基于双向 Dijkstra 算法的无人机配送路径的研究

卢 婷[1] 何 俊[1] 张 敏[1]
LU Ting HE Jun ZHANG Min

摘 要 无人机配送凭借高效、灵活等优势，受到各行业的广泛青睐与关注。在此背景下，开展无人机配送路径规划研究具有重要的现实意义。文章针对无人机配送路径的规划提出了一种双向 Dijkstra 算法，该算法在进行搜索的过程中，同时从正向和反向同步进行搜索，当两个方向搜索相遇时，即获得了配送路径的最优路径。此外，也对提出的双向 Dijkstra 算法进行了算法的复杂度分析。实验结果表明了该算法的可行性和有效性。同时，借助 MATLAB 软件实现了网络示意图以及最短路径的可视化。

关键词 Dijkstra 算法；无人机配送；路径优化；双向 Dijkstra 算法

doi：10.3969/j.issn.1672-9528.2025.07.010

0 引言

随着社会经济的快速发展，机动车保有量持续攀升，城市道路车流量显著增加，尤其在早晚高峰时段，交通拥堵问题日益严峻。这不仅延长了物流配送的运输时间，还增加了配送成本与管理难度，对物流行业的高效运作形成挑战。无人机配送随之兴起，因其配送效率高效，受交通影响较小等优势，不仅可以应用于物流配送领域，还可以拓展到其他领域，如应急救援[1]、灾害救援、影视拍摄等。在确定好配送目的地之后，需要高效快速地规划出最优配送路径，因此研究无人机配送路径的规划方法具有重要意义。

近年来，诸多学者对无人机配送路径的规划展开了一系列的研究。目前对于路径规划常用的算法主要包括基于图搜索的算法（Dijkstra 算法、A* 算法[2-4] 等），基于采样的算法（RTT 算法、PRM 算法[5] 等）和智能算法（遗传算法[6-8]、蚁群算法、神经网络等[9-11]）。

传统的 Dijkstra 算法可以处理较为简单的小规模图，但遇到复杂的地理环境或者大规模图时，可能会导致路径规划效率较低，因此，本文提出了一种在稀疏图和大规模图上更具优势的双向 Dijkstra 算法。该算法通过正反双向搜索，当搜索点相遇，则可合并正反向搜索路径，进而获得最短路径，最后通过算例展示了算法的可行性和有效性。

1 问题描述与基本定义

在现代物流配送场景里，无人机配送较少受道路网络的交通运行态势空间和时间的限制，相较于车辆的物资运输，无人机物流配送因其高效、灵活更受瞩目。在无人机配送场景中，存在配送中心和配送目标地点以及中间经过的地点，这些地点在地理空间中是相互离散分布的，且已知各地点间的距离信息（以地理坐标经纬度换算得出的实际飞行距离或基于地图数据得到的路径长度），各地点之间由于地形、建筑物分布、空域管制等因素，无人机在不同地点间飞行所面临的成本（如某些区域的禁飞限制、不同时间段的风速风向对飞行速度和能耗的影响）存在差异，这些成本可以转化为对应的权重来表示地点间的路径代价。为提高配送效率并减少能源消耗，需要为无人机规划从配送中心到配送目标地点最优的飞行路径。

该问题可以建模为一个图论问题，其中无人机配送的区域可以被表示为一个加权图，如图 1 所示，正方形节点表示配送中心，星形节点表示配送目标地点以及中间点（如路口或充电站），边则表示不同地点之间的直接飞行路径，每条边的权重为地点间的路径代价。

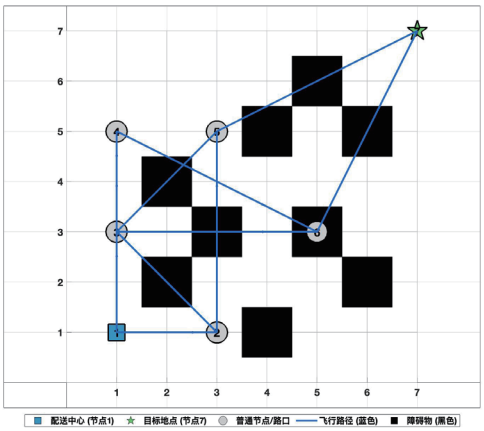

图 1 实际配送区域网络示意图

1. 郑州升达经贸管理学院数学与信息科学学院 河南郑州 451191
[基金项目] 基于应用型人才培养的运筹学教学模式改革与创新（SDJG-2023-ZD08）

图的构建：将无人机的配送区域抽象为一个加权无向图 $G=(V, E)$，其中 V 代表顶点集合，包含配送中心、配送目标地点及中间点；E 代表边集合，每条边连接两个顶点，并有一个非负权重，表示无人机在这两个地点之间飞行的成本（如距离、时间或能量消耗）。

起点和终点：配送中心作为图中的起始点 S，配送目标地点作为终点 T。

路径优化目标：从起点 S 出发，在所有到达终点的所有路径里找到一条到达终点 T 的最短路径。这里的“最短”是指根据边的权重（如最小距离、成本影响、最短时间或最低能量消耗）来定义的。

2 无人机路径规划

2.1 双向 Dijkstra 算法

设无向网络图 $G=(V, E)$，图中各边 (v_i, v_j) 有权 w_{ij}（若 $w_{ij}=\infty$ 表示 v_i 与 v_j 不相邻），v_1 为起始点，v_t 为终点，网络中有多条从 $v_1 \rightarrow v_t$ 的路径，对路径优化即找一条从起始点 $v_1 \rightarrow v_t$ 的最短路径即使从 $v_1 \rightarrow v_t$ 的所有路中总权之和最小的那条路径。

双向 Dijkstra 算法的具体过程为：

（1）初始化

初始化两个距离数组，一个用于记录从起点到标号点的最短距离（正向）记为 u_j^{start}，一个用于终点到标号点的最短距离（反向）记为 u_j^{end}。起始点和终点到自身的距离设 0，到其他点的距离设为无穷。正向搜索已标号点集合记为 S^{start} 和反向搜索已标号点集合为 S^{end}，正向搜索已标号点集合记为 $\overline{S}^{\text{start}}$ 和反向搜索已标号点集合为 $\overline{S}^{\text{end}}$。pred($j$) 记录正向搜索中从起始点 v_1 到 v_j 最短路径当前要标记的点 v_j 的前驱点标号，通过标记，从后向前进行反向的追踪，从而获得 v_1 到 v_j 最短路径。succ(j) 记录反向搜索中从起始点 v_t 到 v_j 最短路径当前要标记的点 v_j 的后继点标号，通过标记，从前向后进行正向的追踪，从而获得 v_j 到 v_t 最短路径。令初始迭代次数 $k=0$。

（2）正向搜索

从起点开始，给起点 v_1 标号 $(0, s)$，pred(1)=s，$\overline{S}^{\text{start}}$ ={除去起始点的所有顶点}，将起始点加入已标号点集合 $S^{\text{start}}=\{v_1\}$。求出弧集 $A=\{(v_i, v_j)|v_i \in S^{\text{start}}, v_j \in \overline{S}^{\text{start}}\}$，若 $A^{\text{start}}=\varnothing$，表明不存在不标号的另一个顶点，则计算结束。对弧集上的每一条弧计算 $\min\limits_j \{u_j + w_{ij} | (v_i, v_j) \in A^{\text{start}}\} = u_j + w_{ij^*} = u_{j^*}$（其中 v_{j^*} 为从 v_i 到 v_j 的所有路径中距离最小的节点记为 v_{j^*}），选出距离最小的节点 v_{j^*}，则新的标号点为 v_{j^*}，标号为 $(v_{j^*}, \text{pred}(j^*))$，更新已标号和未标号点集合 $S^{\text{start}}=\{v_1, v_{j^*}\}$，$\overline{S}^{\text{start}}$ ={除去起始点的所有顶点}/$\{v_{j^*}\}$。

（3）反向搜索

从终点开始，$\overline{S}^{\text{end}}$ ={除去终点的所有顶点}，将终点加入已标号点集合 $S^{\text{end}}=\{v_t\}$。求出弧集 $A=\{(v_i, v_j)|v_i \in S^{\text{end}}, v_j \in \overline{S}^{\text{end}}\}$，若 $A^{\text{end}}=\varnothing$，表明不存在不标号的另一个顶点，则计算结束。对弧集上的每一条弧计算 $\min\limits_j \{l_j + w_{ij} | (v_i, v_j) \in A^{\text{end}}\} = l_j + w_{ij^*} = l_{j^*}$，选出距离最小的节点 v_{j^*}，则新的标号点为 v_{j^*}，标号为 $(l_{j^*}, \text{succ}(j^*))$，更新已标号和未标号点集合 $S^{\text{end}}=\{v_t, v_{j^*}\}$，$\overline{S}^{\text{end}}$ ={除去终点的所有顶点}/$\{v_{j^*}\}$。

（4）终止判断

若正向搜索已标号点集合 S^{start} 与反向搜索已标号点集合 S^{end} 存在共同标号的顶点，则停止搜索，通过交集点以及前趋点和后继点，正向和反向的路径合并，得到从起点到的完整的最短路径；否则继续（2）（3），$k=k+1$。

2.2 双向 Dijkstra 算法的复杂度分析

在无向网络图中，设 V 表示网络图中节点的个数，E 表示网络图中边的数量，对于单向的 Dijkstra 算法的复杂度为 $O((V+E)\log V)$，$\log V$ 是从未标号的顶点中选择到起始点最短距离的顶点操作的时间复杂度。根据上述双向 Dijkstra 算法过程可知，同时从起点和终点进行搜索，可减少搜索空间，从而降低复杂度。

（1）双向 Dijkstra 算法的时间复杂度

正向搜索和反向搜索与单向 Dijkstra 算法类似，所以可知正向搜索的时间复杂度为 $O((V+E)\log V)$，反向搜索的时间复杂度为 $O((V+E)\log V)$，故双向 Dijkstra 算法的总时间复杂度为 $O((V+E)\log V)$。

由于双向搜索的特性，对于正向和反向并行执行，搜索空间在实际应用中通常会减少，因此实际运行时间会比单向的 Dijkstra 算法的运行时间更短。

（2）双向 Dijkstra 算法的空间复杂度

双向 Dijkstra 算法在搜索过程中空间复杂度与单向 Dijkstra 算法是一样的。对于正向搜索，需要存储每个节点的最短距离和前驱节点，空间复杂度为 $O(V)$，对于反向搜索，需要存储每个节点的最短距离和后继点，空间复杂度为 $O(V)$，故双向 Dijkstra 算法的总空间复杂度为 $O(V)$。

（3）单向 Dijkstra 算法和双向 Dijkstra 算法的对比

从对单向 Dijkstra 算法和双向 Dijkstra 算法的分析可知，双向 Dijkstra（并行）的实际运行时间最短，尤其是在稀疏图 $(E \ll V^2)$ 和大规模图上，如表 1 所示。

表 1 单向 Dijkstra 算法和双向 Dijkstra 算法对比

算法	时间复杂度	空间复杂度	执行方式	实际运行时间
单向 Dijkstra	$O((V+E)\log V)$	$O(V)$	正向搜索	$2T$
双向 Dijkstra	$O((V+E)\log V)$	$O(V)$	正向和反向搜索交替进行	T

3 算例分析

例 1：某一无人机配送区域的网络示意图，如图 2 所示，

在赋权的无向图 G 中指定配送中心为发点 v_1，配送目标地点为收点 v_7，其余点称为中间点（即路口），并将 G 中的每一条弧的赋权数 c_{ij} 为弧 (v_i, v_j) 的容量，为此可建立赋权无向图。当各节点之间的连接信息时，使用 MATLAB 软件，绘制出相应的网络示意图。

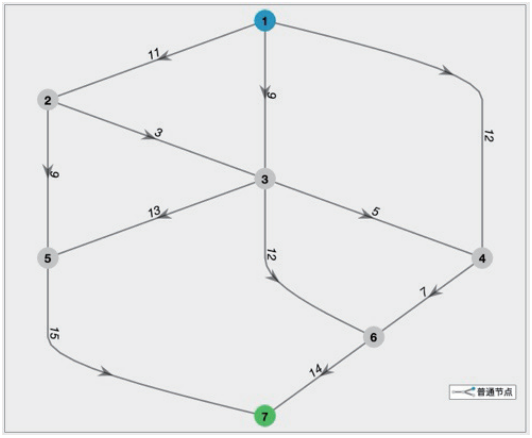

图 2 无人机配送区域网络示意图

解：给初始点和终点标号 $v_1(0, s)$、$v_7(0, e)$，起点到标号点的最短距离 u_1=0，终点到标号点的最短距离 l_7=0。

第 1 次迭代，正向搜索：已标号点 $\{v_1\}$，弧 A: u_1+w_{12}=11、u_1+w_{13}=9、u_1+w_{14}=12，最小弧：$v_1 \rightarrow v_3$，故下次标号点：v_3，前趋点 v_1，u_3=9；反向搜索：已标号点 $\{v_7\}$，弧 A: l_7+w_{57}=15、l_7+w_{67}=14，最小弧：$v_6 \rightarrow v_7$，故下次标号点：v_6，后继点 v_7，l_6=14。

第 2 次迭代，正向搜索：已标号点 $\{v_1, v_3, v_2\}$，弧 A: u_1+w_{12}=11、u_1+w_{14}=12、u_3+w_{34}=14、u_3+w_{35}=22、u_3+w_{36}=21，最小弧：$v_1 \rightarrow v_2$，故下次标号点：v_2，前趋点 v_1，u_2=11；反向搜索：已标号点 $\{v_7, v_6\}$，弧 A: l_7+w_{57}=15、l_6+w_{36}=26、l_6+w_{46}=21，最小弧：$v_5 \rightarrow v_7$，故下次标号点：v_5，后继点 v_7，l_5=15。

第 3 次迭代，正向搜索：已标号点 $\{v_1, v_3\}$，弧 A: u_1+w_{14}=12、u_3+w_{34}=14、u_3+w_{35}=22、u_3+w_{36}=21、u_2+w_{23}=14、u_2+w_{25}=20，最小弧：$v_1 \rightarrow v_4$，故下次标号点：v_4，前趋点 v_1，u_4=12；反向搜索：已标号点 $\{v_7, v_6, v_5\}$，弧 A: l_6+w_{36}=26、l_6+w_{46}=21，最小弧：$v_4 \rightarrow v_6$，故下次标号点：v_4，后继点 v_6，l_4=21。

正向搜索：已标号点 $\{v_1, v_3, v_2, v_4\}$，反向搜索：已标号点 $\{v_7, v_6, v_5, v_4\}$，正向与反向搜索相遇存在交集点 v_4，故停止搜索。

根据正向与反向确定最短路径：$v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7$，最短距离：$u_4+l_4$=33。

对于上述算例，可以通过 MATLAB 编程轻松实现计算，如表 2 所示。

表 2 双向 Dijkstra 算法 MATLAB 部分程序代码

```
% 正向搜索初始化
G% 网络示意图转化为邻接矩阵
 dist_start=inf(1,n); % 将各个节点到起始点的初始距离为无穷大
dist_start(startNode) = 0;  % 起始节点到自身的距离为 0
prev_start = zeros(1, n); % 前趋节点初始为 0
v_start=[];% 已标号点集合
   unv_start = 1:n; % 未标号节点集合
   % 正向搜索探索过程
   [~, idx_start]=min(dist_start(unv_start));
   v_start=unv_start(idx_start);
  unv_start(idx_start)=[];
   for i=1:n
 if G(v_start, i) ~= inf
   alt_start=dist_start(v_start)+G(v_start, i);
  if alt_start < dist_start(i)
  dist_start(i)=alt_start;
        prev_start(i)=v_start;
 end
   end
   end
```

反向搜索程序代码同上，通过运行可以得到计算结果，如图 3 所示。通过 MATLAB 软件绘制出最短路径，如图 4 所示。

```
>> bi_dijkstra
迭代次数:
     4

最短路径距离:
    33

节点:
     1     2     3     4     5     6     7

前趋节点:
     0     1     1     1     2     4     6

程序运行时间: 0.011069 秒
```

图 3 例 1 的 MATLAB 运行结果

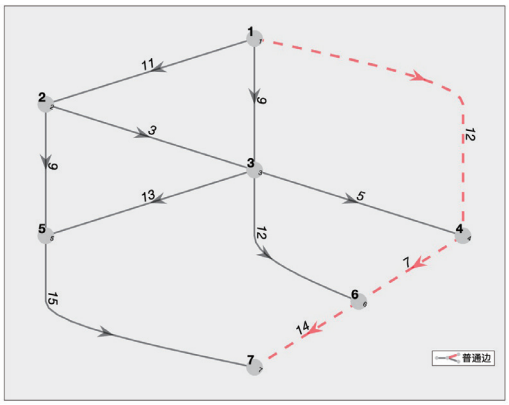

图 4 例 1 的最短路径图

（下转第 52 页）

[3] CANNY J. A computational approach to edge detection[J]. IEEE transactions on pattern analysis and machine intelligence, 1986,8(6): 679-698.

[4] FUJITA Y, SHIMADA K, ICHIHARA M, et al. A method based on machine learning using hand-crafted features for crack detection from asphalt pavement surface images[C]// Proceedings of SPIE - The International Society for Optical Engineering. USA: SPIE, 2017: 117-124.

[5] RONNEBERGER O, FISCHER P, BROX T. U-Net: convolutional networks for biomedical image segmentation[EB/OL].(2015-05-18)[2025-01-26].https://doi.org/10.48550/arXiv.1505.04597.

[6] HAMISHEBAHAR Y, GUAN H, STEPHEN S, et al. A comprehensive review of deep learning-based crack detection approaches[J]. Applied sciences, 2022, 12(3): 1374.

[7] GUO F, LIU J, LÜ C S, et al. A novel transformer-based network with attention mechanism for automatic pavement crack detection[J]. Construction and building materials, 2023, 391: 131852.

[8] HU J, SHEN L, SUN G.Squeeze-and-excitation networks[C/OL]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.Piscataway:IEEE,2018[2025-02-19]. https://ieeexplore.ieee.org/document/8578843.DOI:10.1109/CVPR.2018.00745.

[9] LIU Z H. Road crack detection system using image segmentation algorithm[C]//PCCNT '23: Proceedings of the 2023 International Conference on Power, Communication, Computing and Networking Technologies.NewYork: ACM, 2023: 1-6.

[10]SABOURI M, SEPIDBAR A. SUT-Crack: a comprehensive dataset for pavement crack detection across all methods[J]. Data in brief, 2023, 51:109642.

【作者简介】

郭晓彤（2001—），女，河北邯郸人，硕士研究生，研究方向：灾害信息处理技术。

王跃宝（1999—），女，河北石家庄人，硕士研究生，研究方向：灾害信息处理技术。

鲁王泽（2001—），男，安徽铜陵人，硕士研究生，研究方向：灾害信息处理技术。

（收稿日期：2025-03-13 修回日期：2025-07-07）

（上接第 47 页）

4 结论

本文提出了一种基于双向 Dijkstra 算法的无人机路径规划，通过对路径进行正向与反向的同步搜索，当两个方向的搜索相遇时，合并获得最短路径。后对提出的双向 Dijkstra 算法进行时间、空间的复杂度分析，以及对算法的实际运行时间进行了分析对比。通过分析可知，双向 Dijkstra 算法在实际运行中是高效的，特别是在稀疏图 ($E \ll V^2$) 和大规模图上。最后，通过一个算例验证了该算法的可行性和高效性。

参考文献：

[1] 王其 , 王磊 , 倪世松 , 等 . 基于改进 A* 算法的应急救援无人机路径规划 [J]. 计算机仿真 , 2024, 41(6):84-88.

[2] 唐嘉宁 , 彭志祥 , 李孟霜 , 等 . 基于改进 A* 算法的无人机路径规划研究 [J]. 电子测量技术 , 2023, 46(8):99-104.

[3] 高九州 , 张焯 . 基于改进 A* 算法的无人机三维空间避障路径规划 [J]. 计算机测量与控制 ,2023,31(12):203-209.

[4] 罗超 . 基于改进 A* 算法的无人机路径规划算法及应用 [D]. 长沙 : 中南大学 ,2023.

[5] 程谦 , 高嵩 , 曹凯 , 等 . 基于 PRM 优化算法的移动机器人路径规划 [J]. 计算机应用与软件 , 2020,37(12):254-259.

[6] 周金亮 , 胡涛 . 基于改进遗传算法的无人机路径规划研究 [J]. 机电技术 , 2024(2):13-16.

[7] 黄书召 , 田军委 , 乔路 , 等 . 基于改进遗传算法的无人机路径规划 [J]. 计算机应用 , 2021, 41(2):390-397.

[8] 杨晨 , 肖博文 , 刘郡怡 , 等 . 基于改进遗传算法的无人机路径规划方法 :202311512310[P].2024-12-17.

[9] 郭一凡 . 基于深度强化学习的四旋翼无人机路径规划研究 [D]. 西安 : 西安工业大学 ,2024.

[10] 骆文冠 , 于小兵 . 基于强化学习布谷鸟搜索算法的应急无人机路径规划 [J]. 灾害学 , 2023, 38(2):206-212.

[11] 王兴旺 , 张清杨 , 姜守勇 , 等 . 基于改进鲸鱼优化算法的动态无人机路径规划 [J]. 计算机应用 ,2025,45(3):928-936.

【作者简介】

卢婷 (1993—)，女，河南周口人，硕士，助教，研究方向：最优化理论算法及应用，email: luting19@yeah.net。

（收稿日期：2025-03-06 修回日期：2025-07-02）