ↓	位置变更↓	新请求↓	取消 ↓
GLS										
TSP200	EoH	6.443±0.709	9.139±1.106	4.412±0.437	2.757±0.276	4.025±0.443	1.860±0.167	0.588±0.053	0.782±0.077	0.443±0.036
	+ERDR	**2.673±0.176**	**3.806±0.276**	**2.547±0.151**	**0.443±0.027**	**0.508±0.034**	**0.361±0.019**	**0.220±0.012**	**0.242±0.014**	**0.189±0.009**
	MEoH	1.420±0.209	2.009±0.070	0.930±0.054	0.582±0.041	0.822±0.073	0.381±0.014	0.102±0.006	0.144±0.010	0.067±0.004
	+ERDR	**0.986±0.039**	**1.349±0.073**	**0.672±0.042**	**0.439±0.045**	**0.601±0.061**	**0.299±0.012**	**0.091±0.004**	**0.124±0.006**	**0.062±0.002**
	ReEvo	1.346±0.123	1.954±0.230	0.880±0.096	0.535±0.059	0.783±0.075	0.367±0.036	0.109±0.021	0.153±0.012	0.078±0.007
	+ERDR	**0.566±0.054**	**0.822±0.052**	**0.373±0.016**	**0.236±0.007**	**0.336±0.026**	**0.161±0.010**	**0.049±0.003**	**0.068±0.005**	**0.035±0.002**
	Hercules	1.079±0.059	1.553±0.094	0.691±0.034	0.428±0.021	0.614±0.034	0.282±0.013	0.079±0.004	0.111±0.005	0.054±0.002
	+ERDR	**0.432±0.017**	**0.622±0.026**	**0.279±0.010**	**0.180±0.006**	**0.251±0.010**	**0.118±0.004**	**0.034±0.002**	**0.047±0.002**	**0.023±0.002**
TSP500	EoH	10.014±0.218	10.018±0.304	5.552±0.262	3.224±0.646	4.987±1.033	2.171±0.369	0.516±0.101	0.896±0.147	0.401±0.064
	+ERDR	**2.447±0.334**	**3.134±0.532**	**1.476±0.191**	**1.031±0.104**	**1.578±0.155**	**0.793±0.066**	**0.414±0.028**	**0.489±0.037**	**0.348±0.019**
	MEoH	8.101±0.824	10.436±0.171	5.301±0.667	3.317±0.432	4.690±0.361	2.171±0.137	0.582±0.037	0.822±0.157	0.381±0.029
	+ERDR	**5.270±0.261**	**7.212±0.393**	**3.591±0.160**	**2.348±0.106**	**3.213±0.159**	**1.600±0.065**	**0.486±0.020**	**0.664±0.030**	**0.331±0.012**
	ReEvo	1.853±0.465	1.700±0.651	0.626±0.262	0.453±0.147	0.573±0.230	0.218±0.072	0.087±0.024	0.096±0.044	0.046±0.020
	+ERDR	**0.668±0.146**	**0.921±0.236**	**0.399±0.082**	**0.263±0.050**	**0.357±0.080**	**0.168±0.030**	**0.053±0.010**	**0.076±0.014**	**0.035±0.006**
	Hercules	4.341±0.239	6.355±0.484	2.847±0.141	1.717±0.086	2.546±0.110	1.104±0.050	0.304±0.018	0.422±0.041	0.188±0.008
	+ERDR	**2.045±0.079**	**3.489±0.148**	**1.345±0.047**	**0.878±0.031**	**1.342±0.052**	**0.697±0.022**	**0.188±0.006**	**0.200±0.007**	**0.090±0.003**
ILS										
TSP200	EoH	3.421±0.431	4.441±0.252	2.609±0.206	2.044±0.118	2.455±0.238	1.738±0.102	1.140±0.192	1.197±0.065	1.030±0.031
	+ERDR	**0.370±0.044**	**0.525±0.061**	**0.313±0.007**	**0.220±0.063**	**0.284±0.066**	**0.175±0.041**	**0.080±0.002**	**0.095±0.006**	**0.064±0.001**
	MEoH	0.277±0.041	0.354±0.046	0.208±0.008	0.148±0.007	0.170±0.012	0.099±0.024	0.050±0.003	0.073±0.004	0.046±0.008
	+ERDR	**0.155±0.020**	**0.189±0.007**	**0.120±0.021**	**0.090±0.010**	**0.100±0.020**	**0.062±0.005**	**0.036±0.002**	**0.051±0.006**	**0.034±0.004**
	ReEvo	6.787±1.140	8.956±0.884	6.112±0.593	4.642±0.903	5.701±0.252	3.935±0.234	2.012±0.349	2.447±0.218	1.670±0.037
	+ERDR	**1.005±0.031**	**1.406±0.066**	**0.735±0.128**	**0.534±0.035**	**0.689±0.026**	**0.395±0.052**	**0.187±0.036**	**0.224±0.008**	**0.147±0.016**
	Hercules	0.973±0.066	1.449±0.118	0.679±0.125	0.436±0.065	0.607±0.012	0.304±0.024	0.109±0.019	0.141±0.020	0.085±0.011
	+ERDR	**0.288±0.022**	**0.388±0.020**	**0.197±0.012**	**0.137±0.011**	**0.186±0.021**	**0.099±0.012**	**0.043±0.004**	**0.052±0.006**	**0.033±0.006**
TSP500	EoH	8.536±0.362	8.991±1.234	6.594±0.425	4.825±0.376	6.261±0.442	4.128±0.613	2.165±0.330	2.728±0.097	1.913±0.053
	+ERDR	**1.388±0.181**	**1.729±0.169**	**1.178±0.049**	**0.895±0.102**	**1.198±0.083**	**0.792±0.086**	**0.514±0.028**	**0.568±0.069**	**0.444±0.047**
	MEoH	1.555±0.175	1.997±0.091	1.127±0.133	0.771±0.122	0.932±0.121	0.519±0.047	0.242±0.030	0.371±0.042	0.214±0.002
	+ERDR	**1.015±0.041**	**1.216±0.062**	**0.791±0.108**	**0.556±0.083**	**0.610±0.058**	**0.391±0.042**	**0.198±0.023**	**0.286±0.012**	**0.182±0.007**
	ReEvo	2.031±0.045	2.875±0.057	1.250±0.128	0.776±0.104	1.158±0.183	0.599±0.065	0.210±0.024	0.305±0.028	0.162±0.002
	+ERDR	**1.728±0.025**	**2.337±0.224**	**0.946±0.106**	**0.607±0.053**	**0.838±0.102**	**0.399±0.043**	**0.138±0.012**	**0.175±0.003**	**0.098±0.011**
	Hercules	5.372±0.203	8.881±1.108	3.498±0.161	2.078±0.442	3.196±0.245	1.469±0.091	0.448±0.029	0.639±0.071	0.324±0.025
	+ERDR	**1.927±0.229**	**2.564±0.134**	**1.086±0.171**	**0.719±0.096**	**0.989±0.051**	**0.490±0.053**	**0.186±0.033**	**0.230±0.003**	**0.144±0.021**

原始距离矩阵计算得出。在动态评估中，它表示从当前位置出发，经过更新后的未访问节点，最终到达原始起点的剩余路线长度。我们报告每个场景中所有评估实例的平均目标值。在进化和动态评估中，我们测量指定停止准则下从开始到结束的实际搜索时间，其中包括启发式调用和搜索操作，但不包括初始路线构建。目标值越低、耗时越短越好。

D. 基线

我们将四种具有代表性的 AHD 方法——EoH [10], ReEvo [11], Hercules [25], 和 MEoH [26], ——与其 ERDR 变体配对，在保留求解器生成和进化调度的同时，在候选项选择中加入运行时稳健评估和验证引导的重新选择。

E. 搜索框架与算子

在实验中，我们在两个搜索框架 GLS 和迭代局部搜索（ILS）中评估 ERDR。在 GLS 中， [16], 我们进化 update_edge_distance，将原始距离矩阵、当前巡回路线和边使用计数映射为在扰动期间累积的值，以生成引导 2-opt 和重新定位扰动的修改后距离。2-opt 移动会反转两条被移除的巡回路线边之间的线段，并重新连接巡回路线；而重新定位则会将一个节点重新插入到其他位置。在 ILS 中， [42], 我们进化 perturb_tour，将原始距离矩阵和当前巡回路线映射为具有相同节点集合的扰动后巡回路线。我们使用双桥扰动，切断四条巡回路线边，并以不同顺序重新连接由此产生的线段。在这两个框架中，我们都使用固定的 2-opt 和重新定位局部搜索，在原始距离下改进扰动后的巡回路线。对于 ILS，我们接受更短的巡回路线，并像往常一样从新接受的巡回路线开始下一次搜索迭代。

F. 实验设置

实验在一台 64 位 Windows 11 专业版设备上本地运行，设备配备 Intel Core i5-12500 CPU（6 个核心、12 个线程）和 16GB 内存。我们使用 gpt-6-luna，在每个搜索框架内分别演化启发式算法。对于 TSP200 和 TSP500，GLS 与 ILS 共享 500 次迭代的上限，以及每个实例 10 秒的时间预算，任一限制达到时即停止。我们在每次迭代前检查时间，允许正在进行的迭代完成，并返回找到的最佳巡回路线。GLS 使用

表 III GLS 和 ILS 在动态场景下 TSP200 与 TSP500 的最终目标值均值（目标值）。对于每种场景，先对每次运行中的配对成功实例的目标值取平均，再对三次运行所得的实例均值取平均，得到报告值。场景列与表 II 一致。 Δ 注释标示 ERDR 相对于对应基线的目标值差异。数值越低越好。

	方法	20%			50%			80%		
		位置变更	新请求	取消	位置变更	新请求	取消	位置变更	新请求	取消
GLS										
TSP200	EoH	8.822	10.090	7.885	5.619	7.365	4.965	2.319	4.235	2.045
	+ERDR	$8.946^{\Delta+0.124}$	$10.240^{\Delta+0.150}$	$7.979^{\Delta+0.094}$	$5.709^{\Delta+0.090}$	$7.507^{\Delta+0.142}$	$5.001^{\Delta+0.036}$	$2.338^{\Delta+0.019}$	$4.279^{\Delta+0.044}$	$2.051^{\Delta+0.006}$
	MEoH	9.063	10.248	7.925	5.685	7.534	5.016	2.341	4.155	2.049
	+ERDR	$9.083^{\Delta+0.020}$	$10.251^{\Delta+0.003}$	$7.933^{\Delta+0.008}$	$5.699^{\Delta+0.014}$	$7.550^{\Delta+0.016}$	$5.020^{\Delta+0.004}$	$2.341^{\Delta+0.000}$	$4.160^{\Delta+0.005}$	$2.049^{\Delta+0.000}$
	ReEvo	8.916	10.163	7.876	5.628	7.460	4.980	2.314	4.143	2.047
	+ERDR	$8.935^{\Delta+0.019}$	$10.165^{\Delta+0.002}$	$7.881^{\Delta+0.005}$	$5.632^{\Delta+0.004}$	$7.477^{\Delta+0.017}$	$4.983^{\Delta+0.003}$	$2.313^{\Delta-0.001}$	$4.146^{\Delta+0.003}$	$2.047^{\Delta+0.000}$
	Hercules	9.020	10.207	7.914	5.665	7.533	4.989	2.314	4.142	2.047
	+ERDR	$9.025^{\Delta+0.005}$	$10.210^{\Delta+0.003}$	$7.922^{\Delta+0.008}$	$5.665^{\Delta+0.000}$	$7.529^{\Delta-0.004}$	$4.987^{\Delta-0.002}$	$2.313^{\Delta-0.001}$	$4.146^{\Delta+0.004}$	$2.047^{\Delta+0.000}$
TSP500	EoH	13.965	15.798	12.042	8.840	11.473	7.571	3.635	6.408	3.051
	+ERDR	$14.148^{\Delta+0.183}$	$15.900^{\Delta+0.102}$	$12.207^{\Delta+0.165}$	$9.010^{\Delta+0.170}$	$11.654^{\Delta+0.181}$	$7.641^{\Delta+0.070}$	$3.695^{\Delta+0.060}$	$6.482^{\Delta+0.074}$	$3.065^{\Delta+0.014}$
	MEoH	14.288	16.026	12.232	8.912	11.601	7.624	3.643	6.370	3.059
	+ERDR	$14.437^{\Delta+0.149}$	$16.084^{\Delta+0.058}$	$12.321^{\Delta+0.089}$	$9.060^{\Delta+0.148}$	$11.703^{\Delta+0.102}$	$7.660^{\Delta+0.036}$	$3.678^{\Delta+0.035}$	$6.408^{\Delta+0.038}$	$3.067^{\Delta+0.008}$
	ReEvo	14.036	15.752	12.150	8.907	11.566	7.626	3.628	6.292	3.062
	+ERDR	$14.070^{\Delta+0.034}$	$15.766^{\Delta+0.014}$	$12.165^{\Delta+0.015}$	$8.932^{\Delta+0.025}$	$11.589^{\Delta+0.023}$	$7.629^{\Delta+0.003}$	$3.637^{\Delta+0.009}$	$6.294^{\Delta+0.002}$	$3.064^{\Delta+0.002}$
	Hercules	13.980	15.772	12.092	8.834	11.518	7.597	3.636	6.351	3.057
	+ERDR	$14.154^{\Delta+0.174}$	$15.853^{\Delta+0.081}$	$12.202^{\Delta+0.110}$	$8.994^{\Delta+0.160}$	$11.634^{\Delta+0.116}$	$7.645^{\Delta+0.048}$	$3.674^{\Delta+0.038}$	$6.386^{\Delta+0.035}$	$3.064^{\Delta+0.007}$
ILS										
TSP200	EoH	8.803	10.096	7.808	5.519	7.398	4.901	2.265	4.052	1.990
	+ERDR	$8.895^{\Delta+0.092}$	$10.155^{\Delta+0.059}$	$7.832^{\Delta+0.024}$	$5.535^{\Delta+0.016}$	$7.426^{\Delta+0.028}$	$4.903^{\Delta+0.002}$	$2.265^{\Delta+0.000}$	$4.054^{\Delta+0.002}$	$1.990^{\Delta+0.000}$
	MEoH	9.104	10.266	7.913	5.632	7.507	4.934	2.280	4.078	1.993
	+ERDR	$9.206^{\Delta+0.102}$	$10.338^{\Delta+0.072}$	$7.946^{\Delta+0.033}$	$5.747^{\Delta+0.115}$	$7.555^{\Delta+0.048}$	$4.939^{\Delta+0.005}$	$2.280^{\Delta+0.000}$	$4.090^{\Delta+0.012}$	$1.993^{\Delta+0.000}$
	ReEvo	8.874	10.139	7.829	5.537	7.421	4.903	2.266	4.054	1.990
	+ERDR	$8.824^{\Delta-0.050}$	$10.114^{\Delta-0.025}$	$7.821^{\Delta-0.008}$	$5.520^{\Delta-0.017}$	$7.407^{\Delta-0.014}$	$4.904^{\Delta+0.001}$	$2.265^{\Delta-0.001}$	$4.052^{\Delta-0.002}$	$1.990^{\Delta+0.000}$
	Hercules	8.811	10.092	7.814	5.516	7.401	4.901	2.265	4.052	1.990
	+ERDR	$8.953^{\Delta+0.142}$	$10.201^{\Delta+0.109}$	$7.855^{\Delta+0.041}$	$5.563^{\Delta+0.047}$	$7.446^{\Delta+0.045}$	$4.914^{\Delta+0.013}$	$2.266^{\Delta+0.001}$	$4.057^{\Delta+0.005}$	$1.990^{\Delta+0.000}$
TSP500	EoH	14.103	15.767	12.114	8.821	11.496	7.584	3.641	6.213	3.068
	+ERDR	$14.259^{\Delta+0.156}$	$15.800^{\Delta+0.033}$	$12.185^{\Delta+0.071}$	$8.943^{\Delta+0.122}$	$11.555^{\Delta+0.059}$	$7.612^{\Delta+0.028}$	$3.662^{\Delta+0.021}$	$6.242^{\Delta+0.029}$	$3.073^{\Delta+0.005}$
	MEoH	14.466	16.019	12.336	9.072	11.717	7.628	3.761	6.353	3.069
	+ERDR	$14.497^{\Delta+0.031}$	$16.008^{\Delta-0.011}$	$12.333^{\Delta-0.003}$	$9.073^{\Delta+0.001}$	$11.696^{\Delta-0.021}$	$7.621^{\Delta-0.007}$	$3.756^{\Delta-0.005}$	$6.347^{\Delta-0.006}$	$3.069^{\Delta+0.000}$
	ReEvo	14.158	15.790	12.165	8.873	11.537	7.603	3.644	6.233	3.069
	+ERDR	$14.133^{\Delta-0.025}$	$15.757^{\Delta-0.033}$	$12.140^{\Delta-0.025}$	$8.861^{\Delta-0.012}$	$11.502^{\Delta-0.035}$	$7.586^{\Delta-0.017}$	$3.636^{\Delta-0.008}$	$6.229^{\Delta-0.004}$	$3.069^{\Delta+0.000}$
	Hercules	14.117	15.785	12.157	8.894	11.535	7.601	3.654	6.234	3.071
	+ERDR	$14.204^{\Delta+0.087}$	$15.798^{\Delta+0.013}$	$12.173^{\Delta+0.016}$	$8.907^{\Delta+0.013}$	$11.527^{\Delta-0.008}$	$7.606^{\Delta+0.005}$	$3.652^{\Delta-0.002}$	$6.225^{\Delta-0.009}$	$3.070^{\Delta-0.001}$

每次迭代进行两次扰动移动。对于每个基线对，我们使用相同的搜索设置和评估种子。我们让 EoH 运行十代，每个代中由五个算子各生成四个子代。对于 Hercules 和 ReEvo，我们分别使用 50 和 100 的候选解生成预算，种群规模分别为四，初始种群规模为十。我们将 MEoH 限制为 220 个采样候选解，配备四个采样器和四个评估器。默认情况下，每当底层 AHD 方法更新其种群时，都会执行验证引导的重新选择。

a) 演化过程中的失败处理： 我们为失败的候选评估分配不可行质量惩罚和 $2T$ 的时间惩罚，其中 T 是每实例搜索预算。我们在独立进程中评估候选解，对 N 个实例设置 $\max(300, NT + 15)$ 秒的批处理超时，并排除记录的启动、数据加载和初始化时间。我们终止超时的批次，并拒绝其中的候选解。

b) 动态评估过程中的失败处理： 我们将执行错误、无效路径、非有限成本和看门狗终止计为失败，将预算耗尽但获得有效路径计为成功。对于每个基线–ERDR比较，我们仅在两种方法均成功的相同实例上，对搜索时间和目标值取平均。在所评估的比较中，至少保留了 50 个实例对中的 46 个，观察到的最大排除数量为 4。

G. 主要结果

表 II 和表 III 分别报告了四种 AHD 方法及其在 GLS 和 ILS 下的 ERDR 变体的求解时间和最终目标值。评估涵盖 TSP200 和 TSP500，并包含三种变化类型和三个巡回进度级别。在相同的停止准则下，ERDR 在每种测试配置中都减少了重新求解时间，在 GLS 和 ILS 下分别实现了平均 $2.2\times$ 和 $4.4\times$ 的加速比，而平均最终巡回长度的最大相对增幅仅约为 1.93% 和 2.04%。ILS 下更大的收益可能源于其直接的巡回扰动；与 GLS 中的距离引导扰动相比，这使得运行时间感知型选择能够更直接地影响后续的局部搜索工作量。某些配置在 TSP500 上的运行速度快于 TSP200，这是因为运行时间不仅取决于实例规模，还取决于由分别进化得到的启发式算法所产生的局部搜索工作量。

在不同变化类型中，新请求通常带来最大的绝对时间节省。新请求会扩大剩余问题规模，并引入额外的插入和排序决策，因此高效搜索尤为重要。取消请求和位置变化下持续取得的收益，进一步表明该方法适用于问题规模和几何结构的变化。在不同执行阶段，绝对时间节省

表 IV Hercules+在进度比例为 20%、50% 和 80% 时，GLS 上 TSP500 的 ERDR 求解时间消融结果（秒）。对于每个场景，首先对每次运行中的成功实例的求解时间取平均值，然后将三次运行中的这些实例平均值再取均值作为报告值。括号内报告相对于完整版本的求解时间比。

变体	20%			50%			80%		
	位置变更	新请求	取消	位置变更	新请求	取消	位置变更	新请求	取消
Hercules									
Full	**2.045**	**3.489**	**1.345**	**0.878**	**1.342**	**0.697**	**0.188**	**0.200**	**0.090**
不含 Val.	2.577 (1.26×)	4.187 (1.2×)	1.775 (1.32×)	1.089 (1.24×)	1.597 (1.19×)	0.795 (1.14×)	0.224 (1.19×)	0.232 (1.16×)	0.101 (1.12×)
不计时间$_1$	3.579 (1.75×)	5.234 (1.5×)	2.179 (1.62×)	1.370 (1.56×)	2.120 (1.58×)	0.934 (1.34×)	0.269 (1.43×)	0.330 (1.65×)	0.145 (1.61×)
不含时间$_2$	2.495 (1.22×)	4.605 (1.32×)	1.601 (1.19×)	1.027 (1.17×)	1.785 (1.33×)	0.920 (1.32×)	0.224 (1.19×)	0.236 (1.18×)	0.098 (1.09×)
不含 σ_t	2.536 (1.24×)	4.222 (1.21×)	1.412 (1.05×)	1.045 (1.19×)	1.651 (1.23×)	0.899 (1.29×)	0.250 (1.33×)	0.262 (1.31×)	0.105 (1.17×)
不含 w_t	2.802 (1.37×)	4.954 (1.42×)	1.762 (1.31×)	1.264 (1.44×)	2.040 (1.52×)	0.920 (1.32×)	0.227 (1.21×)	0.262 (1.31×)	0.107 (1.19×)

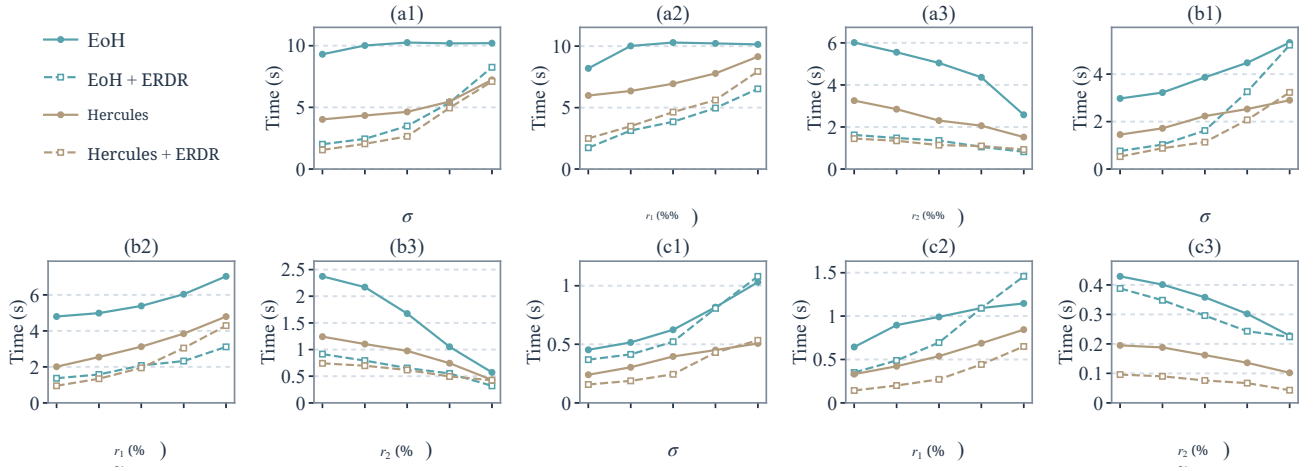

图4。EoH、Hercules 及其 ERDR 变体在 GLSonTSP500 中的平均求解时间。面板 (a1)–(a3)、(b1)–(b3) 和 (c1)–(c3) 分别对应 20%、50% 和 80% 的路径进度。在每组中，面板 1、2 和 3 分别改变其对应动态场景中的坐标扰动 σ、节点添加比例 r_1 和节点移除比例 r_2。

在20%进度时最大，并随着剩余问题规模缩小而减少。在80%进度时，未访问节点较少，因此在相同变化设置下受影响的节点也较少。工作量减少限制了绝对时间节省，而较小的剩余问题也与某些设置下几乎相同的目标值相吻合。收益还取决于基线对效率的重视程度。EoH在两个框架中都实现了最大的平均加速比，而MEoH已在进化过程中纳入运行时间，因此收益较小。这表明，ERDR带来的效益在一定程度上取决于底层搜索和选择机制对求解效率的促进程度。

H. 消融实验

我们在 GLS 中将完整 ERDR 与 Hercules 的五种消融变体进行了比较。不含 Val. 移除了验证引导的重新选择，同时保留两项时间准则；而不含 Time$_1$ 则保留重新选择，但移除这两项准则。不含 Time$_2$ 以每实例平均求解时间替代组间变异性和尾部风险，同时保留原始目标和验证引导的重新选择。不含 σ_t 和不含w_t 则进一步分别单独考察组间变异性和尾部风险的作用。在每种基线中，所有变体使用相同的代数预算和搜索设置。表 IV 报告了平均求解时间。在所评估的 Hercules 消融变体中，完整框架始终实现了最低的求解时间。移除两项时间准则（不含 Time$_1$）导致整体增幅最大，表明它们对缩短搜索时间贡献显著。以平均求解时间替代两项鲁棒性准则后，效率高于不含 Time$_1$，但在所有场景下仍慢于完整版本。移除尾部风险导致的整体增幅大于移除组间变异性，说明在所评估的设置下，尾部风险准则的贡献更大。移除验证引导的重新选择也会增加求解时间，表明除时间准则外，该方法还能带来额外益处。在新请求下以及路径进度较早时，观察到的绝对时间节省更大，因为此时仍有更多节点需要优化。

I. 边界分析

我们考察 ERDR 的求解时间优势如何随路径进度和动态变化强度而变化。评估在 TSP500 上使用 GLS 中的 EoH 和 Hercules。每次比较都使用相同的残余实例和暖启动路径，并报告各方法的平均求解时间。在每个路径进度水平 $\rho \in \{20\%、50\%、80\%\}$，我们分别在每种动态场景中改变变化强度。对于位置变化，我们将 σ 调整为 $\{0.01, 0.02, 0.03, 0.04, 0.05\}$；对于新请求，将$r_1$ 调整为 $\{10\%、20\%、30\%、40\%、50\%\}$；并且

r_2 对于请求取消，也使用相同的比例，同时将所有其他实验设置保持为默认值。

图 4 显示，ERDR 在大多数测试的变化强度和进度水平下都保持求解时间优势，这表明其搜索效率不仅适用于默认场景。(a1) 和 (a2) 中 EoH 的求解时间停滞在约 10 秒，反映出搜索时间预算已耗尽；由于需要完成正在进行的迭代，实际耗时略有超出。在坐标扰动强度或节点添加比例较高时，运行时间优势可能消失，甚至反转，如 (b1)、(c1) 和 (c2) 所示。较大的坐标偏移或节点插入可能改变候选边的顺序，并引发连锁的巡回路径调整。如果 ERDR 扰动的连接更多，这些连锁反应造成的局部搜索成本可能高于更保守的基线扰动，从而逆转其运行时间优势。在大量取消和巡回路径后期进度的情况下，较少的剩余任务会降低搜索工作量和可供改进的绝对空间，因此在若干设置下，运行时间较短，差异也较小。

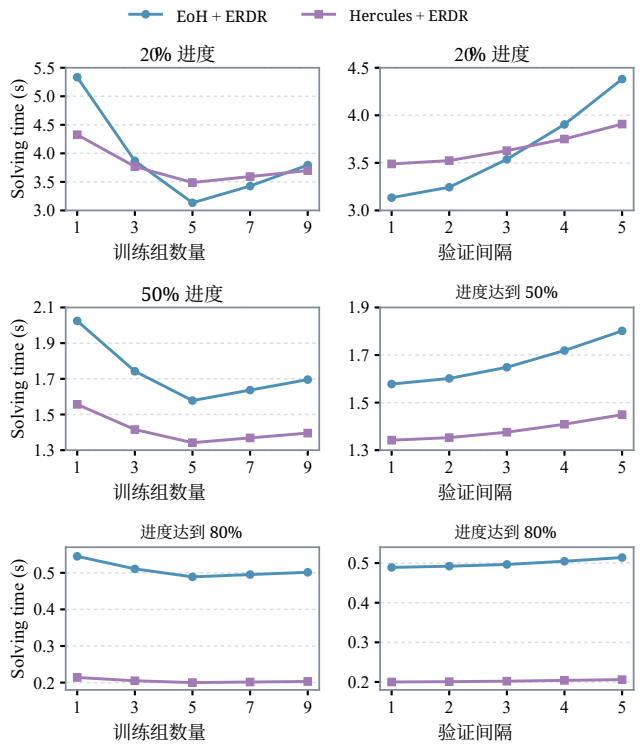

图5。在 GLS 中，对于以 20% 的节点添加比例加入新请求的 TSP500，求解时间对训练组数量（左）和验证间隔（右）的敏感性。验证间隔表示两次连续验证引导的重新选择之间的种群更新次数。各行分别对应 20%、50% 和 80% 的路径进度。各面板采用独立的纵轴刻度。蓝色和紫色分别表示 EoH+ERDR 和 Hercules+ERDR。数值越低越好。

J. 参数分析

我们通过改变训练组数量以及连续两次验证引导的重新选择之间的种群更新次数，考察 ERDR 对训练与验证分配比例及验证间隔的敏感性。我们改变每次只调整一个参数，在 GLS 的 TSP500 上以 20% 的节点添加比例加入新请求，并在路径进度达到 20%、50% 和 80% 时评估 EoH+ERDR 和 Hercules+ERDR。

图5显示，随着训练组数量增加，求解时间先减少后增加。增加训练分配会扩大常规候选评估的实例覆盖范围，同时减少用于验证引导的重新选择的实例数量。在三个进度水平下，训练与验证之间进行均等分配时，两种方法的求解时间均最低。每次种群更新进行一次验证与每两次更新进行一次验证所得到的求解时间相近。更长的间隔会增加求解时间，其中在 20% 路径进度下的绝对增幅最大，表明在这一设置中，频繁的验证引导的重新选择有利于提高重新求解效率。

表V 在20%路径进度时，各动态场景下每个基线–ERDR配对中首次匹配或改进较高最终目标值所需的搜索时间（秒）和迭代次数。数据为配对成功实例的均值。每个基线对中较低的数值以粗体显示。

方法	位置变更		新请求		取消	
	时间（秒）↓	迭代次数↓	时间（秒）↓	迭代次数↓	时间（秒）↓	迭代次数↓
GLS – TSP200						
EoH	0.3592	40.42	0.2818	37.00	0.0968	18.14
+ERDR	**0.0158**	**23.44**	**0.0144**	**21.84**	**0.0132**	**5.30**
ReEvo	0.1820	242.68	0.2119	196.84	0.0903	195.00
+ERDR	**0.0780**	**203.38**	**0.0973**	**150.50**	**0.0420**	**167.98**
GLS – TSP500						
EoH	0.5057	93.38	2.1124	109.64	0.5624	88.10
+ERDR	**0.1680**	**16.58**	**0.6982**	**93.04**	**0.2834**	**50.02**
ReEvo	0.6162	397.80	0.7055	249.76	0.3011	235.94
+ERDR	**0.5418**	**312.04**	**0.4763**	**230.88**	**0.1954**	**220.40**
ILS – TSP200						
EoH	0.5321	223.46	1.0484	339.80	0.6390	270.16
+ERDR	**0.2085**	**87.98**	**0.3394**	**125.58**	**0.1567**	**131.70**
ReEvo	4.8729	374.52	5.2356	305.42	2.1675	195.32
+ERDR	**0.4074**	**229.24**	**0.4167**	**174.08**	**0.1668**	**134.96**
ILS – TSP500						
EoH	1.1277	201.58	2.4566	285.80	1.2149	198.16
+ERDR	**0.7821**	**87.98**	**0.5478**	**144.32**	**0.5897**	**118.70**
ReEvo	1.4465	415.38	1.6231	341.48	0.6422	337.14
+ERDR	**1.3790**	**386.26**	**1.6108**	**284.30**	**0.6224**	**271.36**

K. 搜索效率分析

我们分析搜索轨迹，以考察 ERDR 是否确实加快了搜索过程。对于每个实例，我们比较各基线及其 ERDR变体在搜索预算内首次达到或优于两者最终目标值中较高者所需的时间和迭代次数。按照默认实验设置，我们评估了 EoH 和 ReEvo 在 TSP200 和 TSP500 上、巡回路线进度为 20% 时采用 GLS 和 ILS 的表现。表V报告了各项评估配置在这两项指标上的降幅。结果表明，ERDR 能以更短时间和更少搜索迭代达到配对的共同目标。对于动态事件，新请求通常比取消请求带来更大的绝对时间节省。这可能是因为新请求引入了额外的插入和排序决策，

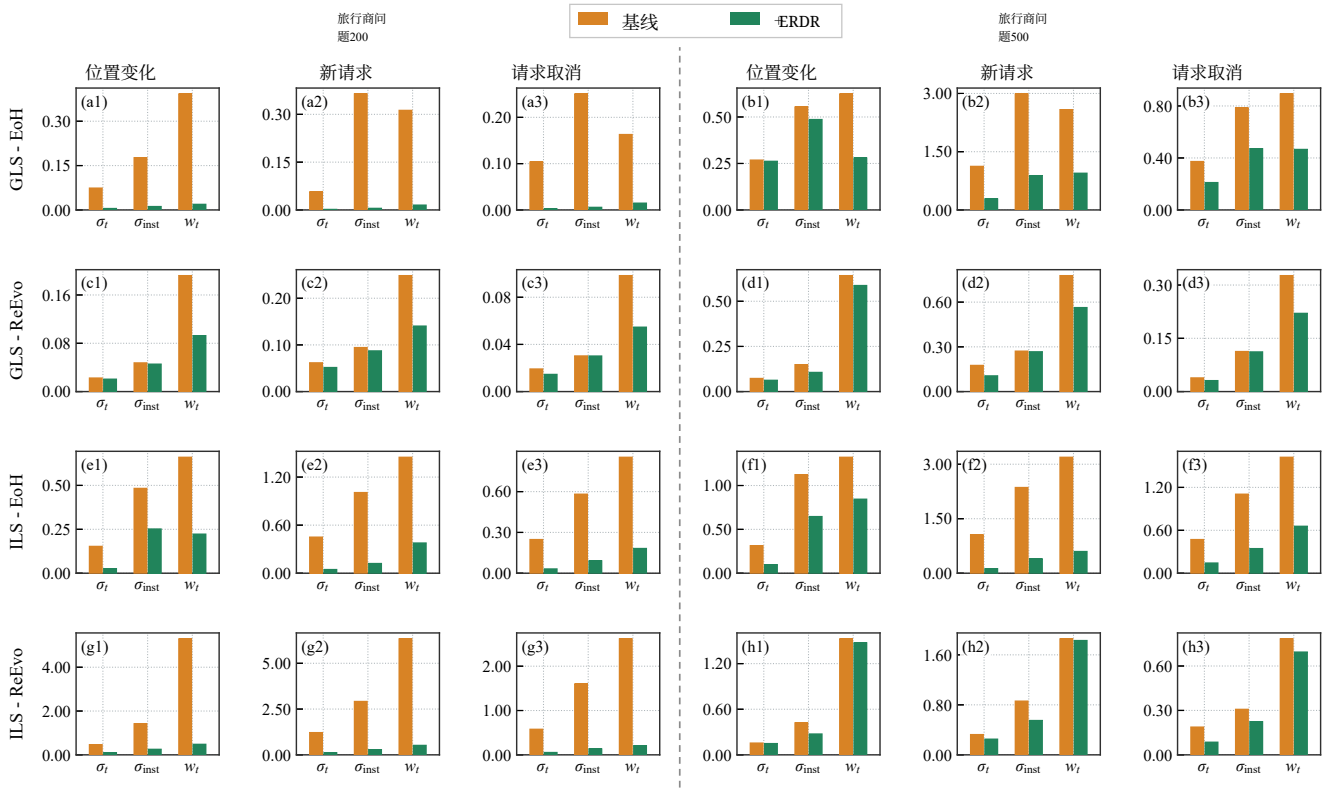

图6. 在巡回路线进度达到20%后发生动态变化时，到达配对共同目标所需时间的波动性和尾部成本。各行分别展示搜索框架和基线。左、右两组分别报告TSP200和TSP500，每组的列依次对应位置变化、新请求和请求取消。柱形显示基线（橙色）及其ERDR变体（绿色）的 σ_t、 σ_{inst}和 w_t 。所有数值均以秒为单位。

这些决策为 ERDR 留下了更多可削减的高成本探索，而取消请求会缩小剩余任务规模，从而限制可节省的时间。不同方法的搜索进度也有所差异。EoH 通常表现出更大的相对迭代次数降幅，这表明其进化启发式算法可能包含更多可由运行时间感知选择消除的冗余探索。在大多数设置下，ReEvo 的迭代次数仍然较高，这可能反映出其改进分布在更多搜索步骤中，因此缩短搜索轨迹的空间较小。

L. 运行时间稳定性与尾部性能

我们进一步考察：ERDR 所实现的加速是否伴随着动态重新求解过程中跨实例搜索时间变异性的降低，以及较慢实例组成本的下降。按照默认实验设置，我们在 TSP200 和 TSP500 的 20% 路径进度下，比较 GLS 和 ILS 中的 EoH、ReEvo 及其 ERDR 变体。利用每个求解器首次匹配或改进第 VI-K 节定义的配对共同目标所需的时间，我们计算组间标准差 σ_t 和尾部均值 w_t，分别用于表征运行时间变异性和慢组成本。我们还报告实例级标准差 σ_{inst}。结果基于每种配置的一次评估运行。

图6显示，新请求通常会带来更大的组间变异性和尾部成本，这与额外节点增加搜索负担的情况一致。在此场景下，ERDR降低了这两项指标，缓解了因剩余任务扩展而导致的运行时间不均和较慢组的高成本问题。请求取消的尾部成本通常低于新请求场景中的尾部成本，ERDR还进一步缩短了较慢组的运行时间。在位置变化场景下，节点数保持不变，两项指标也均有所改善，使空间变化后的重新求解同样受益。这些结果表明，ERDR能够缓解不同动态场景下实例组之间运行时间不均以及较慢组成本偏高的问题，从而提升动态重新求解的效率。

表VI 在 GLS 和 ILS 下 TSP200 和 TSP500 上三次运行的平均端到端演化时间（小时）。每个规模比较基线与基线+ERDR。

		EoH		Hercules		ReEvo		MEoH	
规模	变体	GLS	ILS	GLS	ILS	GLS	ILS	GLS	ILS
TSP200	基线	2.60	12.49	0.25	1.03	0.39	2.61	2.80	11.26
	+ERDR	2.40	11.09	0.23	1.24	0.33	2.01	2.62	11.35
TSP500	基线	4.30	22.06	0.55	7.34	0.84	7.76	4.65	19.82
	+ERDR	3.93	19.95	0.50	5.91	0.75	6.80	4.25	18.68

M. 演化开销

我们在上述匹配设置和生成预算下，使用每组比较中相同的50个源实例，测量TSP200和TSP500上的端到端进化时间。计时涵盖完整的进化运行过程，包括初始候选生成、大语言模型调用、源数据集

表 VII Hercules、ReEvo 及其 ERDR 变体在 GLS 中六个 TSPLIB 实例上的总搜索时间（秒）。启发式算法在 TSP200 实例上演化，并使用默认搜索设置，直接在从每个 TSPLIB 实例派生的动态任务上进行评估。在每个基线对中，时间较短的结果以粗体显示。

实例	方法	20%			50%			80%		
		位置变更	新请求	取消	位置变更	新请求	取消	位置变更	新请求	取消
st70 ($n = 70$)	Hercules	0.078	0.104	0.053	0.036	0.057	0.029	0.014	0.016	0.012
	+ERDR	**0.047**	**0.057**	**0.030**	**0.024**	**0.029**	**0.016**	**0.012**	**0.011**	**0.009**
	ReEvo	0.061	0.095	0.043	0.028	0.041	0.021	0.012	**0.013**	0.009
	+ERDR	**0.042**	**0.063**	**0.027**	**0.021**	**0.026**	**0.017**	**0.009**	**0.013**	**0.007**
pr76 ($n = 76$)	Hercules	0.062	0.075	0.045	0.033	0.042	0.026	0.014	0.015	0.013
	+ERDR	**0.044**	**0.057**	**0.032**	**0.023**	**0.029**	**0.017**	**0.011**	**0.011**	**0.008**
	ReEvo	0.070	0.063	0.042	0.038	0.048	**0.020**	0.011	0.011	0.009
	+ERDR	**0.044**	**0.057**	**0.032**	**0.023**	**0.030**	0.024	**0.009**	**0.010**	**0.008**
ch150 ($n = 150$)	Hercules	0.592	0.854	0.382	0.258	0.344	0.172	0.053	0.066	0.035
	+ERDR	**0.135**	**0.185**	**0.085**	**0.058**	**0.082**	**0.041**	**0.016**	**0.021**	**0.013**
	ReEvo	0.603	0.805	0.441	0.278	0.320	0.182	0.050	0.070	0.034
	+ERDR	**0.150**	**0.240**	**0.115**	**0.133**	**0.093**	**0.048**	**0.026**	**0.023**	**0.016**
a280 ($n = 280$)	Hercules	0.681	0.778	0.346	0.285	0.363	0.151	0.056	0.072	0.043
	+ERDR	**0.475**	**0.650**	**0.312**	**0.175**	**0.281**	**0.122**	**0.040**	**0.061**	**0.029**
	ReEvo	0.639	0.727	0.389	0.332	0.352	0.152	0.048	0.070	0.043
	+ERDR	**0.501**	**0.621**	**0.284**	**0.190**	**0.276**	**0.120**	**0.040**	**0.055**	**0.029**
rd400 ($n = 400$)	Hercules	1.157	1.748	0.702	0.420	0.715	0.301	0.087	0.109	0.066
	+ERDR	**1.030**	**1.499**	**0.635**	**0.387**	**0.557**	**0.245**	**0.075**	**0.107**	**0.049**
	ReEvo	1.393	2.150	0.887	0.579	0.689	0.351	0.123	0.150	0.075
	+ERDR	**1.102**	**1.584**	**0.687**	**0.422**	**0.611**	**0.282**	**0.079**	**0.104**	**0.060**
d493 ($n = 493$)	Hercules	5.095	6.995	3.141	1.885	2.688	1.092	0.322	0.484	0.190
	+ERDR	**1.797**	**2.073**	**1.012**	**0.767**	**0.873**	**0.343**	**0.112**	**0.175**	**0.074**
	ReEvo	2.845	3.071	2.973	1.619	1.817	0.415	0.218	0.265	0.104
	+ERDR	**1.680**	**2.036**	**0.964**	**0.649**	**0.785**	**0.397**	**0.114**	**0.152**	**0.070**

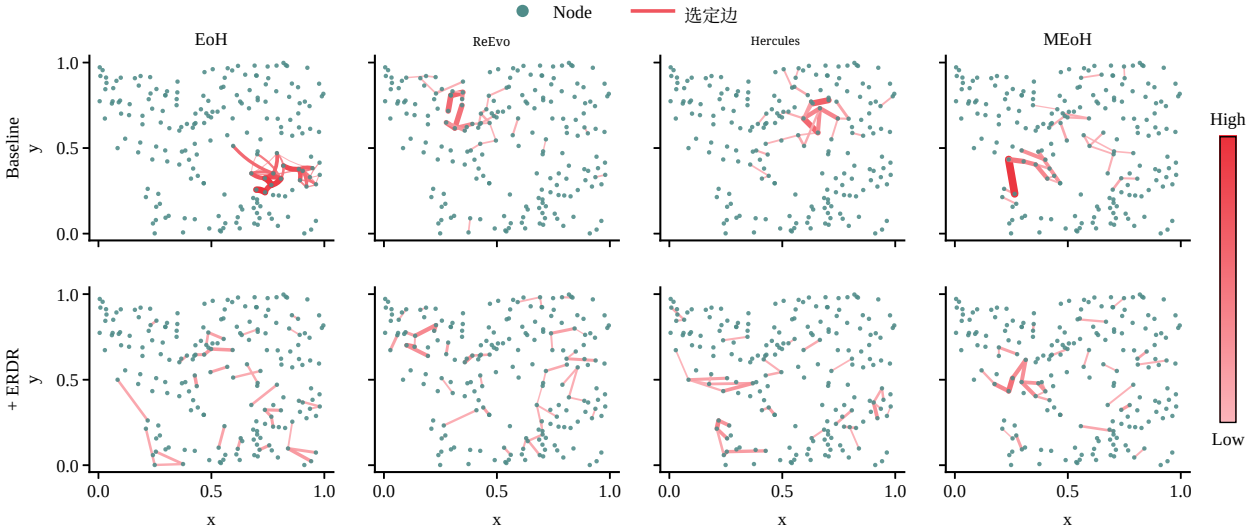

图 7. 在采用 20% 节点添加比例、并于路径进度达到 20% 时添加新请求的 TSP200 上，GLS 内惩罚边选择的覆盖情况。各列依次显示 EoH、ReEvo、Hercules 和 MEoH，上方为基线，下方为其 ERDR 变体。为提供共同的空间参照，节点坐标与 EoH 基线对齐，边端点映射至该布局，同时保留边颜色和宽度。

评估以及幸存者选择。表VI报告了GLS和ILS三次运行的均值。尽管ERDR引入了验证引导的重新选择和运行时稳健评估，但在GLS和ILS上，其端到端进化时间通常更低。这一减少得益于更小的常规评估工作负载。基线会在完整源数据集上评估新生成的候选解，而ERDR仅使用训练子集，并在预定的种群重新选择步骤中评估验证子集，从而使常规评估节省的时间抵消额外的验证成本。端到端的节省还取决于中间候选解的评估成本。API 延迟、重试和评估超时也会进一步影响这种平衡。例如，尽管最终启发式算法的搜索速度更快，但 Hercules 和 MEoH 在 TSP200 上采用 ILS 时，进化所需时间更长。

N. 对 *TSPLIB* 实例的泛化

我们进一步在源自 TSPLIB 的动态任务上评估 ERDR，其中包括来自实际应用的实例。我们在 TSP200 实例上，分别使用 Hercules、ReEvo 及其 ERDR

EoH

```
q_low =np.quantile(edge_n_used, 0.33)
q_高 = np.quantile(edge_n_used, 0.66)
#  <-- 来自稀疏 N×N 矩阵的阈值
for i in range(len(local_opt_巡回路线)):
    u = local_opt_巡回路线[i]
    v = local_opt_巡回路线[(i + 1) %
                          len(local_opt_巡回路线)]
    用法_计数 =边_n_已使用[u、v]
    if usage_count < q_低:
        已更新_边_距离[u, v]*= 0.75
    elif usage_count >q_高:
        更新_边_距离[u, v]*= 1.10
    更新_边_距离[u, v] *= 0.97
    #  <-- 全局校准后的固定乘数
```

EoH+ERDR

```
更新_边_距离 = 边_距离.copy()
总_边数 = len(local_opt_巡回路线)
平均_距离 =总_距离 / 总_边数
#  <-- 当前巡回的规模

对于 i in range(total_edges):
    u = local_opt_巡回路线[i]
    v = local_opt_巡回路线[(i + 1) % total_edges]
    如果边_n_used[u, v] > 0:
        调整量 = (
            1 + 0.2 * np.random.uniform(-1, 1)
        ) * (
            平均_距离
            / 最大值(
                更新后的_边_距离[u, v],
                1e-12
            )
        ) ** 0.5
        #  <-- 相对边到巡回距离校准
        已更新_边_距离[u、v]*= 调整量
```

图 8. 最终 EoH（橙色）和 EoH+ERDR（绿色）求解器中的边距离更新逻辑。带颜色的箭头注释突出了使用阈值以及边距离到巡回路线距离的校准。图中仅展示比较所需的语句，并缩短了局部变量名称。

变体演化启发式算法，然后直接在源自 TSPLIB 的动态任务上对其进行评估。对于 TSPLIB，表 I 中的动态变化采用各原始实例的坐标尺度和边界。表 VII 报告了 Hercules 和 ReEvo 在 GLS 中应用于六个选定实例时的结果。在 Hercules 和 ReEvo 上，ERDR 的平均加速比达到 $1.89\times$。路径进度较早时，收益通常更大；而剩余任务规模较小时，加速空间则较有限。不同测试实例的收益也有所不同，并不会随着实例规模增大而单调增加，这表明空间结构及由此产生的搜索行为可能会影响运行时感知启发式选择的效果。这些结果表明，ERDR 能够有效扩展到实际应用数据。

O. 案例研究

a) 空间选择模式： 我们以一个具有代表性的 TSP200 实例为例，说明 GLS 中基线及其 ERDR 增强变体在惩罚边选择方面的差异。在路径进度达到 20% 时，我们新增数量等于未访问节点 20% 的请求，并比较四种基线及其 ERDR 增强变体后续的重新求解行为。我们记录每种演化启发式算法选作惩罚对象的边，并汇总其选择频率，涵盖整个搜索过程。图 7 在统一的坐标布局中呈现了这些模式：节点与 EoH 基线对齐，边端点相应映射，同时保留原始线型。基线启发式算法会反复选择位于有限区域内的惩罚边，但不同骨干网络的集中程度有所差异。在全部四组比较中，ERDR 增强启发式算法覆盖了更广的区域，降低了局部高频簇的主导地位，同时保留了特定于骨干网络的选择模式。这些观察结果表明，在该实例上进行重新求解时，惩罚边选择的空间集中程度较低。

b) 演化启发式规则： 我们选取 EoH 和 EoH+ERDR 在 TSP500 上演化得到的具有代表性的最终启发式算法，作为考察其边距离更新规则的示例。图 8 展示了嵌入这些最终求解器中的边距离更新逻辑：在下一步搜索之前，利用当前局部最优巡回路线和累积的边使用次数修订边成本。EoH 根据完整的稀疏 $N \times N$ 使用矩阵计算 q_低 和 q_高，然后应用 0.75、1.10 和 0.97 这几个乘数。零值条目占比足够大时，这两个阈值都会变为零，因此所有计数为正的边都会获得相同的组合乘数$1.10 \times 0.97 = 1.067$，而不受其计数大小的影响。EoH+ERDR 同样会选择计数为正的边，然后根据平均_距离与当前边距离之比校准的随机因子，将这些边的成本相乘。因此，调整量会随每条边相对于当前巡回路线的距离而变化。这类距离相关更新可能有助于在实例变化后调整搜索引导，从而为观察到的时间缩短现象提供一种可能的解释。

P. 更多细节与分析

由于篇幅限制，更多实验和实现细节请参阅补充材料，该材料与代码和数据一同发布于 https://github.com/myz12138/ERDR。

VII. 结论

本文研究了当实例在路径执行过程中发生变化时，AHD 进化的 TSP 求解器进行低效重新求解的问题。我们的分析表明，“静态过拟合”是造成这一低效问题的原因之一：发生此类变化后，进化求解器会反复将搜索引向彼此重叠的局部邻域。我们进一步将这一行为与演化过程联系起来：这些过程会保留这样一些启发式算法，其质量提升集中在少数几个实例上，且伴随着更长的求解时间。我们提出了 ERDR，这是一种即插即用框架，将验证引导的重新选择与运行时鲁棒性增强评估相结合。在四种 AHD 方法、两个搜索框架（GLS 和 ILS）、多种 TSP 规模及三种变化类型上的实验表明，ERDR 能够缩短路径重新求解时间。

人工智能生成内容声明

第五节和第六节中的求解器演化过程使用 gpt-6-luna API 生成并修订候选启发式算法代码。ChatGPT 和 Codex 协助起草并修订了全文，包括提升语言清晰度、调整段落结构以及统一术语。它们还协助完成全文的 LaTeX 排版、图表布局，以及第六节实验所用代码的编辑与调试。

参考文献

[1] G. Dantzig、R. Fulkerson 和 S. Johnson，《大规模旅行商问题的求解》，《美国运筹学会志》，第 2 卷第 4 期，393–410 页，1954 年。[2] G. Laporte，《旅行商问题：精确算法与近似算法综述》，《欧洲运筹学杂志》，第 59 卷第 2 期，231–247 页，1992 年。[3] R. M. Karp，《组合问题之间的可归约性》，载于《整数规划 50 年（1958–2008）：从早期发展到最新进展》，2009 年，219–241 页。[4] F. Hutter、H. H. Hoos、K. Leyton-Brown 和 T. St¨utzle，《ParamILS：一种自动算法配置框架》，《人工智能研究杂志》，第 36 卷，267–306 页，2009 年。[5] A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、Ł. Kaiser 和 I. Polosukhin，《注意力就是你所需要的一切》，载于 NeurIPS，2017 年。[6] T. Brown、B. Mann、N. Ryder、M. Subbiah、J. D. Kaplan、P. Dhariwal、A. Neelakantan、P. Shyam、G. Sastry、A. Askell 等，《语言模型是少样本学习器》，NeurIPS，1877–1901 页，2020 年。[7] L. Ouyang、J. Wu、X. Jiang、D. Almeida、C. L. Wainwright、P. Mishkin、C. Zhang、S. Agarwal、K. Slama、A. Ray 等，《训练能够遵循指令并接受人类反馈的语言模型》，载于 NeurIPS，2022 年，27 730–27 744 页。[8] M. Chen、J. Tworek、H. Jun、Q. Yuan、H. P. D. O. Pinto、J. Kaplan、H. Edwards、Y. Burda、N. Joseph、G. Brockman 等，《评估基于代码训练的大语言模型》，arXiv 预印本 arXiv:2107 03374，2021 年。[9] Y. Li、D. Choi、J. Chung、N. Kushman、J. Schrittwieser、R. Leblond、T. Eccles、J. Keeling、F. Gimeno、A. Dal Lago 等，《达到竞赛水平的代码生成：AlphaCode》，《科学》，第 378 卷第 6624 期，1092–1097 页，2022 年。[10] F. Liu、X. Tong、M. Yuan、X. Lin、F. Luo、Z. Wang、Z. Lu 和 Q. Zhang，《启发式演化：利用大语言模型实现高效自动算法设计》，载于 ICML，第 235 卷，2024 年，32 201–32 223 页。[11] H. Ye、J. Wang、Z. Cao、F. Berto、C. Hua、H. Kim、J. Park 和 G. Song，《ReEvo：作为具有反思演化能力的超启发式的大语言模型》，载于 NeurIPS，2024 年。[12] M. Horv´ath 和 T. Tam´asi，《动态车辆路径规划的通用建模与仿真框架》，《EURO交通与物流杂志》，第 14 卷，文章编号 100159，2025 年。[13] V. Pillac、M. Gendreau、C. Gu´eret 和 A. L. Medaglia，《动态车辆路径问题综述》，《欧洲运筹学杂志》，第 225 卷第 1 期，1–11 页，2013 年。[14] M. Gendreau、F. Guertin、J. Potvin 和 ´E. D. Taillard，《用于实时车辆路径规划与调度的并行禁忌搜索》，《交通科学》，第 33 卷第 4 期，381–390 页，1999 年。[15] V. Pillac、C. Gu´eret 和 A. L. Medaglia，《动态技术员路径与调度问题的快速重新优化方法》，载于《元启发式的最新发展》，2018 年，第 62 卷，347–367 页。[16] C. Voudouris 和 E. P. K. Tsang，《引导式局部搜索及其在旅行商问题中的应用》，《欧洲运筹学杂志》，第 113 卷第 2 期，469–499 页，1999 年。[17] K. Helsgaun，《Lin–Kernighan 旅行商启发式算法的有效实现》，《欧洲运筹学杂志》，第 126 卷第 1 期，106–130 页，2000 年。[18] Y. Ma 、 R. Wang 、 S. Liang、G. Luo 和 K. Qin，《TopoLLM：由大语言模型驱动的自适应工具学习，用于实时应急网络拓扑规划》，《数字通信与网络》，第 12 卷第 2 期，273–282 页，2026 年。

[19] M. Li、H. Gou、D. Zhang、S. Liang、X. Xie、D. Ouyang 和 K. Qin，《超越随机：数据集蒸馏中的自动内循环优化》，载于 NeurIPS，2025 年。[20] E. K. Burke、M. Gendreau、M. Hyde、G. Kendall、G. Ochoa、E. ¨Ozcan 和 R. Qu，《超启发式：现状综述》，《运筹学会志》，第 64 卷，2013 年，1724，12，1695 页，第 期。[21] B. Romera-Paredes、M. Barekatain、A. Novikov、M. Balog、M. P. Kumar、E. Dupont、F. J. R. Ruiz、J. S. Ellenberg、P. Wang、O. Fawzi、P. Kohli 和 A. Fawzi，《利用大语言模型通过程序搜索进行数学发现》，《自然》，第 625 卷，468–475 页，2024 年。[22] R. Wang、S. Liang、Q. Chen、J. Zhang 和 K. Qin，《GraphTool-Instruction：通过分解式子任务指令革新大语言模型中的图推理》，载于 SIGKDD，2025 年，1492–1503 页。[23] J. Li、K. Qin、R. Wang、Y. Ma、Q. Chen、M. Li 和 S. Liang，《当进一步推理无益时停止：推理模型中的注意力状态自适应生成》，载于 ICML，第 306 卷，2026 年，69 468–69 489 页。[24] R. Wang、Y. Huang、M. Li、J. Li、D. Liang、B. Simons、P. Ke、S. Liang 和 K. Qin，《重新思考大语言模型驱动的启发式设计：通过动态感知优化生成高效且专用的求解器》，CoRR，第 abs/2601.20868 卷，2026 年。[25] X. Wu、D. Wang、C. Wu、L. Wen、C. Miao、Y. Xiao 和 Y. Zhou，《利用大语言模型生成用于求解组合优化问题的高效启发式算法》，载于 SIGKDD，2025 年，3228–3239 页。[26] S. Yao、F. Liu、X. Lin、Z. Lu、Z. Wang 和 Q. Zhang，《利用大语言模型进行启发式算法的多目标进化》，载于 AAAI，2025 年，27 144–27 152 页。[27] Z. Zheng、Z. Xie、Z. Wang 和 B. Hooi，《用于基于大语言模型的自动启发式设计中全面探索的蒙特卡洛树搜索》，载于 ICML，2025 年，78 338–78 373 页。[28] Z. Huang、W. Wu、K. Wu、W.-B. Lee 和 J. Wang，《CALM：用于自动启发式设计的算法与语言模型协同进化》，载于 ICLR，2026 年。[29] C. Chen、M. Zhong、J. Shi、J. Sun 和 Y. Fan，《HiFo-Prompt：利用后见信息与前瞻信息为基于大语言模型的自动启发式设计提供提示》，载于 ICLR，2026 年。[30] C. Archetti、L. Bertazzi 和 M. G. Speranza，《重新优化旅行商问题》，《网络》，第 42 卷，2003 年，159，3，154 页，第 期。[31] Y. Tong、Y. Zeng、Z. Zhou、L. Chen、J. Ye 和 K. Xu，《共享出行路径规划的统一方法》，《VLDB基金会会刊》，第 11 卷第 11 期，1633–1646 页，2018 年。[32] R. Tin´os、L. D. Whitley 和 A. E. Howe，《显式记忆在动态旅行商问题中的应用》，载于 GECCO，2014 年，999–1006 页。[33] M. Mavrovouniotis、F. M. M¨uller 和 S. Yang，《结合局部搜索的动态旅行商问题蚁群优化》，《IEEE控制论汇刊》，第 47 卷第 7 期，1743–1756 页，2016 年。[34] C. Wang、R. Zhu、Y. Jiang、W. Liu、S.-W. Jeon、L. Sun 和 H. Hang，《基于方案库并结合 2-opt 局部搜索的动态旅行商问题蚁群优化》，《CMES-工程与科学计算机建模》，第 135 卷第 2 期，2023 年。[35] G. C. Cawley 和 N. L. Talbot，《关于模型选择中的过拟合及其后续性能评估中的选择偏差》，《机器学习研究杂志》，第 11 卷，2079–2107 页，2010 年。[36] J. R. Rice，《算法选择问题》，载于《计算机进展》，爱思唯尔，1976 年，第 15 卷，65–118 页。[37] R. T. Rockafellar、S. Uryasev 等，《条件风险价值优化》，《风险杂志》，第 2 卷，21–42 页，2000 年。[38] K. Deb、A. Pratap、S. Agarwal 和 T. Meyarivan，《快速且精英化的多目标遗传算法：NSGA-II》，《IEEE进化计算汇刊》，第 6 卷第 2 期，182–197 页，2002 年。[39] M. T. Emmerich 和 A. H. Deutz，《多目标优化教程：基础与进化方法》，《自然计算》，第 17 卷第 3 期，585–609 页，2018 年。[40] G. Reinelt，《TSPLIB——旅行商问题库》，《ORSA计算杂志》，第 3 卷第 4 期，376–384 页，1991 年。[41] D. L. Applegate、R. E. Bixby、V. Chv´atal 和 W. J. Cook，《旅行商问题：一项计算研究》，载于《旅行商问题》，2011 年。[42] H. R. Lourenc¸o、O. C. Martin 和 T. St¨utzle，《迭代局部搜索：框架与应用》，载于《元启发式手册》，2010 年，363–397 页。