计算机科学
COMPUTER SCIENCE

面向工业动态取送货问题的分解多目标进化算法

蔡俊创, 朱庆灵, 林秋镇, 李坚强, 明仲

引用本文

蔡俊创, 朱庆灵, 林秋镇, 李坚强, 明仲. 面向工业动态取送货问题的分解多目标进化算法[J]. 计算机科学, 2025, 52(1): 331-344.

CAI Junchuang, ZHU Qingling, LIN Qiuzhen, LI Jianqiang, MING Zhong. Decomposition-based Multi-objective Evolutionary Algorithm for Industrial Dynamic Pickup andDelivery Problems [J]. Computer Science, 2025, 52(1): 331-344.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.231200132

面向工业动态取送货问题的分解多目标进化算法

蔡俊创　朱庆灵　林秋镇　李坚强　明　仲
深圳大学计算机与软件学院　广东 深圳 518060
(caijunchuang2020@email.szu.edu.cn)

摘　要　由于工业动态取送货问题具有垛口、时间窗、容量、后进先出装载等多种约束,现有的车辆路径算法大多只优化一个加权目标函数,在求解过程中难以保持解的多样性,所以容易陷入局部最优区域而停止收敛。针对上述问题,提出了一种融合高效局部搜索策略的分解多目标进化算法。首先,该算法将工业动态取送货问题建模成多目标优化问题,进一步将其分解为多个子问题并同时进行求解。然后,利用交叉操作增强解的多样性,再使用局部搜索加快收敛速度。因此,该算法在求解该多目标优化问题时能够更好地平衡解的多样性和收敛性。最后,从种群中选择一个最好的解来完成当前时段的取送货任务。基于64个华为公司实际测试问题的仿真结果表明,该算法在求解工业动态取送货问题上的性能表现最优;同时,在20个京东物流大规模配送问题上的实验也验证了该算法良好的泛化性。

关键词:动态取送货问题;分解方法;多目标进化算法;局部搜索;组合优化
中图分类号　TP301

Decomposition-based Multi-objective Evolutionary Algorithm for Industrial Dynamic Pickup and Delivery Problems

CAI Junchuang,ZHU Qingling,LIN Qiuzhen,LI Jianqiang and MING Zhong
College of Computer Science and Software Engineering,Shenzhen University,Shenzhen,Guangdong 518060,China

Abstract　Due to the constraints of industrial dynamic pickup and delivery problems(DPDPs),such as docks,time windows,capacity,and last-in-first-out loading,most of the existing vehicle routing algorithms only optimize a single weighted objective function,which is difficult to maintain the diversity of solutions,so it is easily get stuck in local optimal region and stop converging. To alleviate this issue,this paper introduces a decomposition-based multi-objective evolutionary algorithm with efficient local search for solving the above DPDPs. Firstly,our algorithm models the DPDP into a multi-objective optimization problem(MOP),which is further decomposed into multiple sub-problems and solves them simultaneously. Then,crossover operation is used to enhance the diversity of solutions,followed by using an efficient local search to speed up the convergence. By this way,our algorithm can better balance the diversity and convergence of solutions when solving this MOP. Finally,the best solution can be selected from the population to complete the pickup and delivery tasks. Simulation results on 64 test problems from practical scenario of Huawei company demonstrate that our algorithm outperforms other competitive algorithms for tackling DPDPs. Meanwhile,the algorithm is also tested on 20 large-scale delivery problems of JD Logistics to validate its generalization.

Keywords　Dynamic pickup and delivery problem,Decomposition method,Multi-objective evolutionary algorithm,Local search,Combinatorial optimization

1　引言

取送货问题(Pickup and Delivery Problem,PDP)是一类应用广泛的车辆路径问题(Vehicle Routing Problems,VRP)[1-2],也是一种NP-难组合优化问题[3],其目标是将物品或人以最小的代价从起点运输到终点[4]。动态取送货问题(Dynamic PDP,DPDP)是PDP的一种复杂变体,其进一步考虑了PDP中动态生成订单的现实特性,需要动态地将每个订单分派给车辆以最小代价进行服务。近年来,随着在现实生活中的广泛应用,DPDP受到越来越多的关注。通常,求解

到稿日期:2023-12-20　返修日期:2024-05-26
基金项目:国家自然科学基金(62272315,62376163,62203308);广东省区域联合基金(2022B1515120076);广东省自然科学基金(2023A1515011238);深圳市科技计划项目(JCYJ20220531101411027)
This work was supported by the National Natural Science Foundation of China(62272315,62376163,62203308),Guangdong Regional Joint Foundation Key Project(2022B1515120076),Natural Science Foundation of Guangdong Province,China(2023A1515011238) and Shenzhen Science and Technology Program(JCYJ20220531101411027).
通信作者:明仲(mingz@szu.edu.cn)

DPDP 的方法主要可以分为以下两类[5]。

第一种是精确式算法[6]，其将 DPDP 分解为一系列静态问题，再使用线性规划或混合整数规划对每个静态问题进行优化。Savelsbergh 等[6]先将 DPDP 分解为一系列静态优化问题，使得每个静态问题包含一段时间跨度上已知的配送订单子集。然后，基于一种分支定界算法求解静态问题，并通过使用近似技术、不完全优化技术以及复杂的列管理策略来平衡动态环境下的求解速度和求解质量。Swihart 和 Papastavrou[7]假设服务订单按照泊松过程及时到达，建立了 DPDP 的随机动态模型。在该模型中，订单的取、送货位置独立且均匀地分布在服务区域上。此外，作者给出了路径规划策略在单位容量车辆或多容量车辆中的性能上下限。Arslan 等[8]将众包交付场景中的 DPDP 分解为一系列静态优化问题，并提出了一种精确递归算法来实现静态问题中的包裹配送任务和驾驶员的实时匹配。然而，上述方法需要重新优化每个静态子问题，因此存在计算复杂度较高的问题，并不能满足工业场景中快速服务动态订单的需求。

另一种是启发式算法，其通过将新的配送订单动态地插入到已规划好的路线中，来降低计算复杂度。这类方法在现有 DPDP 研究中被广泛使用[3]。针对信件和小包裹运输的当日取送货问题，Mitrovi-Mini 等[9]将动态问题转化为多个跨度滚动的静态问题。每个静态问题通过两阶段启发式算法进行优化：第一阶段使用禁忌搜索算法，而第二阶段使用插入启发式算法。Gendreau 等[10]设计了一种禁忌搜索算法，利用基于弹射链的邻域结构来探索新的候选解。Ma 等[11]提出了一个分层优化框架来求解 DPDP。该框架由上层代理和下层代理组成：上层代理动态地将 DPDP 划分为一系列不同规模的子问题，而下层代理使用包含 4 种局部搜索策略的元启发式算法来提升子问题的解的质量。针对共享物流平台中的 DPDP，Su 等[12]提出了一种结合遗传算法、变邻域搜索算法和禁忌搜索算法的混合并行搜索算法；Xu 和 Wei[13]提出一种自适应大邻域搜索启发式算法来求解 DPDP，并使用 Q-learning 调整算子权重以提高求解效率；Tao 等[14]使用自适应可变邻域启发式算法来求解最后一英里配送问题；Ulmer 等[15]通过使用订单延迟分配策略，更灵活地求解餐馆外卖配送问题；Du 等[16]使用 3 种直观的策略（紧急订单优先配送策略、搭便车策略、拼单策略）来求解 DPDP；Li 等[17]通过建立基于图的价值函数来对车辆间的关系进行建模，并使用基于注意力的图嵌入与深度 Q-learning 对 DPDP 的配送需求进行预测；Ghiani 等[18]提出了一种参数策略函数近似方法，用于预测 DPDP 的订单特征；Cai 等[19]提出了一种变邻域搜索算法求解带有垛口、时间窗、容量和后进先出（Last-in-First-out，LIFO）装载等约束的实际 DPDP，该算法融合了 4 种局部搜索策略和一种扰动算子，在国际自动规划与调度会议（ICAPS 2021）的 DPDP 竞赛中取得了第一名的成绩。其他启发式方法也被用于求解 DPDP，如等待和缓冲策略[20-21]、订单预测算法[22-23]、变邻域搜索算法[24-25]。更多的 DPDP 研究现状可参考最新的综述[5]。

上述方法在求解一些特定 DPDP 应用中表现出优异的性能，但在处理复杂场景的 DPDP 时性能不佳。例如，华为根据实际业务场景在 ICAPS 2021 竞赛中提出的工业 DPDP[26]，由于垛口、时间窗、容量和 LIFO 装载等约束，其性能显著下降。尽管 ICAPS 2021 竞赛中排名前三的算法在求解该工业 DPDP 时取得了很好的效果，但它们都只优化了一个基于订单总超时和车辆平均行驶距离的加权目标，在其搜索过程中难以保持解的多样性，容易陷入局部最优解。针对这一问题，本文提出了一种融合高效局部搜索策略的多目标分解进化算法（Decomposition-based Multi-objective Evolutionary Algorithm with Efficient Local Search Strategies，MOEA/D-ES）来求解工业 DPDP。MOEA/D-ES 首先将 DPDP 建模为多目标优化问题（Multi-objective Optimization Problem，MOP），然后结合基于分解的多目标优化算法和局部搜索方法高效求解上述 MOP。具体来说，MOEA/D-ES 使用不同的权重向量将 MOP 进一步分解为 N 个子问题，然后从存档中恢复上一时间段的解 x_{restore}；同时为当前时间段的所有新订单生成 N 个随机序列，并根据每一序列将所有新订单使用最便宜插入（Cheapest Insert，CI）方法插入到 x_{restore} 上生成初始种群。在每次迭代中，随机选择两个父代个体进行交叉操作生成子代个体，并使用 4 种有效的局部搜索策略对子代个体进行迭代进化。在 DPDP 中的每个执行时间段内，动态地从种群中选择一个最好的解来完成当前存在的订单。与 ICAPS 2021 DPDP 竞赛中排名前三的算法[1)]以及基准算法相比，本文提出的算法在求解该问题时能够有效平衡解集的收敛性与多样性，从而进一步提升了最终解的质量。

综上所述，本文的主要贡献如下：

1）设计了一种新颖的交叉算子，有助于生成多样化的子代解，避免陷入局部最优区域。首先，每个子代从两个父代中随机继承每条路径。然后，删除重复节点以确保可行性，并将未分配的订单插入到已规划的路径中，从而形成完整的可行解。

2）设计了一种局部搜索方法对上述的子代个体进行进一步优化。该局部搜索方法结合了对交换、块交换、对重定位和块重定位 4 种有效的局部搜索策略。通过这种方式，进一步提升了子代个体的质量。

3）将上述交叉算子与局部搜索相结合，提出了一种基于分解的多目标进化算法（MOEA/D-ES）来求解工业 DPDP。MOEA/D-ES 先将 DPDP 建模得到的 MOP 分解为多个子问题，再通过交叉算子和局部搜索方法提升解集的质量。实验结果表明，在求解华为工业 DPDP 时，MOEA/D-ES 的性能表现最佳。此外，MOEA/D-ES 在求解京东配送问题时也取得了最佳结果，有效验证了其泛化能力。

本文第 2 章介绍了一些背景信息和设计 MOEA/D-ES 的动机；第 3 章介绍了所提出的 MOEA/D-ES；第 4 章给出了实验结果并进行了分析；最后给出了结论并展望了未来的研究方向。

1) https://competition.huaweicloud.com/information/1000041411/Winning

2 背景和研究动机

为了阐明本文的背景，第 2.1 节描述了工业 DPDP 的数学模型，第 2.2 节介绍了一种最新的分解多目标进化算法（MOEA/D[27]），第 2.3 节阐述了设计 MOEA/D-ES 的动机。

2.1 工业 DPDP 的数学模型

在工业 DPDP 中，不同的工厂在一天内会动态产生订单，需要车辆以最低的运输成本从起点将货物运输到终点。在该 DPDP 中，使用模拟器模拟物流场景。该模拟器将一天划分为若干固定的时间段，以使新到来的订单能在最近的时间段内动态完成。如图 1 所示，每个时间段的路径规划结果并不是相互独立的，当前的规划结果将影响车辆剩余容量以及与后续订单的相对位置。此外，在每个时间段的规划决策之后，一些现有订单将动态完成。因此，这是一个序列决策过程。

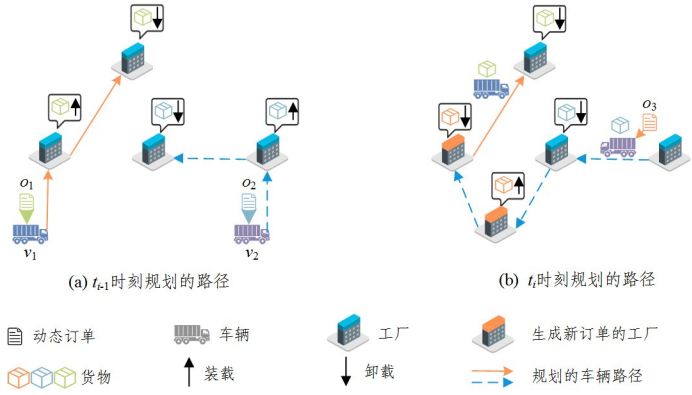


图 1 工业 DPDP 的简单例子

Fig. 1 Simple example of industrial DPDP

本文研究的工业场景 DPDP 的输入包括：M 个节点（工厂）$\{F_i \mid i=1,\cdots,M\}$，$F$ 是具有 M 个节点的集合，每个节点都有 L_d 个可供货车装、卸货物的垛口，如果一个工厂的所有垛口都被占用了，后续到达的车辆则需排队等待；完备有向图路线网络 $G=(F,A)$，其中边集合 $A=\{\langle i,j\rangle \mid i,j\in M\}$，每条边 $\langle i,j\rangle$ 都有非负实际距离 D_{ij}；一个大小为 C 的订单集合 $O=\{o_i \mid i=1,\cdots,C\}$，每个订单 $o_i=(F_p^i,F_d^i,Q^i,t_e^i,t_l^i)$ 可由不同工厂在一天内任意时刻动态生成，其中 F_p^i 和 F_d^i 分别表示取货节点和送货节点，Q^i 是订单的货物数量，t_e^i 指订单创建时间，t_l^i 是承诺完成订单的时间；同种类型车队 $V=\{v_k \mid k=1,\cdots,K\}$，$K$ 是车辆数量，每辆车 v_k 随机选择一个工厂作为初始位置，且具有装载容量限制；T_{da} 是车辆泊靠垛口的时间，即车辆从工厂直接泊靠到工厂垛口所需的时间。

该工业 DPDP 输出的是每一时间段的路线规划 $RP=\{rp_k=\{n_1^k,n_2^k,\cdots,n_{l_k}^k\} \mid k=1,\cdots,K\}$，其中 n_i^k 和 l_k 分别是车辆 v_k 应访问的第 i 个工厂和工厂数量。该 DPDP 考虑两个优化目标。第一个目标是最小化订单的总超时，即 f_1：

$$f_1=\sum_{k=1}^{K}\sum_{i=1}^{l_k}\max(0,t_i^k-t_i^{y_i^k})d_i^k,\ \forall i\in l_k,\forall k\in V \tag{1}$$

其中，t_i^k 是车辆 v_k 到达第 i 个节点的时间，y_i^k 是车辆 v_k 访问的第 i 个节点对应的订单，$t_i^{y_i^k}$ 是订单 y_i^k 承诺完成的时间。二进制变量 d_i^k 表示车辆 v_k 到达的第 i 个节点的类型，1 表示送货节点，0 表示取货节点。只有当车辆 v_k 到达配送节点时，才需要计算订单的总超时。此外，车辆 v_k 到达第 $(i+1)$ 个节点的时间 t_{i+1}^k 可以使用以下等式确定：

$$t_{i+1}^k=t_i^k+w_i^k+T_{da}+t_s^{y_i^k}+t_{i,i+1}^k,\ \forall i,i+1\in l_k,\forall k\in V \tag{2}$$

其中，w_i^k 是车辆 v_k 到达节点 i 后排队等待分配垛口的时间，T_{da} 是车辆 v_k 从工厂到工厂垛口的泊靠时间，$t_s^{y_i^k}$ 是车辆 v_k 在第 i 个节点装载或卸载单 y_i^k 货物的时间，$t_{i,i+1}^k$ 是车辆 v_k 从第 i 个节点行驶到第 $(i+1)$ 个节点的时间。

第二个目标 f_2 是最小化车辆的平均行驶距离。如果车辆 v_k 的路线规划是 $rp_k=\{n_1^k,n_2^k,\cdots,n_{l_k}^k\}$，其中 n_i^k 表示车辆 v_k 路线中的第 i 个工厂，l_k 是车辆 v_k 需要访问的总节点数，则 f_2 可由式(3)计算：

$$f_2=\frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{l_k-1}D_{n_i^k,n_{i+1}^k} \tag{3}$$

其中，$D_{n_i^k,n_{i+1}^k}$ 是从节点 n_i^k 到节点 n_{i+1}^k 的距离。以上两个目标用于指导群体搜索，进而增强解的多样性。

在每个时间段内，需要给出一个路径规划解以尽可能地完成现有订单。在这种情况下，式(4)定义了总目标来选择解，其中 α 是一个正常数，表示每单位订单延迟的惩罚，强调了在实际场景中订单的及时配送对提高客户满意度的重要性。式(5)和式(6)定义了车辆容量约束。每个订单交付到目的地的时间窗由式(7)表示，其中时间窗的右侧不等式是一个软约束。当订单交付到目的地超过承诺完成时间时，会产生惩罚值（即 f_1）。式(8)确保车辆先访问取货节点再访问对应送货节点。式(9)保证在一个工厂内同时占用垛口的车辆数不超过该工厂的总垛口数。式(10)对 LIFO 装载策略进行建模。式(11)确保所有订单都指派给车辆进行服务。

$$TC=\alpha\times f_1+f_2 \tag{4}$$

$$\text{s.t. } q_{i+1}^k-q_i^k-Q^{y_{i+1}^k}=0,\ \forall i,i+1\in l_k,\forall k\in V \tag{5}$$

$$0\leqslant q_i^k\leqslant Q,\ \forall i\in l_k,\forall k\in V \tag{6}$$

$$t_e^i\leqslant a_d^i\leqslant t_l^i,\ \forall i\in O \tag{7}$$

$$0\leqslant t_{F_p^i}^k\leqslant t_{F_d^i}^k,\ \forall i\in O,\forall k\in V,\forall F_p^i,F_d^i\in F \tag{8}$$

$$0\leqslant\sum_{k=1}^{K}z_i^k\leqslant L_d,\ \forall i\in F,\forall k\in V \tag{9}$$

$$x_{F_p^i,F_p^i}^k-x_{F_d^i,F_d^i}^k=0,\ \forall i,j\in O,\forall F_p^i,F_p^j,F_d^j,F_d^i\in F \tag{10}$$

$$\sum_{k=1}^{K}\sum_{i=1}^{l_k}y_i^k d_i^k=O,\ \forall i\in l_k,\forall k\in V \tag{11}$$

2.2 MOEA/D 框架

本文使用了一种基于分解的多目标进化算法(MOEA/D[27])。该算法使用 Tchebycheff 方法[27]将 MOP 分解为一组单目标优化问题:

$$g^{tch}(\boldsymbol{x}|\boldsymbol{\lambda},\boldsymbol{z})=\min[\max_{1\leqslant i\leqslant m}\{\lambda_i|f_i(\boldsymbol{x})-\boldsymbol{z}_i|\}] \quad (12)$$

s.t. $\boldsymbol{x}\in\boldsymbol{\Omega}$

其中,$\boldsymbol{\lambda}=\{\lambda_1,\lambda_2,\cdots,\lambda_m\}$是一个权重向量,$m$ 是目标数,$\boldsymbol{\Omega}$ 是目标空间,$\lambda_i\geqslant 0, i=1,\cdots,m$,且$\sum_{i=1}^{m}\lambda^i=1$。$f_i(\boldsymbol{x})$表示第 i 个目标函数,$\boldsymbol{z}=\{z_1,z_2,\cdots,z_m\}$是记录每个目标函数最小值的理想点。

算法 1 给出了 MOEA/D 算法的框架。MOEA/D 有 5 个输入:1)子问题的个数 N;2) N 个权重向量集合 $\boldsymbol{\lambda}=\{\lambda_1,\lambda_2,\cdots,\lambda_m\}$;3)权重向量的邻域个数 T;4)从邻居中选择父代解的概率 δ;5)子代个体替换解的最大数量 n_r。MOEA/D 的输出是一个近似帕累托解集$(\boldsymbol{x}^1,\cdots,\boldsymbol{x}^N)$。

算法 1 MOEA/D(N,λ,T,δ,n_r)

1. /* 初始化 */
2. /* B=Initialize-Weight-Vector(N,λ,T) */
3. for i=1:N do
4. B(i)={i_1,…,i_T},$\boldsymbol{\lambda}_1^i,\cdots,\boldsymbol{\lambda}_T^i$是 $\boldsymbol{\lambda}^i$的 T 个最近的权重向量;
5. end
6. 产生初始化种群 $\mathbf{x}^1,\cdots,\mathbf{x}^N$并评估其目标值;
7. /* z=Initialize-Ideal-Point(x) */
8. 通过 $z^j=\min_{1\leqslant i\leqslant N} f^j(\mathbf{x}^i)$初始化参考点 $\mathbf{z}=\{z^1,z^2,\cdots,z^m\}$;
9. while 终止条件未达到 do
10. for i=1:N do
11. /* 进化过程 */
12. P=rand$_1$< δ? $\mathbf{B}^i$:{1,…,N};
13. r_1,r_2←从 P 中随机选择两个个体索引;
14. 依次对解 $\mathbf{x}^i,\mathbf{x}^{r_1},\mathbf{x}^{r_2}$执行 DE 和 PM 操作得到子代个体 $\mathbf{y}$;
15. /* 更新理想点 z=Update-Ideal-Point(y,z) */
16. 对 j=1,…,m,若 $z^j>f^j(\mathbf{y})$,则 $z^j=f^j(\mathbf{y})$;
17. /* 更新种群 x=Update-Population(P,y,x,n_r) */
18. 设 c=0;
19. while c < n_r & P ≠ Ø do
20. j ← 从 P 中随机选择一个个体索引;
21. if $g^{tch}(\mathbf{y}|\lambda^j,\mathbf{z})<g^{tch}(\mathbf{x}^j|\lambda^j,\mathbf{z})$
22. $\mathbf{x}^j=\mathbf{y}$;
23. c=c+1 并将 j 从 P 中删除;
24. end if
25. end while
26. end for
27. end while

输出:一组近似帕累托解集$(\mathbf{x}^1,\cdots,\mathbf{x}^N)$

算法 1 可以分为 4 部分。

1)初始化(第 2－8 行):首先,计算每个权重向量之间的欧氏距离,并获取与每个权重向量最近的 T 个权重向量,将其存储在集合 B 中(第 3－5 行)。然后,在搜索空间中随机均匀地生成初始种群(第 6 行),每个目标的当前最小值形成理想点 z(第 8 行)。

2)进化过程(第 12－14 行):对于每个子问题 $i=1,\cdots,N$,从范围[0,1]中产生一个随机实数 $rand$。随后,设:

$$P=\begin{cases}B(i), & rand<\delta\\ \{1,\cdots,N\}, & \text{其他}\end{cases} \quad (13)$$

从 P 中随机选取两个索引 r_1 和 r_2(第 13 行)。然后,使用差分进化(DE)和基于多项式的变异(PM),以 $\boldsymbol{x}^i,\boldsymbol{x}^{r_1},\boldsymbol{x}^{r_2}$ 作为输入,生成后代 $\boldsymbol{y}$(第 14 行)。

3)更新理想点(第 16 行):对所有 $j=1,\cdots,m$,若 $z^j>f^j(\boldsymbol{y})$,则 $z^j=f^j(\boldsymbol{y})$。

4)更新种群(第 18－25 行):先初始化解替换计数 c 为 0,并从 P 中随机选取一个索引 j,然后利用式(12)计算 $\boldsymbol{x}^j$ 和 $\boldsymbol{y}$ 的聚合函数值。若 $g^{tch}(\boldsymbol{y}|\lambda^j,\boldsymbol{z})<g^{tch}(\boldsymbol{x}^j|\lambda^j,\boldsymbol{z})$,则将 $\boldsymbol{x}^j$ 替换为 $\boldsymbol{y}$,且将 c 加 1,并将索引 j 从 P 中移除。其中,替换解的最大数量为 n_r。

2.3 研究动机

工业 DPDP 是一种具有垛口、时间窗、容量和后进先出装载等复杂约束的单目标优化问题(Single-objective Optimization Problem,SOP),给现有的 DPDP 算法带来了重大挑战。大多数 DPDP 算法在搜索过程中只优化一个加权目标函数,难以保持解的多样性,从而很容易陷入局部最优区域而停止收敛。

在进化计算中,将 SOP 转化为 MOP 的过程被称为"多目标化"。这一过程使得我们可以使用多目标优化算法来寻找目标 SOP 的最优解[28-30]。实现多目标化的一种常用方法是将原始目标分解为多个子目标[28,31],然后同时求解多个子目标,以找到目标 SOP 的最优解。多目标化方法通过求解 MOP(而不是目标 SOP),可以为目标 SOP 产生改进解,因为此方法提高了解集的多样性[28-29,32],避免了陷入局部最优区域[28,33]。

受此启发,我们在优化过程中将目标 DPDP 转化为 MOP 以增强解的多样性,进而提高优化性能。具体地,本文将目标 DPDP 转化为包含两个子目标(f_1 和 f_2)的 MOP,并使用一组权重向量 $\lambda^1,\lambda^2,\cdots,\lambda^N$ 将 MOP 进一步分解为多个子问题。为了解决这些子问题,使用交叉操作来增强解集的多样性,然后利用局部搜索来加快收敛速度。如图 2 所示,在每个时间段里,不同的子问题在整个优化过程中相互协作,最终选择具有最小 TC 值的解作为最优解(用红点标识),来完成目标 DPDP 的一些现有订单。通过这种方式,MOEA/D-ES 可以增强解的多样性,避免陷入局部最优区域,从而为目标 DPDP 搜索到更好的候选解。

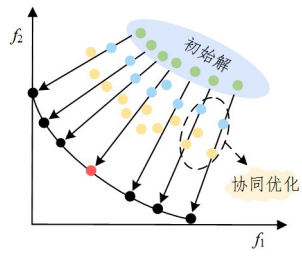


图 2 使用 MOP 的子问题协同优化原始 DPDP 的简单示例(电子版为彩图)

Fig. 2 Simple example of co-optimizing the original DPDP using the sub-problems of MOP

3 算法设计

本章首先介绍 MOEA/D-ES 的算法框架，然后依次介绍两个主要组成部分（即交叉操作和局部搜索）。MOEA/D-ES 的框架流程图如图 3 所示。在初始化后，MOEA/D-ES 迭代运行交叉操作、局部搜索和更新种群，直到满足终止条件。这些组件确保生成的解始终满足时间窗、容量、垛口和 LIFO 装载等约束。

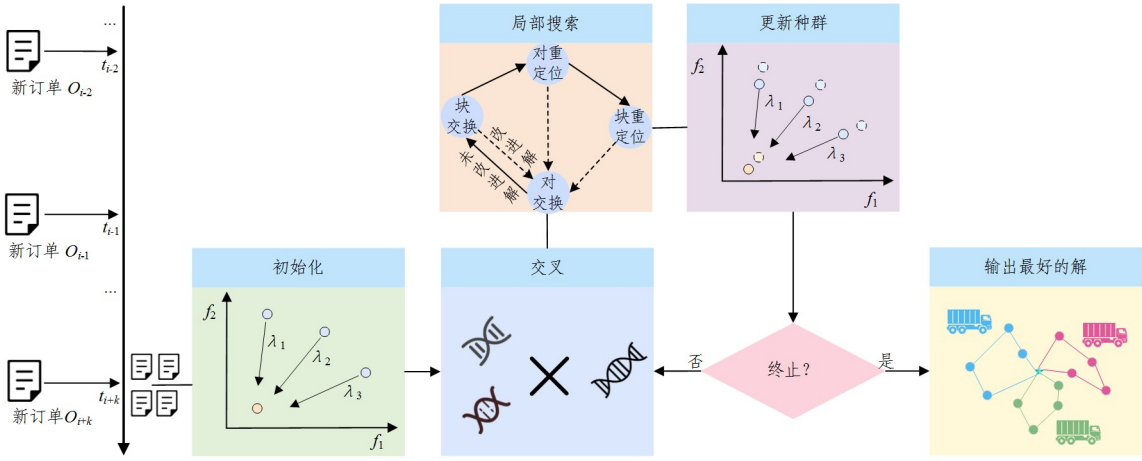


图 3 MOEA/D-ES 框架

Fig. 3 Framework of MOEA/D-ES

3.1 整体框架

在工业 DPDP 中，通常将一天划分为多个固定的时间段，因此动态到来的订单可在最近的一个时间段内完成。在每个时间段内，当新订单动态发布时，运行 MOEA/D-ES 获取包含车辆路径规划的候选解。算法 2 阐述了 MOEA/D-ES 的总体框架。MOEA/D-ES 的初始化阶段主要由 3 部分组成：生成每个子问题的 T 个最近的权重向量（第 1 行），初始化种群（第 2—5 行），以及初始化理想点（第 6 行）。MOEA/D-ES 生成每个子问题的 T 个最近权重向量和初始化理想点的过程与 MOEA/D[27] 一致。MOEA/D-ES 在初始化种群时，先从存档中恢复上一时间段的解 x_{resotore}，并生成新订单的 N 个随机排列序列。根据每个序列，使用 CI 算法将新订单插入到 x_{resotore} 中生成 N 个绑定子问题的解，即初始种群。在使用 CI 初始化每个子问题的候选解时的评估标准是式（12），从而保证了初始种群的多样性。停止准则是达到最大迭代次数或最大运行时间。初始化后，MOEA/D-ES 依次迭代以下操作（第 11—14 行）：交叉操作、局部搜索、更新种群和理想点。交叉操作和局部搜索的详细介绍见后文。

算法 2 MOEA/D-ES 总体框架

输入：DPDP 实例，N（种群大小），m（目标数）

输出：x*（当前时间段的解）

1. B=Initialize-Weight-Vector(N,λ,T)；//算法 1 函数
2. /* 初始化种群 */
3. 从存档中恢复上个时间段的解 x_{resore}；
4. 生成新订单的 N 个随机排列序列；
5. 根据每个序列，将新订单用 CI 法插入到 x_{resore} 中生成种群 $\mathbf{x}^1,\cdots,\mathbf{x}^N$；
6. z=Initialize-Ideal-Point(x)；//算法 1 函数
7. while 不满足算法终止条件 do
8. for i=1,…,N do
9. P=rand$_1$＜δ? $\mathbf{B}^i$:{1,…,N}；
10. 从 P 随机选择两个个体索引 r_1，r_2；
11. x_{child}←交叉操作($\mathbf{x}^{r_1}$,$\mathbf{x}^{r_2}$,K)；//算法 3
12. x_{child}←局部搜索(x_{child})；//算法 4
13. z=Update-Ideal-Point(x_{child},z)；//算法 1 函数
14. x=Update-Population(P,x_{child},x,n_r)；//算法 1 函数
15. end for
16. end while
17. x^*←Select_Best(**x**)；//根据 TC 选最好的解
18. 将 x^* 保存至存档。

3.2 交叉操作

如算法 3 所示，MOEA/D-ES 设计了一种简单而有效的交叉算子来求解工业 DPDP。对于子代解 x_{child} 中每辆车的路径，交叉算子依次随机复制任意父代解中的对应路径给 x_{child}；同时删除 x_{child} 中重复的节点，将 x_{child} 修复成可行解（第 1—5 行）。上述两个步骤重复进行，直到 x_{child} 中的所有车辆路径都从父代中继承。然后，将剩余的未分配订单（如果存在）插入 x_{child} 以形成完整的可行候选解。具体来说，首先将所有剩余的未分配订单保存在集合 U 中（第 6 行）。接下来，将 U 中的每个订单 o_i 插入到 x_{child} 中对应问题函数值（式（12））最小的位置（第 7—9 行）。最后，交叉操作生成一个完整的可行解 x_{child}。由于工业 DPDP 的复杂性，传统方法很容易陷入局部最优。本文设计的交叉算子能够生成具有多样性的子代，从而降低陷入局部最优区域的概率。随后，使用局部搜索以更有效地挖掘全局最优解。

算法 3 交叉操作(p_1,p_2,K)

输入：p_1，p_2（父代解），所有车辆的数量 K

输出：x_{child}（子代解）

1. x_{child}←Ø
2. for k=1 to K do
3. 从 p_1，p_2 随机复制车辆 k 的路径 r_k 给 x_{child}；
4. 删除冗余节点修正 x_{child} 成为可行解；
5. end for

6. U←剩下未指派的订单；
7. for 订单 $o_i \in U$ do
8. 将 o_i 插入到 x_{child} 中式(12)值最小的位置；
9. end for

3.3 局部搜索

MOEA/D-ES 的局部搜索方法由 4 种局部搜索策略组成：对交换、块交换、块重定位和对重定位。这些局部搜索策略的详细描述见文献[34]。如图 4 所示，x, y, z, w 分别表示一个具体订单，而“+”“−”则分别表示车辆对订单的取、送货操作。具体来说，图 4(a)是初始路径；对交换操作交换两个对(如对$[x^+, x^-]$和$[w^+, w^-]$)以创建一条新路径，如图4(b)所示；块交换操作交换两个块(如块 B_x 和 B_w)以创建新路径，如图 4(c)所示；对重定位操作将一个对(如对$[z^+, z^-]$)重新插入到另一个可行的位置以创建新的路线，如图 4(d)所示；块重定位操作将一个块(如块 B_x)重新插入到另一个可行位置以创建新路径，如图 4(e)所示。

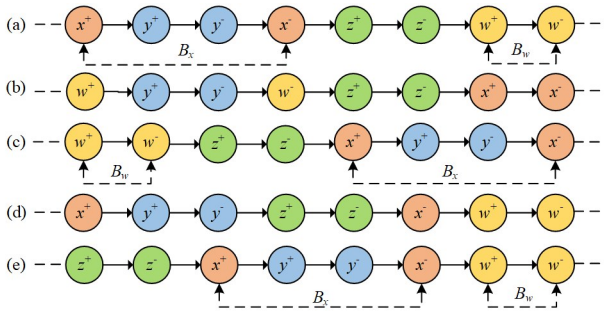


图 4 初始路径以及不同策略形成的路径

Fig. 4 Initial route and route formed by different strategies

本文使用的局部搜索伪代码如算法 4 所示，其中 4 种局部搜索策略的使用顺序如图 3 的局部搜索模块所示。对于当前解，使用任意一种局部搜索策略都可生成当前解的一个邻域解。在 MOEA/D-ES 每次迭代生成子代 x_{child} 后，为了找到更小 TC(式(4))值的改进解，将连续执行第一种局部搜索策略(对交换)用于生成改进解 x_1，并将其赋值给 x_{child}。如果对交换策略没有生成改进解，则使用第二种局部搜索策略(块交换)对 x_{child} 的邻域进行局部搜索。如果块交换策略找到改进解 x_1，则更新 x_{child}，并且将重新使用第一种局部搜索策略继续搜索改进解。否则，执行第三种局部搜索策略(块重定位)来搜索 x_{child} 的改进解。如果第三种局部搜索策略找到改进解 x_1，块重定位策略会更新 x_{child} 并重新使用第一种局部搜索策略继续搜索改进解。否则，执行第四种局部搜索策略(对重定位)来执行与前边局部搜索策略类似的操作。当第四种局部搜索策略无法找到改进解 x_1 时，则可认为 x_{child} 在 4 种策略的搜索空间中已达到局部最优，可结束局部搜索。因此，本文的局部搜索方法通过迭代使用 4 种搜索策略，能够更细粒度地搜索改进解。

算法 4 局部搜索(x_{child})

输入：x_{child}(子代解)

输出：x_{imp}(改进解)

1. 令 idx=0，$r_{max}=4$；//r_{max}是局部搜索算子的个数
2. 给定一组局部搜索算子 N_r，r=1,2,…,r_{max}；
3. //N_1：对交换，N_2：块交换，N_3：块重定位，N_4：对重定位
4. while idx<r_{max} do
5. r=1；
6. while r<r_{max} do
7. 使用算子 N_r 对 x_{child} 进行局部搜素，得到局部最优解 x_1；
8. if TC(x_1)<TC(x_{child}) then
9. $x_{child}=x_1$；
10. idx=0；
11. break；
12. else
13. idx=idx+1；
14. end if
15. r=r+1；
16. end while
17. end while
18. $x_{imp}=x_{child}$；

4 实验研究

4.1 测试集和评价准则

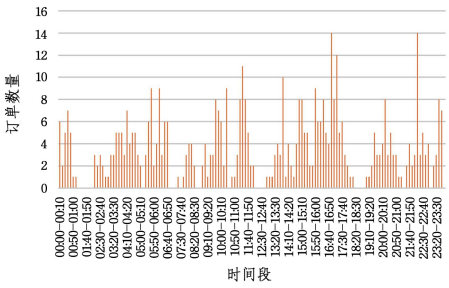
在工业场景中，具有实际约束的 DPDP 是很常见的。华为公司在 ICAPS 2021 举办的物流配送竞赛中提出了一个基于华为实际工业场景的 DPDP 变体和对应的问题测试集(HW 问题集)[26]，同时提供一个模拟器用于模拟订单发布和确定所有车辆的当前信息(车辆的剩余容量、运输的物料信息、当前位置等信息)。该场景下的 DPDP 具有实际约束，如垛口、时间窗、装载容量和 LIFO 装载等约束，这使得问题更具挑战性。这个 HW 问题集来自华为真实系统，包含 30 天的历史数据。如表 1 所列，HW 问题集包含 4 种同质车辆数量规模(5,20,50 和 100)和 8 种订单数量规模(50,100,300,500,1 000,2 000,3 000 和 4 000)。

表 1 HW 问题集的车辆数和订单数

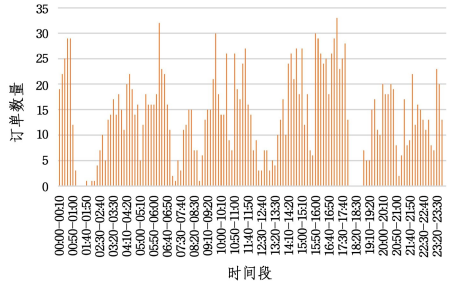
Table 1 Number of vehicles and orders of HW problems

问题	车辆	订单	规模
HW1-HW8	5	50	小
HW9-HW16	5	100	小
HW17-HW24	20	300	小
HW25-HW32	20	500	小
HW33-HW40	50	1 000	中
HW41-HW48	50	2 000	中
HW49-HW56	100	3 000	大
HW57-HW64	100	4 000	大

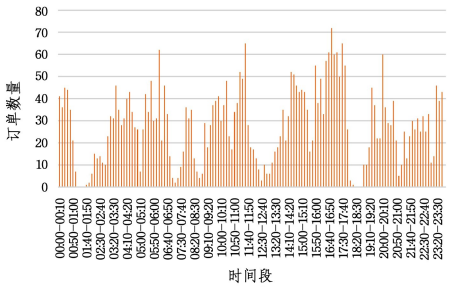
为了便于介绍和分析，本文根据订单数量将这 64 个实例分为 3 种问题规模(小、中、大)。图 5 展示了 3 种问题规模的最后一个实例(HW32，HW48，HW64)在一天内动态生成新订单的分布情况(以每个时间段 10 min 进行统计)。其中，小规模问题中的新订单在一天的 144 个时间段内的分布较为稀疏，且每个时间段的订单数量较少；对于中规模的问题，其新订单在一天内的分布比小规模问题更为密集，订单数量有所增加；而在大规模问题中，新订单在一天中的分布不仅更密集，而且每个时间段生成新订单的数量更大。



(a)小规模



(b)中规模



(c)大规模

图 5　3 种不同规模 HW 问题产生的新订单的分布

Fig. 5　Distributions of new orders generated by three HW problems with different scales

在每个 HW 问题中，多个工厂可以在一天内动态生成配送订单。在求解 HW 问题集时，目标是将所有订单分配给一个车队，以最小的订单总超时和车辆的平均行驶距离完成所有订单。因此，HW 问题集的性能指标是 TC，即式(4)。

4.2　对比算法和实验设置

为了验证 MOEA/D-ES 的有效性，将 4 种最新的用于求解 HW 问题集的启发式算法作为 MOEA/D-ES 的对比算法，其中包括 ICAPS 2021 物流配送比赛中获得金奖、银奖、铜奖的算法和基准算法。有关该比赛的详细介绍在其官网[1)]可见。下面对各个对比算法进行简要介绍。

1)金奖算法(Gold)[19]：该算法采用可变邻域搜索策略，由 4 种局部搜索策略和一种扰动策略组成。它在不同车辆路线之间持续地交换取货节点团和送货节点团，并在相同车辆路线内交换订单节点，以获得更好的结果。取(送)货节点团是相邻且地址相同的取(送)货节点。

2)银奖算法(Silver)：该算法将 DPDP 转化为背包问题，并通过检查是否达到交货时间和车辆容量的阈值来分配订单。

3)铜奖算法(Bronze)：该算法通过 CI 算法构造 DPDP 的初始解，然后通过销毁-重构的局部搜索策略改进路线规划解。

4)基准算法(Baseline)：该算法采用传统处理静态 PDP 的方法[6]求解 DPDP。每次订单发布时，该算法将所有未完成订单的取货和送货节点相邻地插入到可用车辆中，生成可行解。

所有对比算法和 MOEA/D-ES 在每个时间段(10 min)内处理订单，前后两个时间段的订单分配结果并不相互独立，主要原因是上次的订单分配结果会影响当前车辆的剩余容量和后续订单的相对位置。算法运行时间限制为 10min，这是工业 DPDP 的实际需求，也是设计求解算法的一个硬约束和对比条件。MOEA/D-ES 和所有对比算法分别独立运行 HW 问题集的每个实例 20 次。

4 个对比算法(Gold，Silver，Bronze 和 Baseline)的源码可以在比赛官方网站[2)]上找到，它们和 MOEA/D-ES 采用了同一问题设置。1)订单迟到 1h 的惩罚值：$\alpha=10000/3600$；2)每个工厂的最大垛口数量为 6；3)车辆最大装载容量 Q 为 16；4)最大运行时间设置为 10 min。这些设置与 ICAPS 2021 DPDP 竞赛中的设置一致。

每个对比算法的一些特有参数均按照其相应源码中的建议进行设置，确保了算法对比的公平性。此外，针对所有测试问题，MOEA/D-ES 的一些参数设置如下：种群大小(N)为 6；邻域大小(T)为 2。停止准则包括：1)最大迭代次数为 50 次；2)最大运行时间为 600 s。

本文在使用 Ubuntu 16.04 LTS 操作系统、Intel(R) Core (TM) i7-6700 CPU @ 3.40GHz×8 处理器以及 9.7GB 内存环境下，进行了对比实验。

4.3　实验结果和分析

本节为了方便描述与分析，将各对比算法在 HW 问题集上的实验结果分为小、中和大规模 3 部分进行描述与分析。分类情况和每类测试问题的订单分布如表 1 和图 5 所示。表 2－表 5 列出了各个算法在 3 个规模 HW 问题集上独立运行 20 次获得的实验结果，优异的结果以粗体突出。其中，“B. Cost”“Ave. Cost”分别表示各对比算法在 20 次独立运行中获得的最好 TC 值、平均 TC 值和标准差。此外，为了得到统计学上合理的结论，在显著性水平 $\alpha=0.05$ 的情况下进行 Wilcoxon 秩和检验，以比较 MOEA/D-ES 与其他对比算法的结果在统计学上的差异。标准差后边的“＋”“－”和“～”表示对比算法的结果明显优于、差于和近似于 MOEA/D-ES 的统计检验结果。在每个表的倒数第二行，“best/all”的值分别统计了每个算法在“B. Cost”和“Ave. Cost”上的最好结果个数；此外，每个表最后一行的“＋/－/～”总结了相应算法在测试问题上明显好于、差于和近似于 MOEA/D-ES 的个数。

4.3.1　小规模 HW 问题

表 2 列出了各个算法在 32 个小规模 HW 问题上独立运行 20 次获得的实验结果。从表 2 中可以发现，MOEA/D-ES

1) https://competition.huaweicloud.com/information/1000041411/circumstance

2) https://competition.huaweicloud.com/information/1000041411/Winning

在大部分小规模 HW 问题上表现最好，明显优于其他 4 个对比算法。在 B. Cost 方面，MOEA/D-ES 在 32 个小规模 HW 问题中获得 27 个最好结果，验证了该算法的优越性能。而对比算法中只有 Gold 在 5 个 HW 问题上表现最好，Silver，Bronze 和 Baseline 在 32 个小规模 HW 问题中均未取得最好的结果。此外，MOEA/D-ES 在 5 个未取得最好结果的问题上取得的 TC 值与最好结果的差距很小，这说明了 MOEA/D-ES 具有较强的鲁棒性。在 Ave. Cost 方面，MOEA/D-ES 在 32 个 HW 问题上获得 26 个最好结果。具体来说，在 32 个小规模 HW 问题的一对一比较中，MOEA/D-ES 在 25，32，31，32 个测试问题上的表现优于 Gold，Silver，Bronze 和 Baseline，只在 3 个问题上的表现劣于 Gold。

表 2 MOEA/D-ES，Gold，Silver，Brone 和 Baseline 在小规模 HW 问题上的对比结果

Table 2 Comparison results of Gold，Silver，Brone，Baseline，and MOEA/D-ES on mall-scale HW problems

问题	Gold		Silver		Brone	
	B. Cost	Ave. Cost	B. Cost	Ave. Cost	B. Cost	Ave. Cost
HW1	1.35×10^{2}	$1.35\times10^{2}(0.00)-$	2.30×10^{3}	$2.30\times10^{3}(0.00)-$	1.30×10^{2}	$1.30\times10^{2}(0.00)-$
HW2	9.56×10	$9.56\times10(0.00)-$	3.05×10^{4}	$3.05\times10^{4}(0.00)-$	9.14×10	$9.14\times10(0.00)-$
HW3	9.68×10	$9.68\times10(0.00)-$	3.58×10^{4}	$3.58\times10^{4}(0.00)-$	9.65×10	$9.65\times10(0.00)-$
HW4	9.46×10	$9.46\times10(0.00)-$	5.49×10^{3}	$5.49\times10^{3}(0.00)-$	1.04×10^{2}	$1.04\times10^{2}(0.00)-$
HW5	**3.31×10^{3}**	$3.31\times10^{3}(2.11)\sim$	1.69×10^{4}	$1.69\times10^{4}(0.00)-$	5.45×10^{3}	$5.45\times10^{3}(0.00)-$
HW6	1.05×10^{2}	$1.05\times10^{2}(0.00)-$	4.76×10^{3}	$4.76\times10^{3}(0.00)-$	1.18×10^{2}	$1.18\times10^{2}(0.00)-$
HW7	4.39×10^{3}	$4.39\times10^{3}(0.00)-$	1.29×10^{4}	$1.28\times10^{4}(0.00)-$	7.36×10^{3}	$7.36\times10^{3}(0.00)-$
HW8	6.88×10	$6.88\times10(0.00)-$	7.97×10^{2}	$7.97\times10^{2}(0.00)-$	7.69×10^{2}	$7.69\times10^{2}(0.00)-$
HW9	**1.52×10^{2}**	**$1.52\times10^{2}(0.00)+$**	1.81×10^{5}	$1.81\times10^{5}(0.00)-$	8.48×10^{3}	$8.48\times10^{3}(0.00)-$
HW10	1.63×10^{5}	$1.78\times10^{5}(1.47\times10^{4})-$	1.77×10^{6}	$1.77\times10^{6}(0.00)-$	1.87×10^{5}	$1.87\times10^{5}(0.00)-$
HW11	1.98×10^{2}	$1.98\times10^{2}(0.00)-$	1.81×10^{5}	$1.81\times10^{5}(0.00)-$	3.04×10^{3}	$3.04\times10^{3}(0.00)-$
HW12	5.30×10^{4}	$5.30\times10^{4}(0.00)-$	5.67×10^{5}	$5.67\times10^{5}(0.00)-$	8.42×10^{4}	$8.42\times10^{4}(0.00)-$
HW13	7.18×10^{3}	$7.18\times10^{3}(0.00)-$	2.20×10^{5}	$2.20\times10^{5}(0.00)-$	2.77×10^{2}	**$2.77\times10^{2}(0.00)\sim$**
HW14	9.39×10^{3}	$9.39\times10^{3}(0.00)-$	2.37×10^{5}	$2.37\times10^{5}(0.00)-$	7.82×10^{3}	$7.83\times10^{3}(0.00)-$
HW15	**1.36×10^{4}**	$2.91\times10^{4}(1.42\times10^{4})\sim$	7.93×10^{5}	$7.93\times10^{5}(0.00)-$	1.49×10^{5}	$1.49\times10^{5}(0.00)-$
HW16	1.68×10^{4}	$2.83\times10^{4}(6.46\times10^{3})-$	8.54×10^{5}	$8.54\times10^{5}(0.00)-$	5.78×10^{4}	$5.78\times10^{4}(0.00)-$
HW17	8.17×10	$8.17\times10(0.00)-$	9.65×10	$9.65\times10(0.00)-$	4.11×10^{2}	$4.11\times10^{2}(0.00)-$
HW18	8.22×10	$8.22\times10(0.00)-$	1.12×10^{2}	$1.12\times10^{2}(0.00)-$	8.56×10^{3}	$8.56\times10^{3}(0.00)-$
HW19	1.09×10^{2}	**$1.09\times10^{2}(0.00)\sim$**	4.78×10^{2}	$4.78\times10^{2}(0.00)-$	4.15×10^{3}	$4.15\times10^{3}(0.00)-$
HW20	**3.30×10^{3}**	**$3.30\times10^{3}(0.00)+$**	3.98×10^{3}	$3.98\times10^{3}(0.00)-$	1.62×10^{4}	$1.62\times10^{4}(0.00)-$
HW21	1.13×10^{2}	$1.13\times10^{2}(4.47\times10^{-3})-$	2.34×10^{3}	$2.34\times10^{3}(0.00)-$	1.85×10^{4}	$1.85\times10^{4}(0.00)-$
HW22	1.65×10^{3}	**$1.65\times10^{3}(0.00)-$**	3.32×10^{3}	$3.32\times10^{3}(0.00)-$	2.11×10^{4}	$2.11\times10^{4}(0.00)-$
HW23	1.05×10^{2}	$1.05\times10^{2}(0.00)\sim$	6.87×10^{3}	$6.87\times10^{3}(0.00)-$	8.09×10^{2}	$8.09\times10^{2}(0.00)-$
HW24	9.44×10	$9.44\times10(0.00)-$	1.29×10^{3}	$1.29\times10^{3}(0.00)-$	2.00×10^{3}	$2.00\times10^{3}(0.00)-$
HW25	1.03×10^{4}	$1.03\times10^{4}(0.00)-$	9.24×10^{4}	$9.24\times10^{4}(0.00)-$	2.15×10^{4}	$2.15\times10^{4}(0.00)-$
HW26	8.95×10^{3}	$8.95\times10^{3}(1.09)-$	2.76×10^{5}	$2.76\times10^{5}(0.00)-$	5.40×10^{4}	$5.40\times10^{4}(0.00)-$
HW27	1.34×10^{2}	$1.34\times10^{2}(0.00)-$	1.84×10^{4}	$1.84\times10^{4}(0.00)-$	1.88×10^{4}	$1.88\times10^{4}(0.00)-$
HW28	**7.10×10^{3}**	**$7.10\times10^{3}(3.12)+$**	9.45×10^{3}	$9.45\times10^{3}(0.00)-$	1.49×10^{4}	$1.49\times10^{4}(0.00)-$
HW29	6.40×10^{3}	$8.26\times10^{3}(1.04\times10^{3})-$	1.63×10^{5}	$1.63\times10^{5}(0.00)-$	4.24×10^{4}	$4.24\times10^{4}(0.00)-$
HW30	1.19×10^{2}	$1.19\times10^{2}(4.47\times10^{-3})-$	7.40×10^{4}	$7.40\times10^{4}(0.00)-$	2.11×10^{4}	$2.11\times10^{4}(0.00)-$
HW31	2.26×10^{4}	$2.83\times10^{4}(3.19\times10^{3})-$	1.47×10^{5}	$1.47\times10^{5}(0.00)-$	2.33×10^{4}	$2.33\times10^{4}(0.00)-$
HW32	7.67×10^{3}	$7.67\times10^{3}(0.00)-$	6.04×10^{4}	$6.04\times10^{4}(0.00)-$	1.96×10^{4}	$1.96\times10^{4}(0.00)-$
best/all	5/32	5/32	0/32	0/32	0/32	1/32
+/−/∼		3/25/4		0/32/0		0/31/1

问题	Baseline		MOEAD-ES	
	B. Cost	Ave. Cost	B. Cost	Ave. Cost
HW1	1.58×10^{5}	$1.58\times10^{5}(0.00)-$	**1.17×10^{2}**	**$1.18\times10^{2}(1.11)$**
HW2	8.98×10^{4}	$8.98\times10^{4}(0.00)-$	**8.88×10**	**$8.88\times10(0.00)$**
HW3	3.38×10^{4}	$3.38\times10^{4}(0.00)-$	**9.41×10**	**$9.41\times10(0.00)$**
HW4	4.18×10^{4}	$4.18\times10^{4}(0.00)-$	**9.45×10**	**$9.45\times10(0.00)$**
HW5	1.47×10^{5}	$1.47\times10^{5}(0.00)-$	3.31×10^{3}	**$3.31\times10^{3}(1.35)$**
HW6	5.24×10^{4}	$5.24\times10^{4}(0.00)-$	**1.05×10^{2}**	**$1.05\times10^{2}(0.00)$**
HW7	9.58×10^{4}	$9.58\times10^{4}(0.00)-$	**4.32×10^{3}**	**$4.37\times10^{3}(2.97\times10)$**
HW8	3.88×10^{4}	$3.88\times10^{4}(0.00)-$	**6.39×10**	**$6.39\times10(0.00)$**
HW9	2.12×10^{6}	$2.12\times10^{6}(0.00)-$	1.65×10^{2}	$1.68\times10^{2}(1.53)$
HW10	5.42×10^{6}	$5.42\times10^{6}(0.00)-$	**7.47×10^{4}**	**$1.05\times10^{5}(1.70\times10^{4})$**
HW11	1.92×10^{6}	$1.92\times10^{6}(0.00)-$	**1.61×10^{2}**	**$1.73\times10^{2}(9.02)$**
HW12	3.25×10^{6}	$3.25\times10^{6}(0.00)-$	**8.61×10^{3}**	**$2.66\times10^{4}(1.14\times10^{4})$**
HW13	2.50×10^{6}	$2.50\times10^{6}(0.00)-$	**1.73×10^{2}**	$3.23\times10^{2}(2.10^{\times}10^{2})$
HW14	2.61×10^{6}	$2.61\times10^{6}(0.00)-$	**1.50×10^{2}**	**$1.59\times10^{2}(7.31)$**
HW15	3.36×10^{6}	$3.36\times10^{6}(0.00)-$	1.92×10^{4}	**$2.29\times10^{4}(3.44\times10^{3})$**
HW16	3.24×10^{6}	$3.24\times10^{6}(0.00)-$	**1.04×10^{4}**	**$1.38\times10^{4}(2.22\times10^{3})$**
HW17	1.31×10^{6}	$1.31\times10^{6}(0.00)-$	**7.34×10**	**$7.59\times10(1.55)$**

（续表）

问题	Baseline		MOEAD-ES	
	B. Cost	Ave. Cost	B. Cost	Ave. Cost
HW18	9.72×10^5	$9.72\times10^5(0.00)-$	**7.97×10**	**$8.11\times10(1.01)$**
HW19	1.93×10^6	$1.93\times10^6(0.00)-$	**1.06×10^2**	$1.10^{\times}10^2(2.96)$
HW20	$2.10^{\times}10^6$	$2.10^{\times}10^6(0.00)-$	3.30×10^3	$3.44\times10^3(1.29\times10^2)$
HW21	2.60×10^6	$2.60\times10^6(0.00)-$	**9.74×10**	**$1.02\times10^2(2.50)$**
HW22	2.29×10^6	$2.29\times10^6(0.00)-$	**1.64×10^3**	**$1.64\times10^3(1.94)$**
HW23	2.32×10^6	$2.32\times10^6(0.00)-$	**1.00×10^2**	$1.06\times10^2(3.66)$
HW24	1.44×10^6	$1.44\times10^6(0.00)-$	**8.53×10**	**$8.66\times10(1.43)$**
HW25	3.69×10^7	$3.69\times10^7(0.00)-$	**5.93×10^3**	**$8.09\times10^3(1.96\times10^3)$**
HW26	3.54×10^7	$3.54\times10^7(0.00)-$	**4.49×10^3**	**$5.96\times10^3(1.84\times10^3)$**
HW27	3.13×10^7	$3.13\times10^7(0.00)-$	**1.19×10^2**	**$1.22\times10^2(3.46)$**
HW28	2.86×10^7	$2.86\times10^7(0.00)-$	7.12×10^3	$7.13\times10^3(2.33)$
HW29	3.64×10^7	$3.64\times10^7(0.00)-$	**5.92×10^3**	**$6.34\times10^3(2.86\times10^2)$**
HW30	3.19×10^7	$3.19\times10^7(0.00)-$	**$1.10^{\times}10^2$**	**$1.11\times10^2(8.98\times10^{-1})$**
HW31	3.80×10^7	$3.80\times10^7(0.00)-$	**1.45×10^4**	**$1.57\times10^4(7.73\times10^2)$**
HW32	3.63×10^7	$3.63\times10^7(0.00)-$	**5.69×10^3**	**$6.56\times10^3(8.13\times10^2)$**
best/all	0/32	0/32	27/32	26/32
+/−/∼		0/32/0		−

此外，在小规模测试问题上，Baseline 在 B. Cost 和 Ave. Cost 上的表现明显不如其他算法。这是因为工业 DPDP 除了订单动态特性、大规模变量和众多复杂约束外，不同时间段的订单分配结果并不是相互独立的，前一时段的订单分配结果会影响当前阶段车辆的剩余容量和当前车辆与后续订单的相对位置。因此，通过连续重新优化每个静态子问题来解决 DPDP 的 Baseline 需要消耗大量计算资源，导致其效率较低，表现较差。这再次证明了将新订单动态地插入到已规划路线的启发式算法，比传统的通过连续重新优化每个静态子问题来解决 DPDP 的方法更加高效。

4.3.2 中规模 HW 问题

表 3 列出了各个算法在 16 个中规模 HW 问题上独立运行 20 次的结果。从表 3 中可以发现，MOEA/D-ES 在中规模 HW 问题上表现出色。在 B. Cost 方面，MOEA/D-ES 在 16 个中规模 HW 问题上取得了 14 个最好结果，而对比算法中只有 Gold 和 Silver 各获得 1 个最好结果。特别在问题 HW34，HW35，HW38 和 HW40 上，MOEA/D-ES 表现出了明显的优势，其性能优于所有对比算法。在 Ave. Cost 方面，MOEA/D-ES 在 16 个实例上获得 14 个最好结果，进一步验证了 MOEA/D-ES 的有效性和稳定性。另一方面，对于所有中规模 HW 问题的平均 TC 值，MOEA/D-ES 为 1.640×10^4，Gold 为 2.628×10^4，Silver 为 8.535×10^4，Bronze 为 3.734×10^5，而 Baseline 是 2.379×10^8。与 Gold，Silver，Bronze 和 Baseline 相比，MOEA/D-ES 的优化目标值（TC）分别提高了 37.59%，80.78%，95.61% 和 99.99%。

表 3 MOEA/D-ES，Gold，Silver，Brone 和 Baseline 在中规模 HW 问题上的对比结果

Table 3 Comparison results of Gold，Silver，Brone，Baseline，and MOEA/D-ES on medium-scale HW problems

问题	Gold		Silver		Brone	
	B. Cost	Ave. Cost	B. Cost	Ave. Cost	B. Cost	Ave. Cost
HW33	1.59×10^3	$1.59\times10^3(4.47\times10^{-3})-$	7.03×10^3	$7.03\times10^3(0.00)-$	9.50×10^4	$9.50\times10^4(0.00)-$
HW34	1.05×10^4	$1.05\times10^4(3.45\times10^{-1})\sim$	1.05×10^4	$1.05\times10^4(0.00)\sim$	5.44×10^4	$5.44\times10^4(0.00)-$
HW35	3.44×10^3	$3.44\times10^3(0.00)-$	1.58×10^4	$1.58\times10^4(0.00)-$	1.04×10^5	$1.04\times10^5(0.00)-$
HW36	1.75×10^4	$1.75\times10^4(0.00)\sim$	2.32×10^4	$2.32\times10^4(0.00)-$	1.13×10^5	$1.13\times10^5(0.00)-$
HW37	1.12×10^4	$1.12\times10^4(0.00)\sim$	**5.14×10^3**	**$5.14\times10^3(0.00)+$**	9.57×10^4	$9.57\times10^4(0.00)-$
HW38	1.50×10^4	$1.56\times10^4(3.89\times10^2)-$	3.01×10^4	$3.01\times10^4(0.00)-$	9.42×10^4	$9.42\times10^4(0.00)-$
HW39	**1.49×10^4**	**$1.49\times10^4(0.00)+$**	2.23×10^4	$2.23\times10^4(0.00)-$	8.52×10^4	$8.52\times10^4(0.00)-$
HW40	1.02×10^4	$1.02\times10^4(0.00)\sim$	2.43×10^4	$2.43\times10^4(0.00)-$	1.17×10^5	$1.17\times10^5(0.00)-$
HW41	2.93×10^4	$2.93\times10^4(0.00)-$	1.24×10^5	$1.24\times10^5(0.00)-$	5.49×10^5	$5.49\times10^5(0.00)-$
HW42	3.59×10^4	$4.45\times10^4(7.94\times10^3)-$	1.64×10^5	$1.64\times10^5(0.00)-$	6.01×10^5	$6.01\times10^5(0.00)-$
HW43	6.05×10^4	$7.55\times10^4(2.32\times10^4)-$	1.67×10^5	$1.67\times10^5(0.00)-$	5.51×10^5	$5.51\times10^5(0.00)-$
HW44	4.96×10^4	$7.09\times10^4(1.23\times10^4)-$	1.91×10^5	$1.91\times10^5(0.00)-$	7.16×10^5	$7.16\times10^5(0.00)-$
HW45	3.63×10^4	$4.32\times10^4(9.61\times10^3)-$	1.81×10^5	$1.81\times10^5(0.00)-$	6.16×10^5	$6.16\times10^5(0.00)-$
HW46	3.09×10^4	$3.09\times10^4(1.30\times10^{-1})-$	1.77×10^5	$1.77\times10^5(0.00)-$	7.98×10^5	$7.98\times10^5(0.00)-$
HW47	2.68×10^4	$2.68\times10^4(0.00)-$	1.04×10^5	$1.04\times10^5(0.00)-$	6.03×10^5	$6.03\times10^5(0.00)-$
HW48	1.33×10^4	$1.45\times10^4(1.06\times10^3)\sim$	1.20×10^5	$1.20\times10^5(0.00)-$	7.82×10^5	$7.82\times10^5(0.00)-$
best/all	1/16	1/16	1/16	1/16	0/16	0/16
+/−/∼		1/10/5		1/14/1		0/16/0

问题	Baseline		MOEAD-ES	
	B. Cost	Ave. Cost	B. Cost	Ave. Cost
HW33	3.86×10^7	$3.86\times10^7(0.00)-$	**1.39×10^3**	**$1.55\times10^3(8.85\times10)$**
HW34	3.98×10^7	$3.98\times10^7(0.00)-$	**6.32×10^3**	**$8.30\times10^3(1.95\times10^3)$**
HW35	5.11×10^7	$5.11\times10^7(0.00)-$	**8.05×10**	**$1.42\times10^3(1.84\times10^3)$**
HW36	4.67×10^7	$4.67\times10^7(0.00)-$	**1.29×10^4**	**$1.57\times10^4(2.29\times10^3)$**
HW37	4.67×10^7	$4.67\times10^7(0.00)-$	7.63×10^3	$1.08\times10^4(1.79\times10^3)$
HW38	4.70×10^7	$4.70\times10^7(0.00)-$	**6.69×10^3**	**$1.18\times10^4(3.06\times10^3)$**
HW39	4.43×10^7	$4.43\times10^7(0.00)-$	1.65×10^4	$1.75\times10^4(9.91\times10^2)$
HW40	3.73×10^7	$3.73\times10^7(0.00)-$	**9.34×10^3**	**$1.00\times10^4(4.77\times10^2)$**
HW41	4.48×10^8	$4.48\times10^8(0.00)-$	**1.84×10^4**	**$2.05\times10^4(1.53\times10^3)$**
HW42	4.23×10^8	$4.23\times10^8(0.00)-$	**2.00×10^4**	**$2.08\times10^4(5.30\times10^2)$**
HW43	4.24×10^8	$4.24\times10^8(0.00)-$	**2.76×10^4**	**$2.85\times10^4(6.34\times10^2)$**

（续表）

问题	Baseline		MOEAD-ES	
	B. Cost	Ave. Cost	B. Cost	Ave. Cost
HW44	4.27×10^8	$4.27\times10^8(0.00)$ −	**3.62×10^4**	**$4.27\times10^4(3.70\times10^3)$**
HW45	4.26×10^8	$4.26\times10^8(0.00)$ −	**1.71×10^4**	**$2.14\times10^4(2.89\times10^3)$**
HW46	4.50×10^8	$4.50\times10^8(0.00)$ −	**1.38×10^4**	**$1.68\times10^4(1.73\times10^3)$**
HW47	4.24×10^8	$4.24\times10^8(0.00)$ −	**1.73×10^4**	**$2.06\times10^4(1.95\times10^3)$**
HW48	4.32×10^8	$4.32\times10^8(0.00)$ −	**1.27×10^4**	**$1.42\times10^4(1.37\times10^3)$**
best/all	0/16	0/16	14/16	14/16
+/−/∼		0/16/0		−

在中规模 HW 问题中，MOEA/D-ES 在 B. Cost 和 Ave. Cost 方面的表现都显著优于 Gold，Silver，Bronze 和 Baseline，这是因为 4 个对比算法仅优化了单一的加权目标（即 TC）。这些算法虽然在求解这种复杂工业 DPDP 上可以获得较好的可行解，却容易陷入局部最优区域。而 MOEA/D-ES 将目标 DPDP 转化为一个包含两个子目标（f_1 和 f_2）的 MOP，并将该 MOP 进一步分解为多个子问题。每个子问题采用交叉操作和局部搜索协同优化，既提高了解的多样性，又避免了陷入局部最优区域。

4.3.3 大规模 HW 问题

由表 1 可知，每个大规模 HW 问题包含的订单数量巨大，多达 3000 至 4000 个订单，凸显了问题的艰巨性。此外，从图 5 中可以看出，每个大规模 HW 问题生成的订单分布非常密集，这表明这些问题具有很强的动态性。因此，求解大规模 HW 问题的难度很高。

表 4 列出了各个算法在 16 个大规模 HW 问题上独立运行 20 次获得的实验结果。在 B. Cost 方面，MOEA/D-ES 在 16 个大规模 HW 问题上取得了 14 个最好结果。在 Ave. Cost 方面，MOEA/D-ES 在 16 个实例上获得了 11 个最好结果，而 Gold 只在 HW59，HW60 和 HW62 这 3 个问题上取得最好结果，Silve 只在 HW50 和 HW54 这 2 个问题上取得最好结果。总的来说，MOEA/D-ES 的平均性能比其他算法更为优越。另一方面，在所有大规模 HW 问题的平均 TC 上，MOEA/D-ES 为 5.758×10^6，Gold 为 6.155×10^6，Silver 为 9.791×10^6，Bronze 为 8.533×10^6，而 Baseline 为 1.394×10^9。与 Gold，Silver，Bronze 和 Baseline 相比，MOEA/D-ES 的性能提高了 6.45%，41.19%，32.51% 和 99.59%。

表 4 MOEA/D-ES，Gold，Silver，Brone 和 Baseline 在大规模 HW 问题上的对比结果

Table 4 Comparison results of Gold，Silver，Brone，Baseline，and MOEA/D-ES on large-scale HW problems

问题	Gold		Silver		Brone	
	B. Cost	Ave. Cost	B. Cost	Ave. Cost	B. Cost	Ave. Cost
HW49	1.23×10^6	$1.44\times10^6(1.39\times10^5)$ −	1.18×10^6	$1.18\times10^6(0.00)$ −	1.60×10^6	$1.60\times10^6(0.00)$ −
HW50	7.44×10^5	$7.95\times10^5(7.84\times10^4)$ ∼	**5.73×10^5**	**$5.73\times10^5(0.00)$ +**	2.51×10^6	$2.51\times10^6(0.00)$ −
HW51	7.44×10^4	$2.07\times10^5(1.38\times10^5)$ −	5.16×10^5	$5.16\times10^5(0.00)$ −	1.34×10^6	$1.34\times10^6(0.00)$ −
HW52	7.01×10^5	$7.53\times10^5(4.25\times10^4)$ −	4.48×10^5	$4.48\times10^5(0.00)$ ∼	2.03×10^6	$2.03\times10^6(0.00)$ −
HW53	8.47×10^4	$9.56\times10^4(1.47\times10^4)$ −	3.17×10^5	$3.17\times10^5(0.00)$ −	1.93×10^6	$1.93\times10^6(0.00)$ −
HW54	1.93×10^6	$1.98\times10^6(6.16\times10^4)$ ∼	1.63×10^6	**$1.63\times10^6(0.00)$ ∼**	2.00×10^6	$2.00\times10^6(0.00)$ −
HW55	2.62×10^5	$3.02\times105(2.79\times10^4)$ −	7.30×10^5	$7.30\times10^5(0.00)$ −	1.98×10^6	$1.98\times10^6(0.00)$ −
HW56	1.92×10^6	$1.99\times10^6(6.09\times10^4)$ −	1.84×10^6	$1.84\times10^6(0.00)$ −	2.16×10^6	$2.16\times10^6(0.00)$ −
HW57	9.83×10^6	$1.01\times10^7(2.75\times10^5)$ ∼	2.24×10^7	$2.24\times10^7(0.00)$ −	1.65×10^7	$1.65\times10^7(0.00)$ −
HW58	9.37×10^6	$1.01\times10^7(8.15\times10^5)$ ∼	1.32×10^7	$1.32\times10^7(0.00)$ −	1.12×10^7	$1.12\times10^7(0.00)$ ∼
HW59	**7.77×10^6**	**$8.81\times10^6(6.02\times10^5)$ ∼**	1.24×10^7	$1.24\times10^7(0.00)$ −	9.54×10^6	$9.54\times10^6(0.00)$ ∼
HW60	1.07×10^7	**$1.09\times10^7(1.52\times10^5)$ ∼**	2.15×10^7	$2.15\times10^7(0.00)$ −	1.32×10^7	$1.32\times10^7(0.00)$ −
HW61	6.03×10^6	$7.18\times10^6(7.96\times10^5)$ −	9.10×10^6	$9.10\times10^6(0.00)$ −	7.21×10^6	$7.21\times10^6(0.00)$ −
HW62	8.80×10^6	**$9.12\times10^6(2.34\times10^5)$ ∼**	1.45×10^7	$1.45\times10^7(0.00)$ −	1.09×10^7	$1.09\times10^7(0.00)$ ∼
HW63	1.34×10^7	$1.76\times10^7(2.69\times10^6)$ ∼	3.09×10^7	$3.09\times10^7(0.00)$ −	2.50×10^7	$2.50\times10^7(0.00)$ −
HW64	1.57×10^7	$1.71\times10^7(8.68\times10^5)$ ∼	2.54×10^7	$2.54\times10^7(0.00)$ −	2.74×10^7	$2.74\times10^7(0.00)$ −
best/all	1/16	3/16	1/16	2/16	0/16	0/16
+/−/∼		0/7/9		1/13/2		0/13/3

问题	Baseline		MOEAD-ES	
	B. Cost	Ave. Cost	B. Cost	Ave. Cost
HW49	9.20×10^8	$9.20\times10^8(0.00)$ −	**4.03×10^5**	**$5.58\times10^5(9.78\times10^4)$**
HW50	9.20×10^8	$9.20\times10^8(0.00)$ −	7.96×10^5	$8.52\times10^5(7.54\times10^4)$
HW51	9.13×10^8	$9.13\times10^8(0.00)$ −	**2.80×10^4**	**$5.65\times10^4(2.48\times10^4)$**
HW52	9.20×10^8	$9.20\times10^8(0.00)$ −	**2.54×10^5**	**$3.96\times10^5(1.10\times10^5)$**
HW53	9.55×10^8	$9.55\times10^8(0.00)$ −	**3.12×10^4**	**$4.53\times10^4(1.23\times10^4)$**
HW54	9.11×10^8	$9.11\times10^8(0.00)$ −	**1.53×10^6**	$1.76\times10^6(1.84\times10^5)$
HW55	8.97×10^8	$8.97\times10^8(0.00)$ −	**6.61×10^4**	**$1.31\times10^5(9.58\times10^4)$**
HW56	9.45×10^8	$9.45\times10^8(0.00)$ −	**1.50×10^6**	**$1.62\times10^6(1.01\times10^5)$**
HW57	1.83×10^9	$1.83\times10^9(0.00)$ −	**8.44×10^6**	**$9.16\times10^6(8.67\times10^5)$**
HW58	1.81×10^9	$1.81\times10^9(0.00)$ −	**8.52×10^6**	**$9.55\times10^6(1.36\times10^6)$**
HW59	1.90×10^9	$1.90\times10^9(0.00)$ −	8.29×10^6	$9.37\times10^6(7.64\times10^5)$
HW60	1.83×10^9	$1.83\times10^9(0.00)$ −	**1.03×10^7**	$1.18\times10^7(1.02\times10^6)$
HW61	1.81×10^9	$1.81\times10^9(0.00)$ −	**4.32×10^6**	**$5.43\times10^6(8.21\times10^5)$**
HW62	1.87×10^9	$1.87\times10^9(0.00)$ −	**7.60×10^6**	$9.18\times10^6(1.64\times10^6)$
HW63	1.91×10^9	$1.91\times10^9(0.00)$ −	**1.33×10^7**	**$1.60\times10^7(1.92\times10^6)$**
HW64	1.96×10^9	$1.96\times10^9(0.00)$ −	**1.49×10^7**	**$1.62\times10^7(1.38\times10^6)$**
best/all	0/16	0/16	14/16	11/16
+/−/∼		0/16/0		−

MOEA/D-ES 在求解大规模 HW 问题时表现优异的原因是：其可以通过 N 个子问题的独立探索和协作获得多样性较好的解集，所以具备很强的搜索能力。因此，与其他 4 个对比算法相比，MOEA/D-ES 可以定位到更有潜力的解空间区域，从而能够找到更好的路径规划解。

4.3.4 综合对比

为了全面比较所有算法的性能，根据各个算法在每种规模 HW 问题上独立运行 20 次得到的平均 TC 值，使用 Friedman 测试[35]来确定它们的平均性能排名。排名结果如图 6 所示，排名越低表示性能越好。为了便于观察，MOEA/D-ES 在 3 种规模上的排名用红线进行连接。

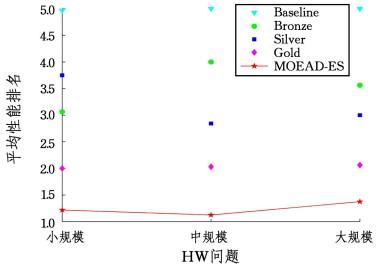


图 6 对比算法在 3 种规模 HW 问题上的平均性能排名（电子版为彩图）

Fig. 6 Average performance ranks of comparison algorithms on HW problems at three scales

从图 6 可以明显看出，MOEA/D-ES 在所有规模的 HW 问题上表现最佳。此外，MOEA/D-ES 在中规模 HW 问题上的平均性能表现略好于在小规模和大规模 HW 问题上。具体原因分析如下：在小规模 HW 问题中，每一时间段动态生成的订单数量相对较少，MOEA/D-ES 不容易陷入局部最优。而 MOEA/D-ES 将目标 DPDP 转化为 MOP，进而将 MOP 分解为多个子问题来增加种群多样性的策略，在求解小规模 HW 问题时会消耗额外的计算资源，这使得其在有限的时间内未能及时向最优解区域收敛。在大规模 HW 问题中，由于订单数量巨大且快速动态生成，问题的解空间异常复杂，这时 MOEA/D-ES 有可能陷入次优解区域而未能搜索到最优解。相反，在中等规模 HW 问题中，每个时间段动态生成的订单数量适中，这时结合交叉操作和局部搜索能更好地平衡种群的多样性和收敛性，从而可以找到近似最优解。尽管如此，MOEA/D-ES 在求解小规模和大规模 HW 问题时依然显著优于其他对比算法。

另外，Brone 在小规模 HW 问题上的平均性能优于 Silver，而在中规模和大规模 HW 问题上却显著差于 Silver。这是因为与 Silver 相比，Brone 每次使用销毁-重构启发式时，插入操作的取货节点和送货节点在路线上都是连续的，这导致 Brone 的搜索空间较小，容易陷入局部最优区域。在小规模 HW 问题中，解空间相对较小，Brone 通过启发式方法可以取得比 Silver 算法更好的效果。然而，随着 HW 问题规模的增大，Brone 容易陷入于局部最优区域，这时 Silver 在求解这些问题时更具优势。

4.4 更多讨论

4.4.1 组件有效性实验

本文对 MOEA/D-ES 中的核心组件进行了消融实验，以评估其有效性。本实验主要包括以下 3 种变体。

1）CI：只包含初始化种群模块，即先生成当前时段新订单的 N 个随机排列序列，再按 N 个序列的顺序使用 CI 方法生成 N 个绑定子问题的解。

2）Crossover：在 CI 方法获取初始种群后，使用交叉算子生成子代个体。

3）LS：交叉算子生成子代个体后，使用 4 种混合局部搜索策略对该个体进行优化。

为了直观比较 3 种变体的性能表现，使用 Friedman 测试[35]对 3 种变体在 3 种规模 HW 问题上的 TC 结果进行评估，获得它们在不同规模 HW 问题上的平均性能排名，如图 7 所示。

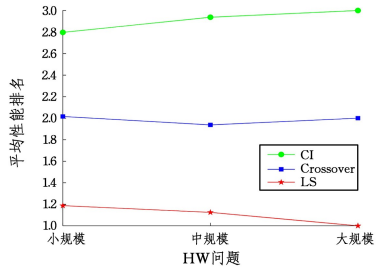


图 7 消融组件在 3 种规模 HW 问题上的平均性能排名

Fig. 7 Average performance ranks of ablation components on HW problems at three scales

一方面，变体 Crossover 在 3 个规模 HW 问题上的平均排名都明显优于变体 CI，这表明交叉操作不仅能有效提高解的多样性，还能提高解的质量；同时，在每种规模的 HW 问题上，变体 LS 与变体 Crossover 相比，平均性能有很大的提升，这说明了 MOEA/D-ES 中的局部搜索组件具有挖掘更高质量解的潜力，也说明了仅靠插入启发式算法和交叉操作来求解工业 DPDP 很难获得近似最优解。而加入局部搜索算法后，可以高效搜索工业 DPDP 的解空间，从而找到更好的解。

另一方面，对于 LS 变体，随着 HW 问题规模的增加，其排名逐渐降低，这表明与其他两种变体相比，LS 的优势在较大规模的问题上表现更为明显。特别在大规模 HW 问题上，LS 的平均性能排名为 1，即相比于其他两种变体，LS 在 16 个大规模问题中的表现都是最好的。这是因为规模越大，HW 问题越复杂，变体 CI 越容易陷入局部最优区域，而变体 Crossover 虽然能够提高解集的多样性，但收敛性不足。相反，变体 LS 通过将交叉操作与局部搜索相结合，能够有效平衡收敛性和多样性，不仅能够跳出局部最优区域，还能找到更高质量的解。

同时，在 3 种不同规模的 HW 问题上，本文对 3 种变体（CI，Crossover 和 LS）的 TC 值进行显著性检验，汇总结果如表 5 所列。其中，变体 Crossover 在 HW 问题上的性能优于变体 CI，再次表明了 MOEA/D-ES 中交叉操作的有效性；而变体 LS 在 3 种规模 HW 问题上的结果显著优于变体 CI 和 Crossover，进一步突显了 MOEA/D-ES 的局部搜索在搜索高质量候选解方面的强大能力。

表 5 变体 CI, Crossover 和 LS 在 3 种规模 HW 问题上关于 TC 值的显著性检验结果

Table 5 Significance test results of variants CI, Crossover, and LS with respect to TC values on HW problems at three scales

问题	CI vs Crossover +/−/∼	CI vs LS +/−/∼	Crossover vs LS +/−/∼
小规模	0/7/25	1/26/5	2/26/4
中规模	0/8/8	0/11/5	0/11/5
大规模	0/13/3	0/16/0	0/16/0

4.4.2 泛化性实验

为了评估 MOEA/D-ES 的泛化性和有效性，将其用于解决复杂物流配送场景下的带时间窗和同时取送货的车辆路径问题[36](Vehicle Routing Problem with Simultaneous Pickup-delivery and Time Windows，VRPPDT)。本文采用一个大规模 VRPPDT 数据集[37]作为测试问题，该数据集来自京东物流配送系统。在该系统中，除了需派送客户购买的商品外，还需要在预定的时间窗内收取客户商品(例如有缺陷的商品或需要维修的商品)。该数据集的具体设置如表 6 所列。该问题的目标是最小化一个加权成本 WC(即车辆调度成本和运输成本的总和)[37]。

表 6 京东数据集的特性

Table 6 Features of Jingdong dataset

问题	车辆	订单	容量
F201-F204	500	200	2.5
F401-F404	500	400	2.5
F601-F604	500	600	2.5
F801-F804	500	800	2.5
F1001-F1004	500	1000	2.5

在京东(JD)数据集上，本文将 MOEA/D-ES 与最近提出的 5 种算法(MATE[37]，EMA[38]，CCMO[39]，GOLD[19] 和 MBEA[40])进行了比较。对比算法的参数根据原论文中给出的推荐参数进行设置。所有算法在大规模京东数据集的 20 个实例上独立运行 20 次。MOEA/D-ES 和比较算法的评价次数都设置为 18000。这里使用 WC 作为评价指标来评估各个算法在求解 VRPPDT 时的性能。

图 8 根据每个算法在京东数据集上的平均 WC 值，使用 Friedman 测试[35]绘制了各个算法在每个测试问题上的平均性能排名，其中 MOEA/D-ES 的排名用红线连接起来，以便观察。从图 8 可以看出，MOEA/D-ES 在所有 20 个测试问题上取得 18 个最好的排名。此外，表 7 给出了 MOEA/D-ES 和 5 种对比算法的 Wilcoxon 秩和检验结果。从表 7 可知，MOEA/D-ES 在 20 个问题中也取得了 18 个最好结果。这是因为 MOEA/D-ES 将目标 VRPPDT 转化为 MOP，进而将 MOP 分解为多个子问题的策略，增加了种群多样性；此外，为求解每个子问题，MOEA/D-ES 结合交叉操作和局部搜索可以更好地平衡种群的多样性和收敛性，从而找到更高质量的解。

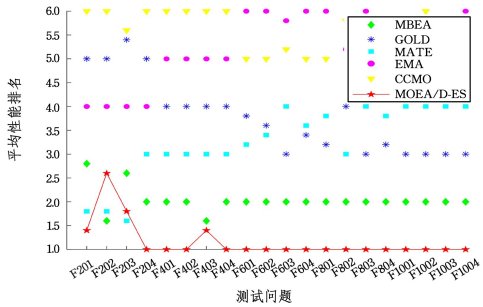


图 8 MOEA/D-ES 和 5 个对比算法在京东数据集每个实例上的平均性能排名(电子版为彩图)

Fig. 8 Average performance ranking of MOEA/D-ES and five comparison algorithms on eachcase of Jingdong dataset

表 7 MOEA/D-ES 和 5 种对比算法在京东数据集上的对比实验结果

Table 7 Comparative experimental results of MOEA/D-ES and five comparison algorithms on Jingdong dataset

问题	MBEA	GOLD	MATE	EMA	CCMO	MOEA/D-ES
F201	$6.68\times10^4(4.23\times10^2)\sim$	$7.55\times10^4(0.00)-$	$6.66\times10^4(5.60\times10^2)\sim$	$6.84\times10^4(0.00)-$	$7.93\times10^4(1.28\times10^3)-$	**$6.63\times10^4(1.74\times10^2)$**
F202	**$6.64\times10^4(2.30\times10^2)\sim$**	$7.44\times10^4(0.00)-$	$6.66\times10^4(2.65\times10^2)\sim$	$7.04\times10^4(0.00)-$	$7.78\times10^4(1.41\times10^3)-$	$6.67\times10^4(2.37\times10^2)$
F203	$6.77\times10^4(4.14\times10^2)\sim$	$8.13\times10^4(0.00)-$	**$6.70\times10^4(7.95\times10^2)\sim$**	$7.28\times10^4(0.00)-$	$8.15\times10^4(1.18\times10^3)-$	$6.72\times10^4(3.87\times10^2)$
F204	$6.62\times10^4(1.95\times10^2)-$	$7.57\times10^4(0.00)-$	$6.72\times10^4(4.66\times10^2)-$	$7.03\times10^4(0.00)-$	$7.71\times10^4(8.29\times10^2)-$	**$6.59\times10^4(1.05\times10^2)$**
F401	$1.24\times10^5(7.01\times10^2)\sim$	$1.42\times10^5(0.00)-$	$1.35\times10^5(2.52\times10^2)-$	$1.48\times10^5(0.00)-$	$1.55\times10^5(1.35\times10^3)-$	**$1.22\times10^5(3.86\times10^2)$**
F402	$1.28\times10^5(7.36\times10^2)-$	$1.47\times10^5(0.00)-$	$1.40\times10^5(4.58\times10^2)-$	$1.53\times10^5(0.00)-$	$1.59\times10^5(2.23\times10^3)-$	**$1.27\times10^5(5.65\times10^2)$**
F403	$1.22\times10^5(1.26\times10^3)\sim$	$1.39\times10^5(0.00)-$	$1.34\times10^5(5.30\times10^2)-$	$1.51\times10^5(0.00)-$	$1.53\times10^5(1.19\times10^3)-$	**$1.21\times10^5(6.58\times10^2)$**
F404	$1.25\times10^5(2.32\times10^2)-$	$1.39\times10^5(0.00)-$	$1.36\times10^5(5.07\times10^2)-$	$1.45\times10^5(0.00)-$	$1.57\times10^5(1.03\times10^3)-$	**$1.24\times10^5(4.92\times10^2)$**
F601	$1.85\times10^5(5.13\times10^2)-$	$2.13\times10^5(0.00)-$	$2.12\times10^5(9.08\times10^2)-$	$2.49\times10^5(0.00)-$	$2.40\times10^5(2.34\times10^3)-$	**$1.83\times10^5(5.71\times10^2)$**
F602	$1.89\times10^5(1.08\times103)-$	$2.18\times10^5(0.00)-$	$2.17\times10^5(1.97\times10^3)-$	$2.54\times10^5(0.00)-$	$2.45\times10^5(2.75\times10^3)-$	**$1.88\times10^5(2.96\times10^2)$**
F603	$1.90\times10^5(6.98\times10^2)-$	$2.14\times10^5(0.00)-$	$2.15\times10^5(3.92\times10^2)-$	$2.48\times10^5(0.00)-$	$2.46\times10^5(2.42\times10^3)-$	**$1.87\times10^5(7.57\times10^2)$**
F604	$1.90\times10^5(1.55\times10^3)-$	$2.15\times10^5(0.00)-$	$2.15\times10^5(1.11\times10^3)-$	$2.52\times10^5(0.00)-$	$2.42\times10^5(3.68\times10^3)-$	**$1.87\times10^5(1.01\times10^3)$**
F801	$2.23\times10^5(1.39\times10^3)-$	$2.43\times10^5(0.00)-$	$2.45\times10^5(2.38\times10^3)-$	$3.05\times10^5(0.00)-$	$2.92\times10^5(4.11\times10^3)-$	**$2.16\times10^5(1.01\times10^3)$**
F802	$2.21\times10^5(2.11\times10^3)-$	$2.46\times10^5(0.00)-$	$2.43\times10^5(1.13\times10^3)-$	$2.90\times10^5(0.00)-$	$2.95\times10^5(5.09\times10^3)-$	**$2.14\times10^5(1.52\times10^3)$**
F803	$2.21\times10^5(1.70\times10^3)-$	$2.42\times10^5(0.00)-$	$2.45\times10^5(1.76\times10^3)-$	$3.02\times10^5(0.00)-$	$2.94\times10^5(2.36\times10^3)-$	**$2.15\times10^5(6.74\times10^2)$**
F804	$2.19\times10^5(1.88\times10^3)-$	$2.38\times10^5(0.00)-$	$2.40\times10^5(1.65\times10^3)-$	$2.91\times10^5(0.00)-$	$2.89\times10^5(3.38\times10^3)-$	**$2.12\times10^5(1.91\times10^3)$**
F1001	$3.29\times10^5(1.69\times10^3)-$	$3.48\times10^5(0.00)-$	$3.60\times10^5(2.83\times10^3)-$	$4.46\times10^5(0.00)-$	$4.47\times10^5(3.09\times10^3)-$	**$3.13\times10^5(9.65\times10^2)$**
F1002	$3.27\times10^5(2.88\times10^3)-$	$3.49\times10^5(0.00)-$	$3.59\times10^5(2.29\times10^3)-$	$4.40\times10^5(0.00)-$	$4.42\times10^5(1.88\times10^3)-$	**$3.10\times10^5(1.27\times10^3)$**
F1003	$3.29\times10^5(2.68\times10^3)-$	$3.44\times10^5(0.00)-$	$3.64\times10^5(3.75\times10^3)-$	$4.41\times10^5(0.00)-$	$4.44\times10^5(2.23\times10^3)-$	**$3.11\times10^5(1.72\times10^3)$**
F1004	$3.26\times10^5(2.68\times10^3)-$	$3.47\times10^5(0.00)-$	$3.56\times10^5(5.20\times10^3)-$	$4.48\times10^5(0.00)-$	$4.41\times10^5(4.58\times10^3)-$	**$3.10\times10^5(1.63\times10^2)$**
best/all	1/20	0/20	1/20	0/20	0/20	18/20
+/−/∼	0/15/5	0/20/0	0/17/3	0/20/0	0/20/0	−

结束语 本文提出了一种基于分解的多目标进化算法，用于求解具有垛口、时间窗、装载容量和 LIFO 装载等约束的工业 DPDP。该方法先将目标 DPDP 建模为包含两个子目标的 MOP，并将该 MOP 进一步分解为多个子问题。然后，对这些子问题同时进行求解：利用交叉操作增强解的多样性，再使用局部搜索加快收敛速度。在每一时段，从种群中选择一个最好的解来完成当前未完成的订单。在 64 个华为工业物流测试问题上的仿真结果验证了本文算法的高效性；同时，在 20 个京东物流大规模配送问题的实验结果验证了本文算法的泛化性和有效性。最后，消融实验也进一步验证了本文算法中各组件的有效性。

由于工业 DPDP 变体每日的订单具有一定的分布规律，因此我们计划在未来的研究中将延迟配送功能和预测订单策略嵌入到 MOEA/D-ES，以进一步提高其求解工业 DPDP 的性能。此外，鉴于深度强化学习在处理序列决策方面的强大能力，将其用于求解 DPDP 将是我们未来的另一个研究方向。具体来说，可将深度强化学习融入到现有的启发式算法框架中，以增强算法在复杂动态环境中决策的高效性和鲁棒性。例如，可通过使用深度 Q 网络评估多种潜在操作的长期效益，自动学习选择能够最大化预期回报的局部搜索算子来迭代改进当前的局部最优解。同时，深度强化学习模型通过与现实世界的数据进行交互来实时学习，这对于解决 DPDP 这类动态复杂问题尤为重要。

参 考 文 献

[1] FENG L, HUANG Y X, ZHOU L, et al. Explicit evolutionary multitasking for combinatorial optimization: A case study on capacitated vehicle routing problem[J]. IEEE Transactions on Cybernetics, 2020, 51(6): 3143-3156.

[2] YANG H X, GAO J, SHAO E L. Vehicle routing problem with time window of takeaway food considering one-order-multi-product order delivery[J]. Computer Science, 2022, 49(S1): 191-198.

[3] BERBEGLIA G, CORDEAU J F, LAPORTE G. Dynamic pickup and delivery problems[J]. European Journal of Operational Research, 2010, 202(1): 8-15.

[4] SAVELSBERGH M W, SOL M. The general pickup and delivery problem[J]. Transportation Science, 1995, 29(1): 17-29.

[5] CAI J C, ZHU Q L, LIN Q Z, et al. A Survey of Dynamic Pickup and Delivery Problems[J]. Neurocomputing, 2023(14): 1. 1-1. 16.

[6] AVELSBERGH M, SOL M. Drive: Dynamic routing of independent vehicles[J]. Operations Research, 1998, 46(4): 474-490.

[7] SWIHART M R, PAPASTAVROU J D. A stochastic and dynamic model for the single-vehicle pick-up and delivery problem [J]. European Journal of Operational Research, 1999, 114(3): 447-464.

[8] ARSLAN A M, AGATZ N, KROON L, et al. Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers[J]. Transportation Science, 2019, 53(1): 222-235.

[9] MITROVIĆ-MINIĆ S, KRISHNAMURTI R, LAPORTE G. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows[J]. Transportation Research Part B: Methodological, 2004, 38(8): 669-685.

[10] GENDREAU M, GUERTIN F, POTVIN J Y, et al. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries[J]. Transportation Research Part C: Emerging Technologies, 2006, 14(3): 157-174.

[11] MA Y, HAO X T, HAO J Y, et al. A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems[J]. Advances in Neural Information Processing Systems, 2021, 34: 23609-23620.

[12] SU Z Y, LI W T, LI J C, et al. Heterogeneous fleet vehicle scheduling problems for dynamic pickup and delivery problem with time windows in shared logistics platform: Formulation, instances and algorithms[J]. International Journal of Systems Science: Operations & Logistics, 2022, 9(2): 199-223.

[13] XU X F, WEI Z F. Dynamic pickup and delivery problem with transshipments and LIFO constraints[J]. Computers & Industrial Engineering, 2023, 175: 108835.

[14] TAO Y, ZHUO H B, LAI X F. The pickup and delivery problem with multiple depots and dynamic occasional drivers in crowdshipping delivery[J]. Computers & Industrial Engineering, 2023, 182: 109440.

[15] ULMER M W, THOMAS B W, CAMPBELL A M, et al. The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times[J]. Transportation Science, 2021, 55(1): 75-100.

[16] DU J H, ZHANG Z Q, WANG X, et al. A hierarchical optimization approach for dynamic pickup and delivery problem with LIFO constraints[J]. Transportation Research Part E: Logistics and Transportation Review, 2023, 175: 103131.

[17] LI X J, LUO W L, YUAN M X, et al. Learning to optimize industry-scale dynamic pickup and delivery problems[C]// 2021 IEEE 37th International Conference on Data Engineering (ICDE). 2021: 2511-2522.

[18] GHIANI G, MANNI A, MANNI E. A scalable anticipatory policy for the dynamic pickup and delivery problem[J]. Computers & Operations Research, 2022, 147: 105943.

[19] CAI J C, ZHU Q L, LIN Q Z. Variable neighborhood search for a new practical dynamic pickup and delivery problem[J]. Swarm and Evolutionary Computation, 2022, 75: 101182.

[20] VONOLFEN S, AFFENZELLER M. Distribution of waiting time for dynamic pickup and delivery problems[J]. Annals of Operations Research, 2016, 236: 359-382.

[21] PUREZA V, LAPORTE G. Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows [J]. INFOR: Information Systems and Operational Research, 2008, 46(3): 165-175.

[22] GHIANI G, MANNI E, QUARANTA A, et al. Anticipatory algorithms for same-day courier dispatching[J]. Transportation Research Part E: Logistics and Transportation Review, 2009, 45(1): 96-106.

[23] SÁEZ D, CORTÉS C E, NÚÑEZ A. Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery pro-

blem based on genetic algorithms and fuzzy clustering[J]. Computers & Operations Research, 2008, 35(11): 3412-3438.
[24] SCHILDE M, DOERNER K F, HARTL R F, et al. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports[J]. Computers & Operations Research, 2011, 38(12): 1719-1730.
[25] SCHILDE M, DOERNER K F, HARTL R F. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem[J]. European Journal of Operational Research, 2014, 238(1): 18-30.
[26] HAO J Y, LU J W, LI X J, et al. Introduction to the dynamic pickup and delivery problem benchmark--ICAPS 2021 competition[J]. arXiv: 2202.01256, 2022.
[27] LI H, ZHANG Q F. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II [J]. IEEE Transactions on Evolutionary Computation, 2008, 13(2): 284-302.
[28] KNOWLES J D, WATSON R A, CORNE D W. Reducing local optima in single-objective problems by multi-objectivization [C]// International Conference on Evolutionary Multi-Criterion Optimization. 2001: 269-283.
[29] LOCHTEFELD D F, CIARALLO F W. Multiobjectivization via helper-objectives with the tunable objectives problem[J]. IEEE Transactions on Evolutionary Computation, 2011, 16(3): 373-390.
[30] MA X L, HUANG Z T, LI X D, et al. Multiobjectivization of single-objective optimization in evolutionary computation: a survey[J]. IEEE Transactions on Cybernetics, 2021, 53(6): 3702-3715.
[31] SEGURA C, SEGREDO E, GONZÁLEZ Y, et al. Multiobjectivisation of the antenna positioning problem [C] // International Symposium on Distributed Computing and Artificial Intelligence. 2011: 319-327.
[32] JENSEN M T. Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation[J]. Journal of Mathematical Modelling and Algorithms, 2004, 3: 323-347.
[33] LOCHTEFELD D F, CIARALLO F W. Multi-objectivization via decomposition: An analysis of helper-objectives and complete decomposition [J]. European Journal of Operational Research, 2015, 243(2): 395-404.
[34] CARRABS F, CORDEAU J F, LAPORTE G. Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading[J]. INFORMS Journal on Computing, 2007, 19(4): 618-632.
[35] ALCALÁ-FDEZ J, SANCHEZ L, GARCIA S, et al. KEEL: a software tool to assess evolutionary algorithms for data mining problems[J]. Soft Computing, 2009, 13: 307-318.
[36] WANG H F, CHEN Y Y. A genetic algorithm for the simultaneous delivery and pickup problems with time window[J]. Computers & Industrial Engineering, 2012, 62(1): 84-95.
[37] LIU S C, TANG K, YAO X. Memetic search for vehicle routing with simultaneous pickup-delivery and time windows[J]. Swarm and Evolutionary Computation, 2021, 66: 100927.
[38] FENG L, ZHOU L, GUPTA A, et al. Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking[J]. IEEE Transactions on Cybernetics, 2019, 51(6): 3171-3184.
[39] TIAN Y, ZHANG T, XIAO J H, et al. A coevolutionary framework for constrained multiobjective optimization problems[J]. IEEE Transactions on Evolutionary Computation, 2020, 25(1): 102-116.
[40] LI J Q, CAI J C, SUN T, et al. Multitask-based evolutionary optimization for vehicle routing problems in autonomous transportation[J]. IEEE Transactions on Automation Science and Engineering, 2023, 21(3): 2400-2411.

CAI Junchuang, born in 1995, postgraduate. His main research interests include intelligent optimization algorithms and their applications in the field of logistics.

MING Zhong, born in 1967, Ph.D, professor, is a senior member of CCF(No. 05569S). His main research interests include software engineering and artificial intelligence.

(责任编辑:柯颖)