

Research on Dynamic Road Net Nodes Based Dijkstra Algorithm for Path Planning

Lie Guo[†], Shujun Sun, Zejian Ren, Bing Li

School of Automotive Engineering, Dalian University of Technology, Dalian Liaoning 116024, China

[†]Email: guo_lie@dlut.edu.cn

Abstract

The time complexity of traditional Dijkstra algorithm is quadratic times of the number of road net nodes, which results in its low efficiency for path planning. Existing improved Dijkstra algorithms are orientated to static road net nodes and thus influence their path planning speeds. This paper presented a dynamic road network nodes based Dijkstra algorithm. The road net was simplified by screening out the corner nodes, cross nodes, T cross nodes as well as those nodes that more than four paths going through of them. Such nodes of other types were abandoned. The number of road nodes of the optimized road net was decreased which is helpful to improve the path planning speed. Experiment results indicate that target nodes of a road net can be added and removed during the path planning process. When planning the same road net, the time consumption of traditional Dijkstra algorithm increases with the number of the task nodes, while that of the proposed Dijkstra algorithm stays the same even though the number of task nodes is sharply increased. The feasibility and effectiveness of the proposed Dijkstra algorithm was validated.

Keywords: Path Planning; Dijkstra Algorithm; Dynamic Road Network Nodes; Shortest Path

基于动态路网节点的 Dijkstra 算法路径规划研究^{*}

郭烈, 孙淑军, 任泽建, 李兵

大连理工大学 汽车工程学院, 辽宁 大连 116024

摘 要: 传统 Dijkstra 算法随着路网节点数目的增多, 其时间复杂度成节点数目平方级增加, 现有的 Dijkstra 改进算法无法动态改变原路网节点数目从而影响算法的计算效率。本文提出一种能够动态增加和删除路网节点的 Dijkstra 算法, 提取原始路网模型中的边界角点、十字交叉节点、T 型交叉节点以及有其他四条以上路段通过的节点, 剔除这些节点之间构成的路段上的其他任务点得到一个新的优化路网模型, 保留有原始路网模型的骨架结构的同时简化了路网结构, 从而使时间复杂度降低, 提高其运行效率。仿真表明, 改进算法能够实现动态增删路网节点, 在同一路网环境中, 随着路网节点数目增多, 传统算法耗时越来越大, 而改进算法基本维持不变, 验证了改进算法的可行性和有效性。

关键词: 路径规划; Dijkstra 算法; 动态路网节点; 最短路径

引言

路径规划是智能交通运输系统的重要组成部分, 有利于实现车辆的准确定位与导航、交通流控制等功能。在这方面, 国内外研究人员开发了多种算法应用于路径规划中, 如 Dijkstra 算法、Floyd 算法、A*算法经典等^[1]。其中, Dijkstra 算法是比较经典的最优路径规划算法。但是随着路网复杂度增加, 该算法的时间复杂度将以节点数目的平方级增加, 导致算法的运行实时性和效率降低^[2]。为此, 需要对该问题进行改进, 如文 Huang 等^[3]采用权值约束来解决数据结构容量问题以降低储存空间和计算时间; 李擎等^[4]针对当前不相连节点间路径长度为无穷大这一特点, 对 Dijkstra 算法进行了改进, 通过判断两个节点是否相连, 若发现两个节点并不

^{*}基金资助: 国家自然科学基金资助项目 (61104165); 中央高校基本科研业务费专项资金资助项目 (DUT13JS02)。

相连时,则舍去相应计算,从而减少大量的无关节点计算,提高运算的效率。Deng 等^[5]采用模糊数分级平均表示法来对 Dijkstra 算法进行改进。但是这些算法只能解决已知固定路网中,从一个已经存在的节点到另一个已经存在的节点的最优路径规划问题,并没有考虑路网结构对路径规划算法的影响。如文献[6]设计的路网结构中,将所有的任务点都作为路网的节点显示在路网中,但随着任务点的增多,算法时间复杂度将越来越大。本文提出一种基于动态路网节点的 Dijkstra 改进算法,能够根据实际环境提取重要的任务点作为路网节点并保存其他任务节点,得到优化的路网模型。然后根据不同任务需求,动态为路网添加其他任务节点,并能实时地根据新任务节点需求及时清除旧的任务节点,从而简化路网模型的复杂度,提高路径规划效率。

1 Dijkstra 算法基本原理

单源最短路径问题,即求某一路网模型中给定节点到其它任一节点的最短路径的问题,求解最短路径是一个经典的网络优化问题^[7]。最短路径最优子结构的性质描述为:

如果 $P(v_i, v_j) = \{v_i, \dots, v_m, \dots, v_n, \dots, v_j\}$ 是一路网模型中从节点 v_i 到节点 v_j 的最短路径, v_m 和 v_n 是这条路径上的两个中间节点,那么 $P(v_m, v_n)$ 必定是从节点 v_m 到节点 v_n 的最短路径。Dijkstra 算法的基本思想是从 v_i 出发,逐步地向外探寻最短路。执行过程中,与每个点对应,记录下一个数,它或者表示从 v_i 到该点的最短路径的权(称为 P 标号)、或者是从 v_i 到该点的最短路径的权的上界(称为 T 标号),方法的每一步是去修改 T 标号,并且把某一个具 T 标号的点改变为 P 标号的点,从而使 D 中具 P 标号的顶点数多一个,这样,至多经过 $p-1$ 步,就可以求出从 v_i 到各点最短路径。对证明如下:

假设 $P(v_m, v_n)$ 不是从节点 v_m 到节点 v_n 的最短路径,那么必定存在另一条从 v_m 到 v_n 的路径 $P'(v_m, v_n)$, 使得 $P'(v_m, v_n) < P(v_m, v_n)$, 则有:

$$P'(v_i, v_j) = P(v_i, v_m) + P'(v_m, v_n) + P(v_n, v_j) < P(v_i, v_j) \quad (1)$$

这与 $P(v_i, v_j)$ 是从 v_i 到 v_j 的最短路径相矛盾,因此该性质得证。Dijkstra 算法根据这个性质规划由一个节点 v_i 到另一个节点 v_j 的最短路径,其表达式为:

$$D[v_i, v_j] = \min\{L[v_i, v_j], L[v_i, v_k] + W_{kj}\} \quad (2)$$

其中, $D[v_i, v_j]$ 为从节点 v_i 到节点 v_j 的最短距离, $L[v_i, v_j]$ 是由节点 v_i 直达节点 v_j 的距离值, W_{kj} 为节点 v_k 与节点 v_j 间有向弧的权值。

假设某一路网模型为 $G = (V, E, W)$, 其中 $V = (v_0, v_1, \dots, v_n)$ 为路网的节点集, E 为路网的有向弧段集, W 为有向弧段的权值集。路网中某一任务起始点为 v_l , 终止点为 v_m , 令

$$U_1 = \{v_l\}, \quad U_2 = \{v_0, v_1, \dots, v_{l-1}, v_{l+1}, \dots, v_n\} \quad (3)$$

其运行步骤如下:

Step 1: 从 U_2 选择使 $D[v_l, v_j]$ 最小的节点 v_i , 并将 v_i 移入到 U_1 中, 这样 $U_1 = \{v_l, v_i\}$, $U_2 = \{v_0, v_1, \dots, v_{l-1}, v_{l+1}, \dots, v_{i-1}, \dots, v_{i+1}, \dots, v_n\}$;

Step 2: 更新与 v_i 直接连接的节点间的最短路径值, 从 U_2 选择使 $D[v_l, v_j]$ 最小的节点 v_j , 并将 v_j 放入到 U_1 中。

Step 3: 重复步骤 2, 直到找到节点 v_m , 并将其放入 U_1 , 这样 $U_1 = \{v_l, v_i, v_j, \dots, v_m\}$ 即为所求的最短路径。

Dijkstra 算法每次循环都要重复判断当前节点与其他节点的权值信息, 对路径节点数目 n 进行了两重循环, 因此, 其时间复杂度为 $O(n^2)$ 。显然, 随着路网节点数目的增多, 路径规划时间复杂度将以其节点数目的平方级数目增加^[8]。

2 对 Dijkstra 算法的改进

2.1 路网简化

由上述分析可以看出 Dijkstra 算法在计算大型路网时, 其时间复杂度将以节点数目的平方级即 $O(n^2)$ 增

加，这将导致算法实时性变差[9]。目前文献所介绍的优化方法通常只对算法本身进行了一定的优化，但是这些优化的前提是原路网已经存在且固定不变，即将所有的任务点都加入到路网 $G=(V,E,W)$ 中，然后进行 Dijkstra 算法规划。而在实际中，如果将所有的任务点加入到路网中，即使使用改进的 Dijkstra 算法，也无法避免算法时间复杂度随着节点增加而增加的问题。

为此，本文提出了一种 Dijkstra 改进算法，它能够动态地增加和删除一部分节点，实时保持路网的简洁性，而且在路网所处的环境一定时，改进算法的时间复杂度只与优化后路网的节点数相关，而与原复杂路网的节点数无关。比如，有一路网，在它的一个交叉口与另一个交叉口之间存在着 n 个任务点。路径规划中，目标车辆的起始点或终止点可以是这 n 个点中的任意一个，按照传统改进算法，需要将这 n 个节点全部加入路网中，然后进行路径规划，单看这个 n 节点，算法的时间复杂度为 $O(n^2)$ ；而本文的改进算法则只需要考虑两个交叉口的坐标点信息，并根据任务需求，动态地将需要的节点(起始点或终止点)加入到路网，因此时间复杂度为 $O((2+i)^2), (0 \leq i \leq 2)$ ，与原路网节点总数 n 无关，可见本算法在 n 取值比较大时，其时间复杂度将远小于传统算法。

本文对路网的优化原则如下：提取原始路网 $G=(V,E,W)$ 中的边界角点、十字交叉节点、T 型交叉节点以及有其他四条以上路段通过的节点，剔除这些节点之间构成的路段上的其他任务点，即保留每一条道路上的端点节点，剔除非端点节点。这样就得到一个新的优化路网。这段路网保留有原路网的骨架结构，同时简化了路网结构。定义原路网节点的数据结构如表 1 所示，根据路网节点类型 P_Style 就可以提取需要的路网节点，组成一个新的优化路网 $G'=(V',E',W')$ 。

表 1 原路网节点的数据结构

符号	<i>Point_ID</i>	<i>x_Pos</i>	<i>y_Pos</i>	<i>P_Style</i>	<i>c_Pi</i>
说明	节点编号	节点横坐标	节点纵坐标	节点类型	与当前节点相连接的节点

假设有如图 1 所示的原始路网模型 $G=(V,E,W)$ ，图中圆圈表示任务节点，两圆圈间的线段表示两节点相互连通，线段上的数值表示两节点之间的距离。对模型进行简化时，通过提取有三条以上路段通过的节点，并将它们重新进行连接，从而得到如图 2 所示的优化路网模型 $G'=(V',E',W')$ 。图 3 是根据任务需求增加两个任务点 B、N 后的新路网。

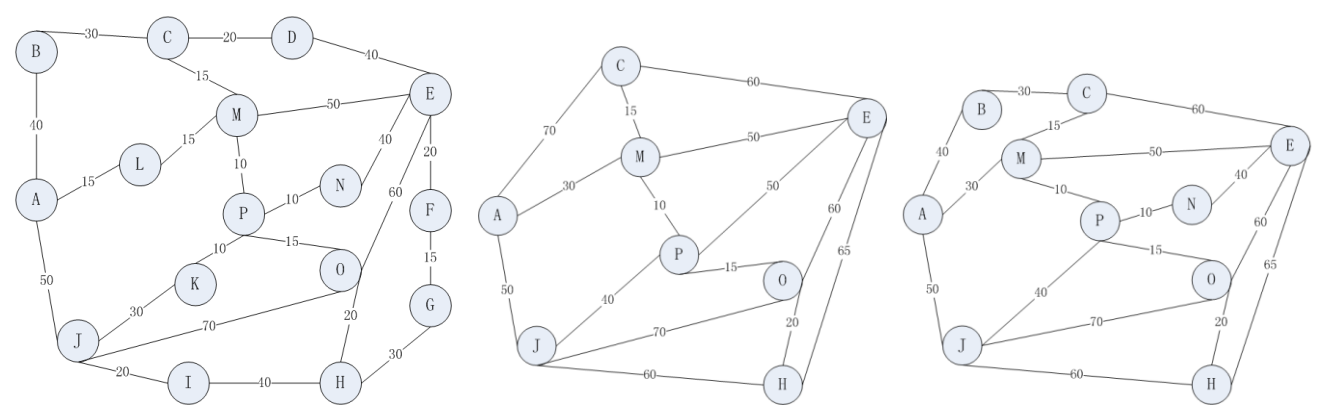


图 1 原始路网模型

图 2 简化路网模型

图 3 增加新任务点后的新路网模型

2.2 改进算法的实现

得到优化路网 $G'=(V',E',W')$ 以后， V' 与 V 相比，其中的许多元素消失了，但规划任务时，这些元素又是必须的，因此，需要在算法中实时地更新这些任务点。如果规划从节点 v_i 到节点 v_j 的任务时，需要分以下几种情况对两节点进行讨论：

1. 如果 $v_i \notin V', v_j \notin V'$, 则需要将 v_i 和 v_j 添加到 V' 中, 这样 $V' = \{v'_0, v'_1, \dots, v'_n, v_i, v_j\}$;
2. 如果 $v_i \notin V', v_j \in V'$, 则只需将 v_i 添加到 V' , $V' = \{v'_0, v'_1, \dots, v'_n, v_i\}$;
3. 如果 $v_i \in V', v_j \notin V'$, 则只需将 v_j 添加到 V' , $V' = \{v'_0, v'_1, \dots, v'_n, v_j\}$;
4. 如果 $v_i \in V', v_j \in V'$, 则不需添加任何点到 V' , $V' = \{v'_0, v'_1, \dots, v'_n\}$ 。

得到更新的路网模型 V' 后, 令 $V' = S' \cup U'$, 其中 S' 为已经求出的以节点 v'_i 为起点的最短路径节点的集合, U' 为待求最短路径节点的集合。这样, $S' = \{v'_i\}$, $U' = \{v'_0, v'_1, \dots, v'_{i-1}, v'_{i+1}, \dots, v'_n\}$ 。

对 $\forall v'_i \in U'$, 求出一点 v'_k 使得两节点间直接相连距离 $D'(v'_i, v'_k)$ 最小; 得到:

$S' = \{v'_i, v'_k\}$, $U' = \{v'_0, v'_1, \dots, v'_{i-1}, v'_{i+1}, \dots, v'_{k-1}, v'_{k+1}, v'_n\}$, 更新 U' 中个节点的距离, 若起点 v'_i 经 v'_k 到 U' 中节点 v'_m 的距离 $D'(v'_i, v'_k, v'_m) < D'(v'_i, v'_m)$, 更新 v'_i , v'_m 的距离为 $D'(v'_i, v'_k, v'_m)$;

重复上述步骤, 直到 $S' = \{v'_i, v'_k, \dots, v'_j\}$, 这样 S' 所包含的节点行径的路径即为所求的最短路径, 该规划任务结束后, 删除 V' 中新增节点, 将其恢复为 $V' = \{v'_0, v'_1, \dots, v'_n\}$ 。

如此循环, 直到所有的任务结束。

3 仿真与分析

3.1 仿真实验

根据上述 Dijkstra 优化算法的运行步骤, 本文运用 VC 开发了路径规划仿真控制软件, 软件运行流程图如图 4 所示。

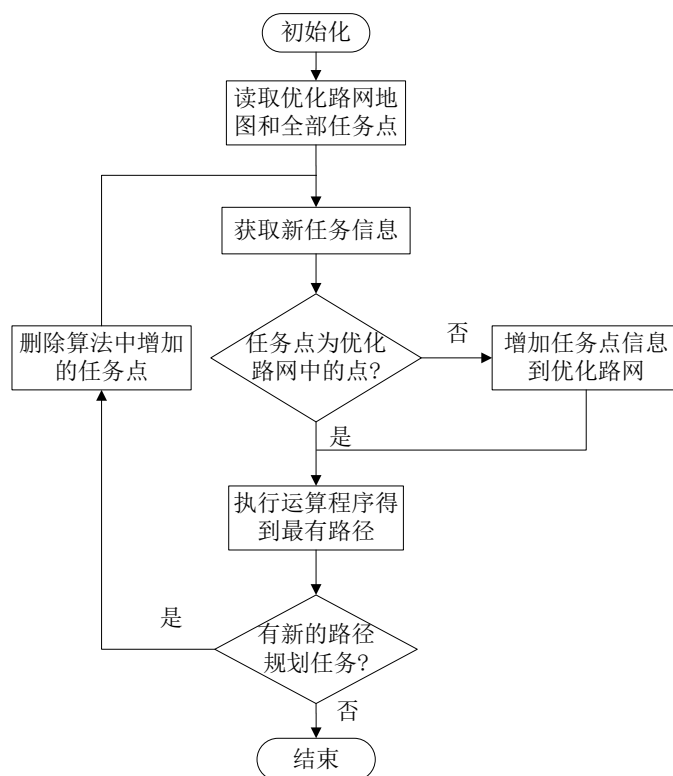


图 4 算法流程图

运行过程中, 程序动态读取优化路网模型 $G' = (V', E', W')$ 的任务点 v'_i, v'_j 的信息, 并保存到 CPtrArray 集合类对象中, 根据获得的任务点信息, 将获取到的路径节点信息存放到对应的集合类对象中。图 5 所示为读取 Excel 地图数据库后获得的所有任务节点的信息图, 图 6 为得到的优化路网信息。

根据图 5 列出的任务点, 在软件的目标选择器中输入所需的起始点, 点击确定既可以得到一条优化后的

路径。如从任务点 16 到任务点 28，首先在目标选择器中输入起始点信息，然后，软件自动将这两个任务加入到优化路网中。将起始点和终止点加入路网后，便可运行软件来获取最短路径。

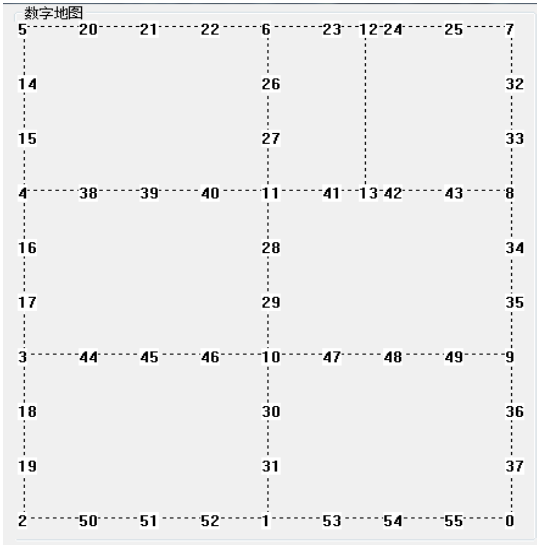


图 5 所有任务点信息图

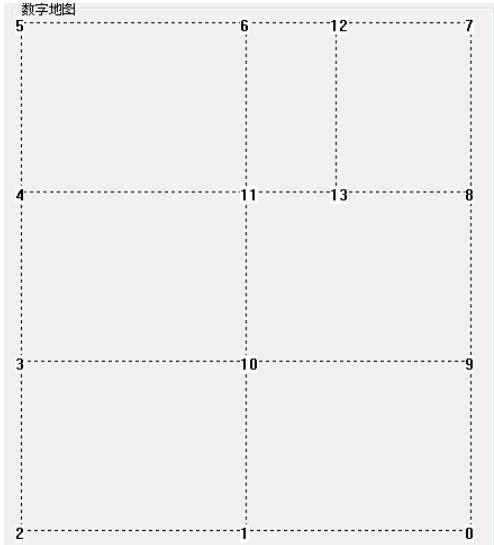


图 6 简化路网信息图

图 7 中绿色实线所示为规划完成的路径图形化信息，图 8 为该路径所经过节点的详细信息和路径的总长度信息。一个任务执行完成以后，剔除该任务增加的路径节点以恢复原优化路网，重新执行新的路径规划任务。

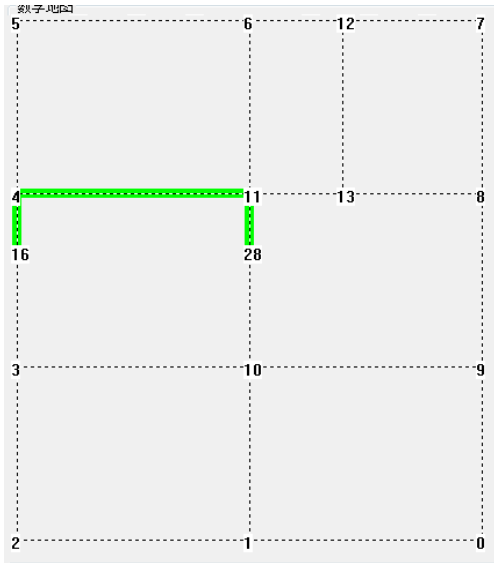


图 7 规划完成后的路线图

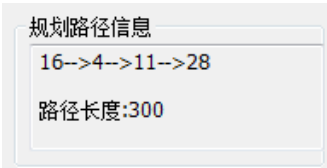


图 8 路径规划结果信息显示

3.2 仿真分析

为了验证改进算法的性能，本文进行了多次对比仿真实验。首先提取出设定的路网中的 14 个主要连通节点，得到优化路网，然后分别向路网中加入不同数目的任务节点，对比改进 Dijkstra 算法与传统 Dijkstra 算法的运算耗时。这里以 50 个、100 个、150 个、200 个、500 个、1000 个任务节点对该路网模型进行了仿真实验，在同一个程序中，每种情况各运行 10 次。统计算法每次运行时，规划路径的耗时，得到图 9 和图 10 的统计数据，其中图 9 中从左到右，从上到下依次为 50 个、100 个、150 个、200 个任务节点的情况，图 10 从左到右依次为 500 个、1000 个任务点的情况。

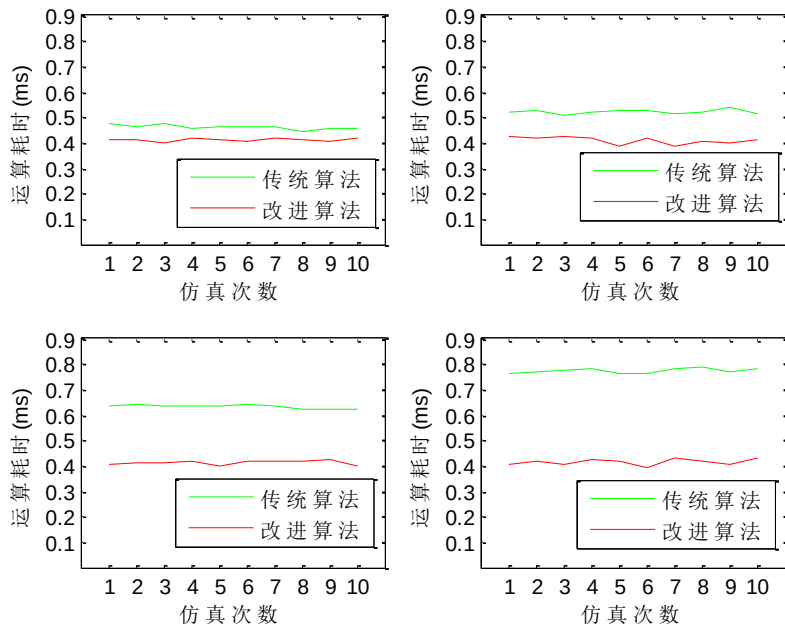


图 9 不同任务节点数下传统 Dijkstra 算法与改进 Dijkstra 算法耗时曲线对比图 1

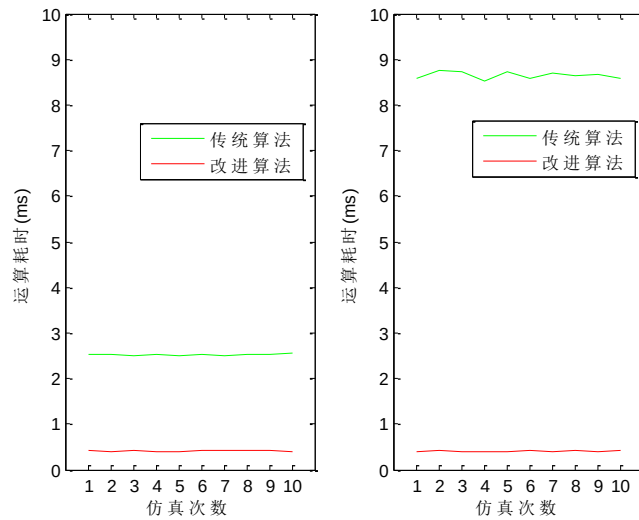


图 10 不同任务节点数下传统 Dijkstra 算法与改进 Dijkstra 算法耗时曲线对比图 2

由图 9、图 10 的数据分别对不同任务节点数下传统算法耗时和改进算法耗时取平均值，得到下表。

表 2 传统 Dijkstra 算法与改进 Dijkstra 算法耗时对比

任务节点数(个)	传统算法耗时(ms)	改进算法耗时(ms)	时间降低了(%)
50	0.4611	0.4106	11.0
100	0.5205	0.4097	21.3
150	0.6324	0.4127	34.7
200	0.7737	0.4148	46.4
500	2.5241	0.4106	83.7
1000	8.6421	0.4136	95.3

根据表中数据绘制不同任务节点下传统算法和改进算法的平均耗时曲线对比如图 11 所示。

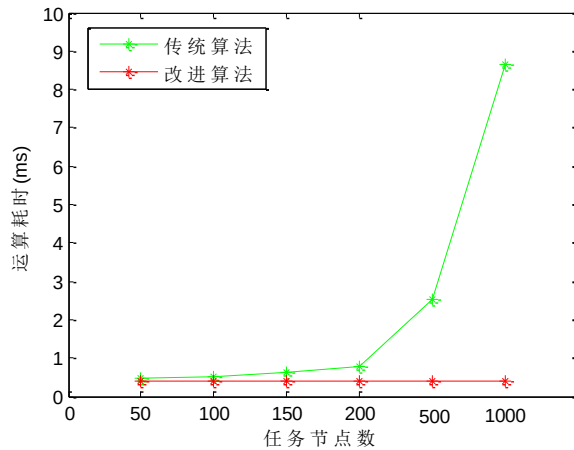


图 11 不同任务节点下传统算法和改进算法平均耗时曲线对比图

上图表明传统 Dijkstra 算法的时间复杂度呈非线性增长趋势, 这是因为同一路网中 $G=(V,E,W)$, 传统 Dijkstra 算法的时间复杂度为 $O(n^2)$ 。其实对于其它没有考虑路网结构的改进算法, 其时间复杂度也将随着 n 的增加而进一步增大, 如文献[2]中改进的 Dijkstra 算法的时间复杂度为 $O(E \lg n)$, E 为结点列表中, 同结点关联的所有弧段数目, 显然当 n 取较大值时, $O(n^2)$ 、 $O(E \lg n)$ 都将随之变大。而本算法空间复杂度 $O((2+I)^2)$ 则与 n 无关, 因而在当前路网中其时间复杂度不变, 且在 $n \gg I$ 时, 本文提出的 Dijkstra 改进算法的时间复杂度要远小于传统算法和其它为对路网进行优化的改进算法。

4 结束语

本文在传统 Dijkstra 算法的基础上提出了一种能够动态地增加和删除路网节点的 Dijkstra 改进算法。改进算法解决了传统算法和传统改进算法只能应用于固定路网规划的问题, 并优化了路网节点数目。理论分析表明, 在同一路网中, 任务节点数目越多, 该改进算法的效率相比传统算法和传统改进算法越高。仿真实验表明, 该算法能够实现动态增删路网节点的功能, 且在同一路网环境中, 随着路网节点数目增多, 传统算法耗时越来越大, 而改进算法基本维持不变, 证明改进算法的可行性和高效性。

REFERENCES

- [1] 葛艳, 王健, 孟友新, 等. 车辆导航动态路径规划的研究进展[J]. 公路交通科技, 2010, 27:113-117.[GE Yan, WANG Jian, MENG Youxin, et al. Research Progress on Dynamic Route Planning of Vehicle Navigation Journal of Highway and Transportation Research and Development[J]. 2010, 27:113-117.]
- [2] 李元臣, 刘维群. 基于 Dijkstra 算法的网络最短路径分析[J]. 微计算机应用, 2004, 25:295-362.[LI Yuancheng, LIU Wei-qun. Analysis of the Shortest Route in Network on Dijkstra Algorithm[J]. Microcomputer Applications, 2004, 25:295-362.]
- [3] Yizhen Huang, Qingming Yi, Min Shi. An Improved Dijkstra Shortest Path Algorithm[C]// Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering, Hangzhou, China, March 2013. Paris: Atlantis Press, 2013: 226-229
- [4] 李擎, 宋顶立, 张双江, 等. 两种改进的最优路径规划算法[J]. 北京科技大学学报, 2005, 27(3) 113-116.[LI Qing, SONG Dingli, ZHANG Shuangjiang, et al. Two improved optimum path planning algorithms [J]. Journal of University of Science and Technology Beijing, 2005, (03) 113-116.]
- [5] Yong Deng, Yuxin Chen, Yajuan Zhang, et al. Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment[J]. Applied Soft Computing, 2012, 12(3):1231-1237.
- [6] 吴晓雨. 物流系统中 AGV 路径规划算法的研究[D]. 长春: 吉林大学, 2004
- [7] 张水舰, 刘学军, 杨洋. 动态随机最短路径算法研究[J]. 物理学报, 2012, 61(16):160201.[Zhang ShuiJian, Liu XueJun, Yang Yang. Dynamic stochastic shortest path algorithm[J]. Acta Phys. Sin., 2012, 61(16):160201.]

- [8] Barbehenn, M. A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices[J]. IEEE Transactions on Computers, 1998, 47(2):263
- [9] D. Cantone, S. Faro. Two-Levels-Greedy: a generalization of Dijkstra's shortest path algorithm[J]. Electronic Notes in Discrete Mathematics, 2004, 17(2004):81-86

【作者简介】



¹郭烈（1978-），男，汉族，博士，副教授，主要从事智能车辆、汽车安全辅助驾驶技术研究，2007年6月获得吉林大学交通学院载运工具运用工程专业博士学位，现为大连理工大学汽车工程学院教师。Email: guo_lie@dlut.edu.cn

²孙淑军（1986-），男，汉族，研究方向为智能车辆，现为大连理工大学汽车工程学院硕士研究生。

Email: suujune@163.com

³任泽建（1989-），男，汉族，研究方向为汽车安全辅助驾驶，现为大连理工大学汽车工程学院硕士研究生。

Email: renzejian2008@163.com

⁴李兵（1989-），男，汉族，研究方向为智能车辆，现为大连理工大学汽车工程学院硕士研究生。

Email: libing19890910@163.com