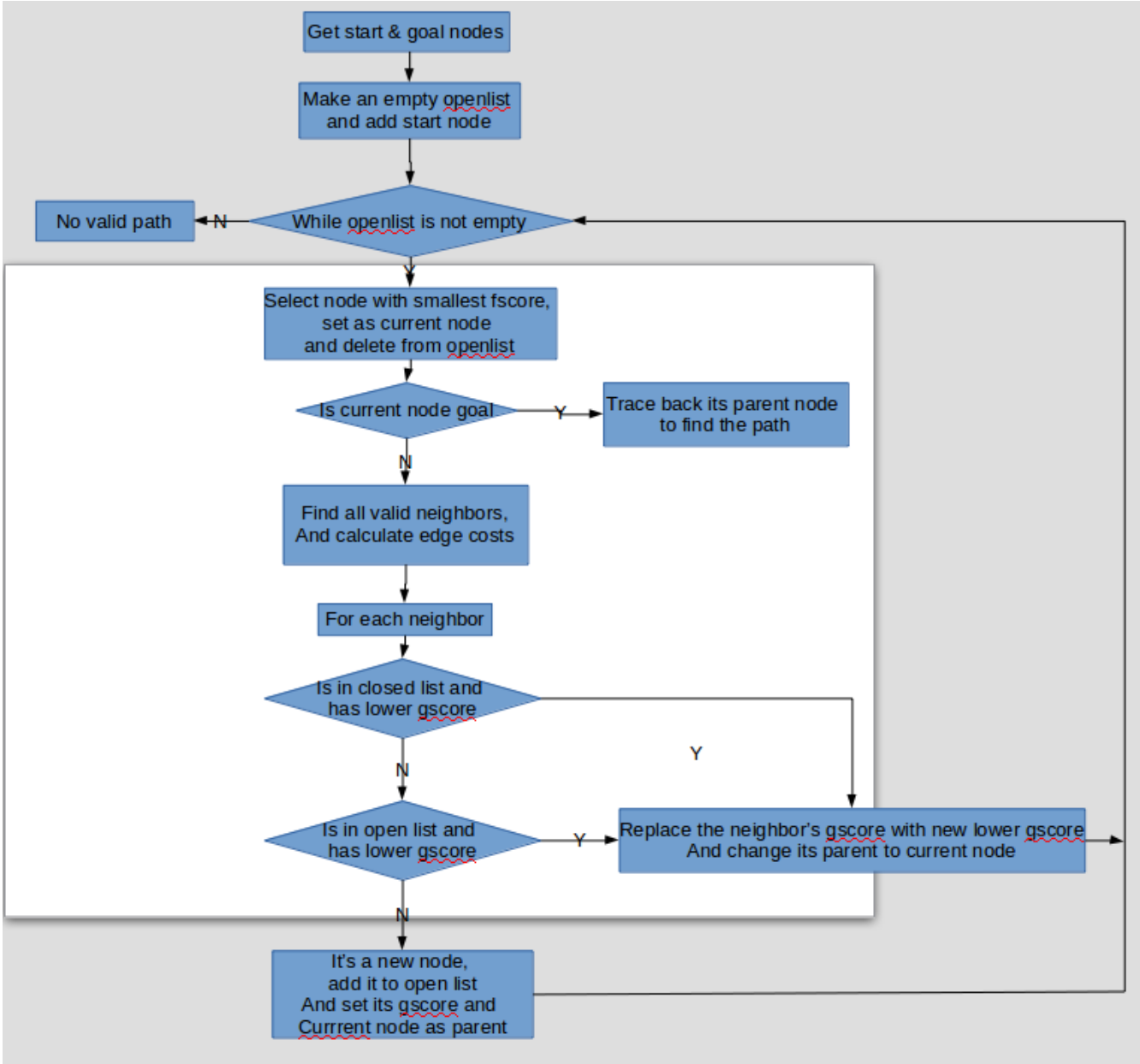
HW2 说明文档

1 算法流程



2 运行结果

表格比较了不同算法方式之间的运行时间、路径长度和遍历总结点数，图片大致显示了运行结果生成的路径，其中黑色路径为 A*算法+DiagonalCost+TieBreaker 生成路径，浅蓝色路径为 JPS+DiagonalCost+TieBreaker 生成路径。

Algorithm Type	Search Time(ms)	Path Cost(m)	Visited Nodes
Dijkstra	589.611985	6.843321	21593
A* with Manhatton	1.118159	6.843321	44
A* with Euclidean(L2)	52.064360	6.843321	1693
A* with L∞	133.433334	6.862596	3779
A* with Diagonal	14.866871	6.843321	405
A* with Diagonal and TieBreaker	14.763537	6.843321	405
JPS with Diagonal	15.864408	6.843321	273
JPS with Diagonal and TieBreaker	13.674994	6.843321	273

表 1 example 1

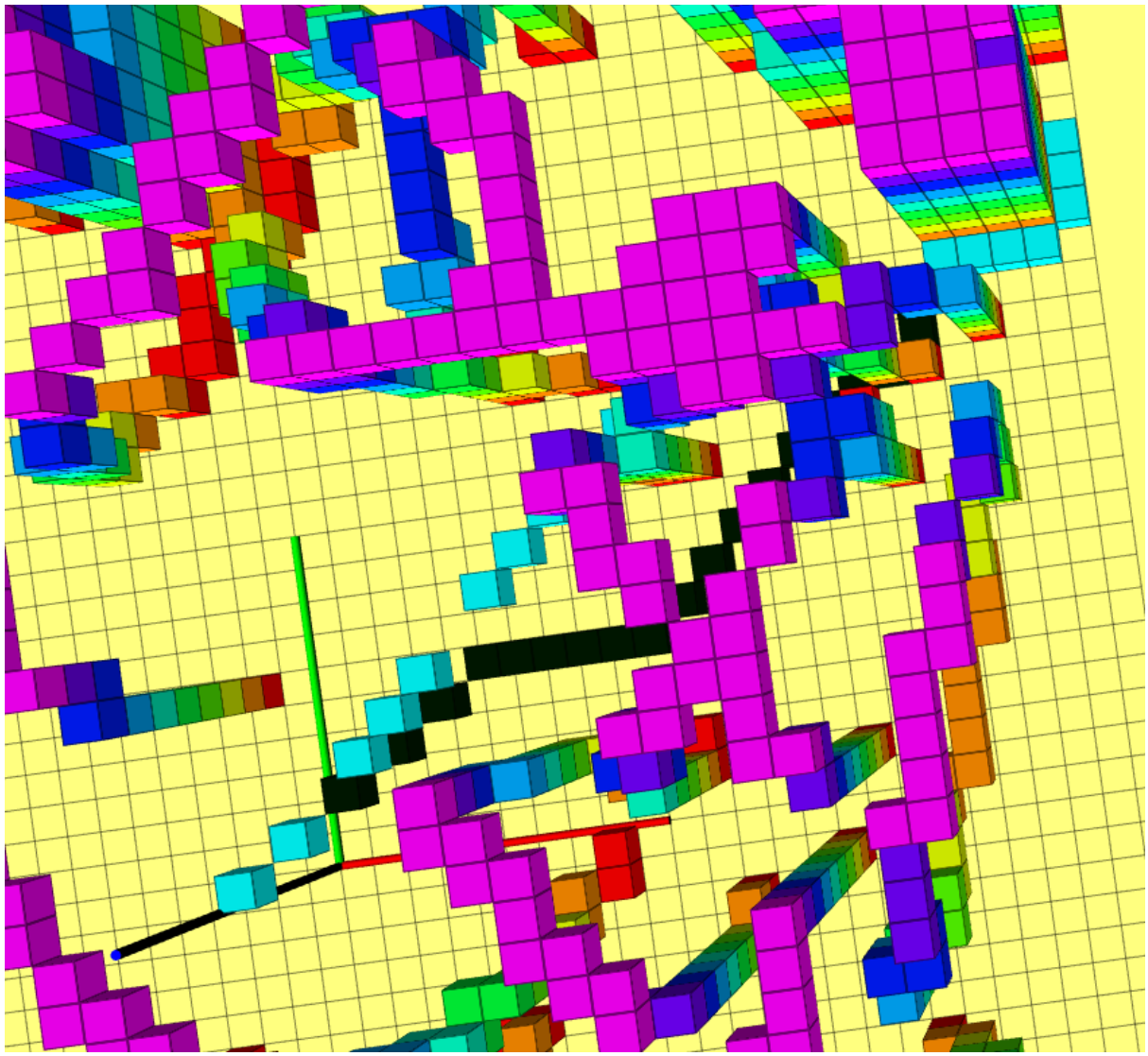


图 1 example 1

Algorithm Type	Search Time(ms)	Path Cost(m)	Visited Nodes
Dijkstra	530.671296	5.980579	20023
A* with Manhatton	2.167830	5.980579	63
A* with Euclidean(L2)	41.327960	5.980579	1319
A* with L∞	84.789892	5.980579	2471
A* with Diagonal	6.523592	6.019129	167
A* with Diagonal and TieBreaker	6.229776	6.019129	167
JPS with Diagonal	6.572204	5.980579	153
JPS with Diagonal and TieBreaker	8.120918	5.980579	153

表 2 example 2

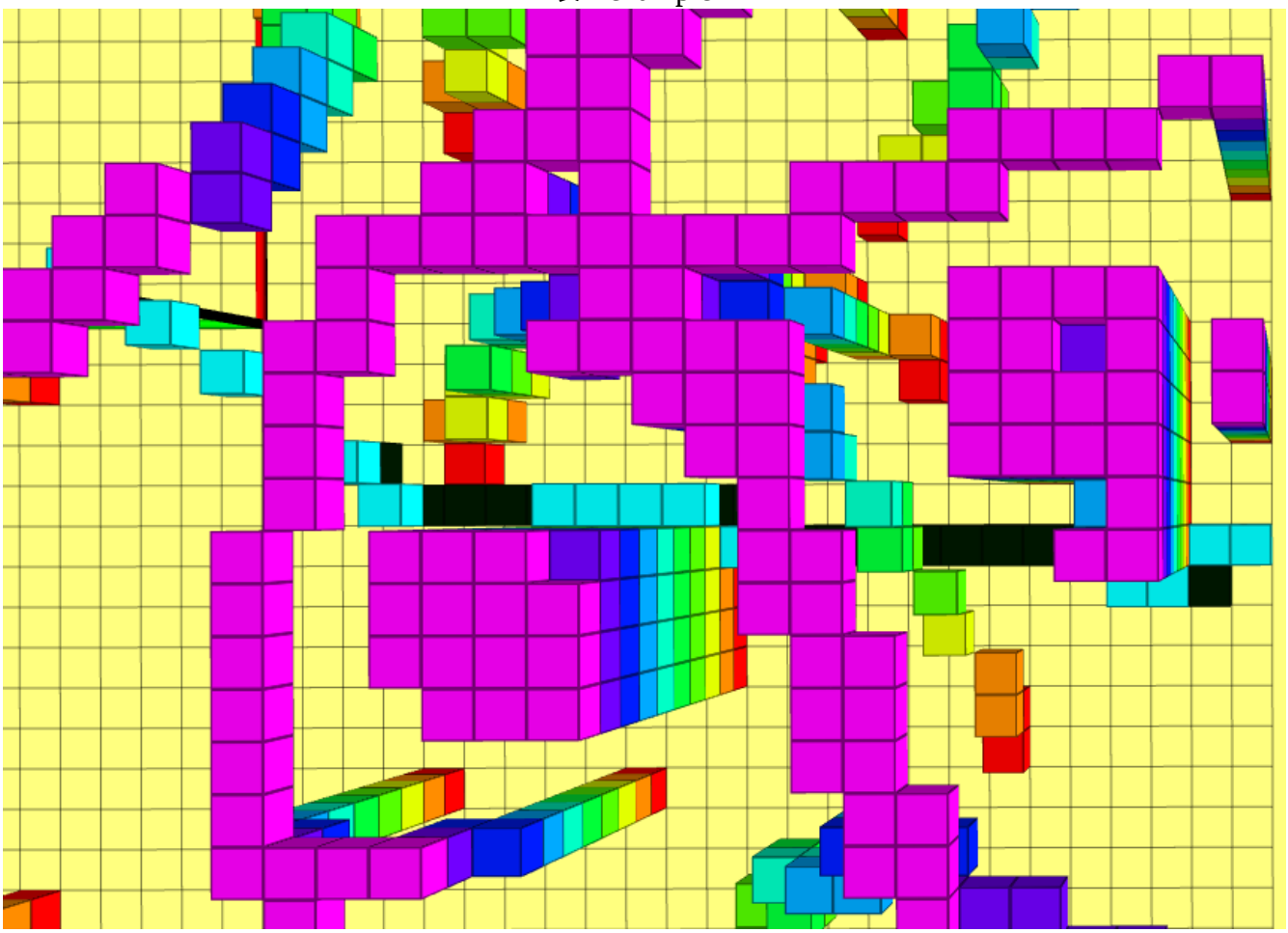


图 2 example 2

Algorithm Type	Search Time(ms)	Path Cost(m)	Visited Nodes
Dijkstra	551.496005	5.926163	19840
A* with Manhatton	0.776904	5.926163	21
A* with Euclidean(L2)	13.702731	5.926163	465
A* with L∞	94.556009	5.926163	2477
A* with Diagonal	4.194535	5.926163	101
A* with Diagonal and TieBreaker	5.670343	5.926163	101
JPS with Diagonal	7.833491	5.926163	90
JPS with Diagonal and TieBreaker	8.305005	5.926163	90

表 3 example 3

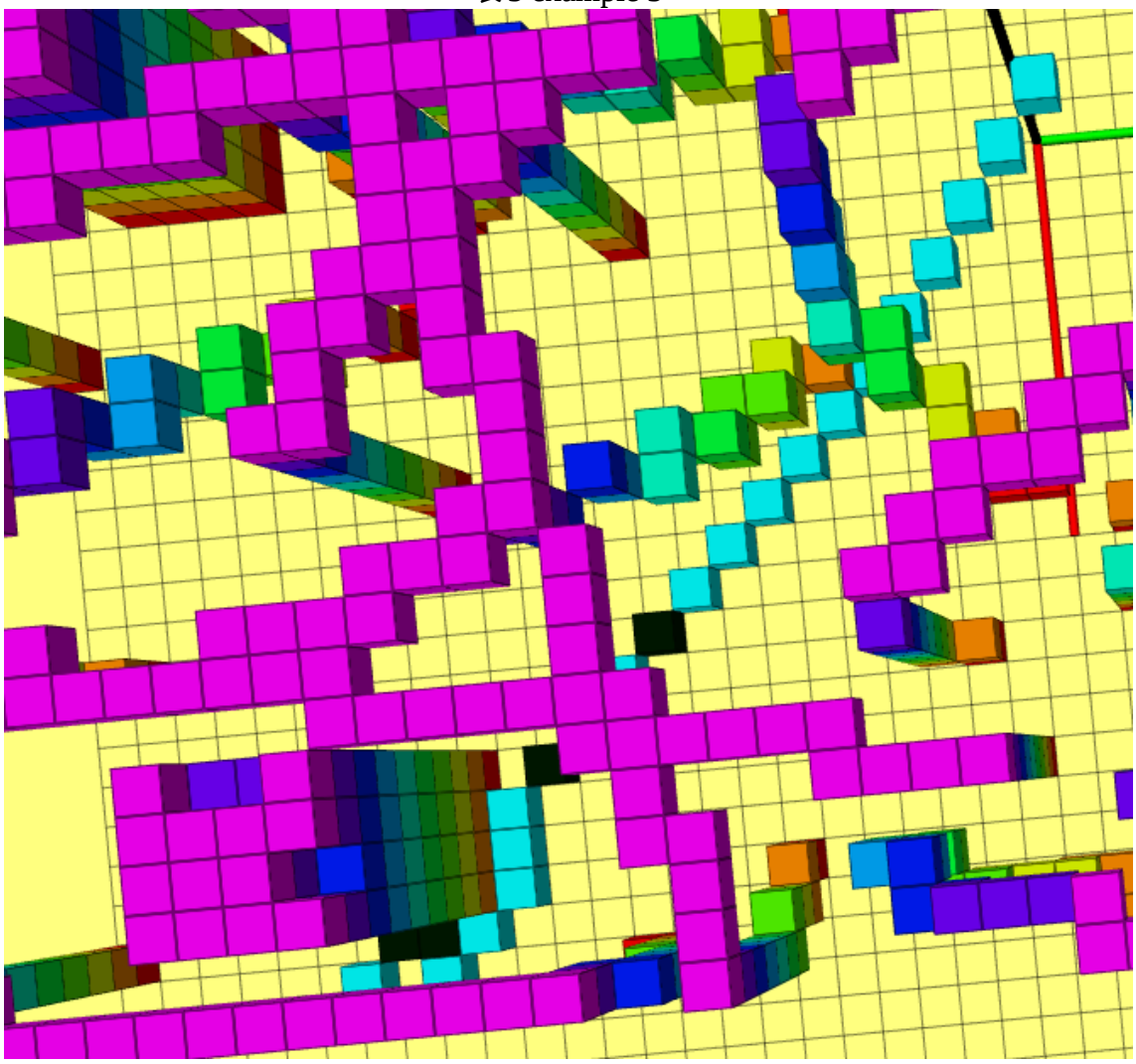


图 3 example 3

Algorithm Type	Search Time(ms)	Path Cost(m)	Visited Nodes
Dijkstra	308.993645	4.146264	10502
A* with Manhatton	0.604309	4.146264	15
A* with Euclidean(L2)	7.580355	4.146264	279
A* with L∞	42.254055	4.146264	1143
A* with Diagonal	1.615815	4.146264	37
A* with Diagonal and TieBreaker	2.017166	4.146264	37
JPS with Diagonal	6.479256	4.146264	62
JPS with Diagonal and TieBreaker	5.405736	4.146264	62

表 4 example 4

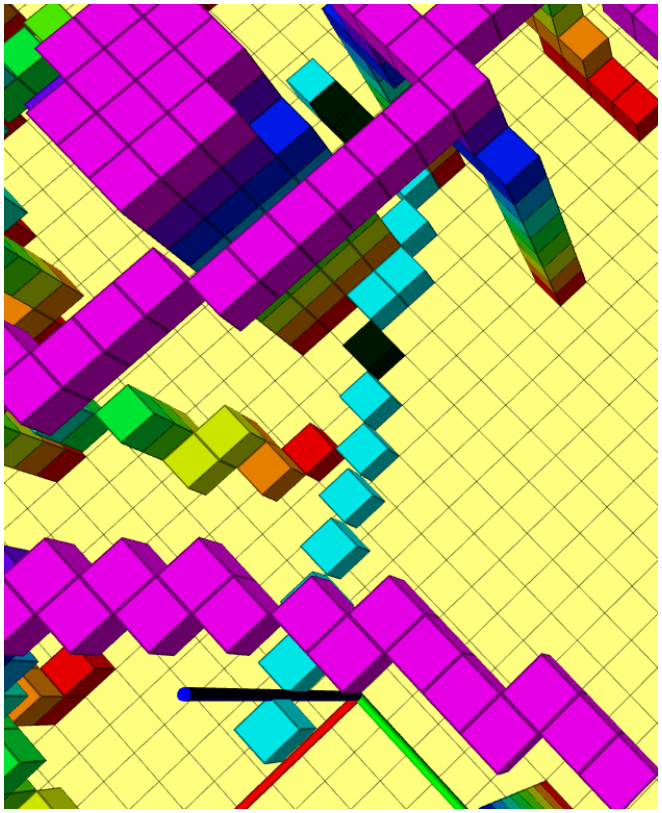


图 4 example 4

3 结论

通过观察数据以及结合算法本身进行分析可以发现：

1. Dijkstra 算法是所有方法中效率最低的，需要遍历的节点数通常是其他方法的一到两个数量级以上，原因是他的搜索没有方向性，但是求出的路径永远是最优的（最短的）；

2. Manhatton 启发函数的效率通常是最高的，需要遍历的节点数通常很少，但是往往最后得出的路径不是最优的（最短的），因为他的距离未必小于真实的最短路径距离，所以搜索不具有完备性；

3. Diagnal 和 Euclidean 距离都满足小于或等于真实最短路径距离的条件，因而都具有完备性，但由于 Diagnal 的距离非常接近真实的最短路径距离，因此搜索空间较小，所需遍历的节点数相对较少；

3. JPS 算法通常比 A*算法所需遍历的节点数要少，但是在某些情况 JPS 需要遍历的节点数会更多（如 example 4）；

4. TieBreaker 的加入不会影响最终所需节点数和路径长度，但是却会增加搜索时间，很可能是 TieBreaker 加的不太合理。