ZTE 迪杰斯特拉派初赛报告

- **团队信息**

团队名称：FFF 团			
团队成员	身份证号码	毕业院校	分工
姚开一	510107199302152693	中国科学院大学	建模，算法
陈璐	210303199201222028	中国科学院大学	建模，算法
刘盼	420984199309083031	中国科学院大学	算法

- **数学模型**

1. 变量和符号

令 $G(V,E)$ 表示无向图，其中点集 V 有 n 个节点，边集 $E=\{(i,\mathrm{j})\,|\,i,\mathrm{j}\in \mathrm{V}\}$，两个节点之间的花费为 $(w_{ij})_{n\times n}$。对于地图中的标红路段 $(\mathrm{N11},\mathrm{N12})$，直接认为两个节点不能直达。

2. 目标函数

（1）判断能否最多只经过 9 个节点就能到达终点：$num\le 9$

（2）求得最优路径：从起点 S 出发，到达终点 E，使得花费最少：

$$\min f(w)=\sum_{j=1}^{num} w_{i_{j-1}i_j}$$

3. 约束函数

1) 必须经过的节点：$N7$、$N12$；必须经过的路径：$(N2,N4)$、$(N14,N13)$

2) 不能经过的路径：$(\mathrm{N11},\mathrm{N12})$

- **模型求解**

一．求解思想

我们采用了**改进的蚁群优化算法**（Ant Colony Optimization, ACO）搜索符合赛题要求的最优路径。蚂蚁在所经过的路径上留下彼此可以识别进行信息传递的信息素，在选择下一步时总是趋向于选择走信息素浓度较高的路径。相对较优的路径，蚂蚁经过的次数会相对较多，该路径上的信息素浓度就相对较大，以后蚂蚁选择该路径的概率就会相对较大，从而促进该条路径的蚂蚁继续增多和信息素的继续变大。以上，大量蚂蚁之间经过适应与协作组成的群体的活动能够表现出一种信息正反馈现象，根据这种正反馈作用，最终蚂蚁通过个体之间的这种信息交流，就能搜索到符合要求的最优路径。

在传统蚁群算法的基础上，我们还做出了改进：

1. 双向搜索策略

设计了双向搜索策略，将蚂蚁等分分配在起点和终点，并行寻优提高蚂蚁的搜索效率，同时，通过相遇机制增强了蚂蚁之间的协作能力。

2. 信息启发因子与方向启发因子相结合

设计了两种启发因子：信息启发因子和方向启发因子。信息启发因子反映了每个节点的信息素对蚂蚁下一步路径选择的作用强度，方向启发因子表面了启发信息对蚂蚁下一步路径选择的影响程度。在两种启发因子的共同作用下，增大了蚂蚁选择较优路径的概率，从而有效地加快了算法的收敛速度。

3. 信息素局部更新与全局更新相结合

传统蚁群算法因为不能实现全局更新而易出现停滞的现象，因此，我们提出了一种信息素局部更新和全局更新相结合的方法。两种信息素相结合后除最优路径上的信息素会得到更新外，花费较少的节点之间的信息素也会得到强化。这一方面加强了最短路径上的信息反馈，加速了算法的收敛；另一方面，也对算法的停滞现象产生一定的抑制作用。

二．算法描述

1. 总体思路

算法总体思路如图 2.1 所示。首先判断能否在 9 个节点以内在满足题目要求的情况下到达终点，如果可以，则再次基础上寻找花费最少的最短路径（即最优路径）；否则，在满足题目要求的情况下寻找花费最少的最短路径（即参考路径）。其中，参考路径考虑了两种情况：

1) 用于在不考虑花费的情况下，使用 **aco1.cpp**，判断蚂蚁能否在 9 个节点内到达终点，如果不能，则输出经过尽量少的节点到达终点的路径；
2) 当已经判断出到达终点的路径的节点数一定大于 9 时，使用 **aco2.cpp**，输出花费最少的最优路径，且路径允许折返。

两个算法都是基于改进的蚁群优化算法输出结果，原理相似，不同之处在于：前者的地图上每个线段的权重相同，后者的地图上线段的权重为实际的权重。

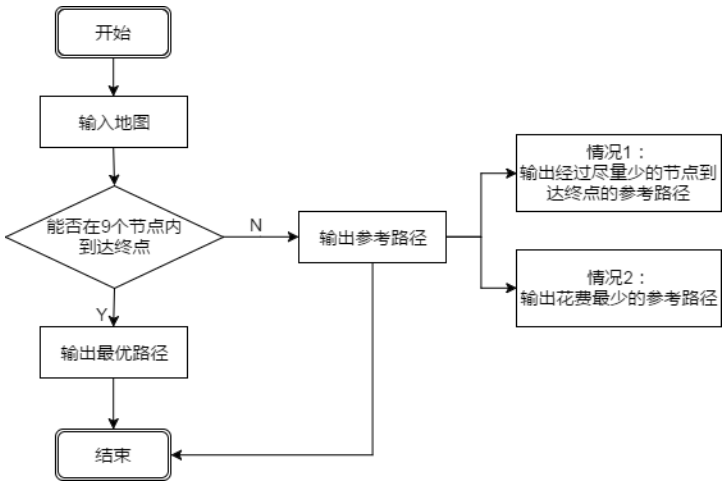

图 2.1 算法总体思路

2. 基于改进的蚁群优化算法的最优路径搜索算法

算法步骤如下：

Step1. 读入地图，初始化邻接矩阵每个元素的信息素，初始化蚂蚁的数量和数据结构。

Step2. 将蚂蚁等分为两组分别置于起点和终点，开始迭代搜索。

Step3. 对于第 t 次迭代：

（1） 执行双向搜索，并求得第 t 次迭代的最优路径。

（2） 比较全局最优路径和第 t 次迭代的最优路径，选取较优的路径最为当前的全局最优路径。

（3） 进行信息素的局部更新和全局更新，再初始化蚂蚁的数据结构。进入第 $t+1$ 次迭代。

Step4. 迭代搜索结束。如果该算法以判断能否在 9 个节点内达到终点为目的，则输出判断结果，并给出最优路径或参考路径；如果该算法以求取花费最少的路径为目的，则输出花费最少的全局最优路径。

三．算法细节

1. 地图的初始化

如图 2.2，用邻接矩阵 $M=(m_{ij})_{v\times v}$ 表示地图 $G(V,E)$，对于地图中的标红路段 (v_{11},v_{12})，直接认为两个节点不能直达。邻接矩阵中的元素定义如下：

$$m_{ij}=\begin{cases} w_{ij} & if\ (v_i,v_j)\in E \\ 0 & if\ (v_i,v_j)\notin E\cup i=j \end{cases}$$

```
int M[18][18] = { 0, 3, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                  3, 0, 1, 0, 1, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0,
                  1, 1, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                  1, 0, 1, 0, 0, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                  0, 1, 2, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                  0, 0, 1, 2, 1, 0, 1, 0, 0, 3, 1, 0, 3, 0, 0, 0, 0, 0,
                  0, 0, 0, 2, 0, 1, 0, 1, 2, 0, 0, 0, 2, 4, 3, 0, 0, 0,
                  0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                  0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0,
                  0, 4, 0, 0, 1, 3, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
                  0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 2, 0, 0, 0, 0, 0,
                  0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,     //11、12直接写成不可达
                  0, 0, 0, 0, 0, 3, 2, 0, 0, 0, 2, 0, 0, 2, 0, 0, 1, 0,
                  0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 2, 0, 1, 2, 2, 1,
                  0, 0, 0, 0, 0, 0, 3, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
                  0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 2, 1, 0, 0, 4,
                  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 0, 0, 0, 1,
                  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 4, 1, 0 };
```

图 3.1 地图的初始化

2. 蚂蚁的定义和初始化

（1） 蚂蚁的定义

对于每只蚂蚁定义数据结构 *MaYi*，该数据结构记录禁忌表 *road*、路径的花费 *len*、蚂蚁的生命状态 *life* 和节点访问标记 *flag*。其中 $life=0$ 表示蚂蚁遇到了“死点”，无路可走；$life=2$ 表示蚂蚁顺利从起点走到了终点，或者从终点走到了起点；$life=1$ 表示蚂蚁还在探索路径。

（2） 蚂蚁的初始化

对于全体蚂蚁初始化时，令 $len=0$，$life=1$。对于从起点出发的蚂蚁，*road[0]*为起点编号；对于从终点出发的蚂蚁，*road[0]*为终点编号。

3. 信息素初始化

对于邻接矩阵上的元素 m_{ij}，假设该元素的信息素为 p_{ij}。初始化信息素时，令各处的信息素相等。在程序源码中，我们令各处的 $p_{ij}=0.2$

4. 信息素的更新

（1） 局部更新

局部更新在当前最好最优路径上加强信息素浓度，而其他的路径上不进行信息素加成。数学表达式如下：

$$p_{ij}=(1-r)\times p_{ij}+\frac{K}{f(t)}$$

其中，r 为信息素更新系数，K 为常数，$f(t)$ 表示第 t 次迭代过程中蚂蚁搜索到的最优路径包含的节点个数。在程序源码中，我们令 $r=0$，$K=1$

（2） 全局更新

全局更新中，引入因子 *ro*，动态更新每次迭代最优路径的全局信息素浓度。数学表达

式如下：

$$p_{ij} = ro \times p_{ij}$$

在程序源码 aco1.cpp 中，我们令 $ro = 0.3$；在程序源码 aco2.cpp 中，我们令 $ro = 0.25$。

5. 信息启发因子与方向启发因子相结合

假设蚂蚁到达了节点 j，且节点 j 有 m 个邻接点，信息启发因子和方向启发因子共同决定了蚂蚁在下一时刻应该走到节点 j 的哪个邻接点。

（1） 信息启发因子

对于图中的每个节点 i，定义它相对于节点 j 的信息启发因子 sd_{ji}。当蚂蚁到达节点 j 时，遍历邻接矩阵的第 j 行，定义 $total$ 为第 j 行元素信息素 p_{ji} 的累加之和，初始化 $total = 0$。遍历第 j 行的时，计算信息素的累加和 $total$，并在遍历到 m_{ji} 时，将此时的 $total$ 值赋予 sd_{ji}。数学表达式如下：

$$\begin{cases} total = total + p_{ji} \\ sd_{ji} = total \end{cases}$$

（2） 方向启发因子

定义方向启发因子 $direction$，在区间 $[1, D]$ 中随机初始化 $direction$，其中 D 为常数。对于节点 j 的每个邻接点 i，分别计算 $D \times \frac{sd_{ji}}{total}$，其中 sd_{ji} 为节点 i 相对节点 j 的信息启发因子，$total$ 为节点 j 全部邻接点的信息素之和。如果节点 i 满足 $direction < D \times \frac{sd_{ji}}{total}$，那么就选择节点 i 为蚂蚁下一个要到达的节点。

在程序源码中，我们令 $D = 160$。

6. 双向搜索策略

双向搜索策略将蚂蚁等分分配在起点和终点，两组蚂蚁同时出发，分别寻找终点和起点。双向搜索过程会出现四种情况：（1）从起点出发的蚂蚁与从终点出发的蚂蚁在地图中的某点相遇；（2）从起点出发的蚂蚁顺利到达终点；（3）从终点出发的蚂蚁顺利到达起点；（4）蚂蚁遇到“死点”，无路可走。最优路径只可能出现在情况（1）（2）（3）中。为了方便描述，算法做如下定义：

定义 1 节点访问信息 Ant，对于节点 i，它的访问信息为 $Ant[i][7]$。其中 $Ant[i][2]$ 用于标识是否有蚂蚁访问过节点 i，如果 $Ant[i][2] = 1$，则表明有蚂蚁访问过节点 i；$Ant[i][0]$、$Ant[i][3]$、$Ant[i][5]$ 分别存储了从起点出发的第一个到达节点 i 的蚂蚁的编号、节点 i 在蚂蚁路径中的序号、蚂蚁到达节点 i 时已经耗费的花费；$Ant[i][1]$、$Ant[i][4]$、$Ant[i][6]$ 分别存储了从终点出发的第一个到达节点 i 的蚂蚁的编号、节点 i 在蚂蚁路径中的序号、蚂蚁到达节点 i 时已经耗费的花费。

定义 2 路径矩阵 Minroad，顺序记录当次迭代的最优路径上的节点。

定义 3 路径长度 mini，记录当次迭代的最优路径的花费

定义 4 最优路径搜索成功标识 $find$，当 $find = 1$ 时，说明当次迭代成功搜索到最优路径。

当次迭代的双向搜索的过程如图 3.2 所示，其中，从起点出发的蚂蚁和从终点出发的蚂蚁并行搜索。当次迭代的最优路径的确认过程如图 3.3 所示，依次根据相遇路径、从起点直达终点的蚂蚁路径、从终点直达起点的蚂蚁路径来寻找当次迭代的全局最优路径。

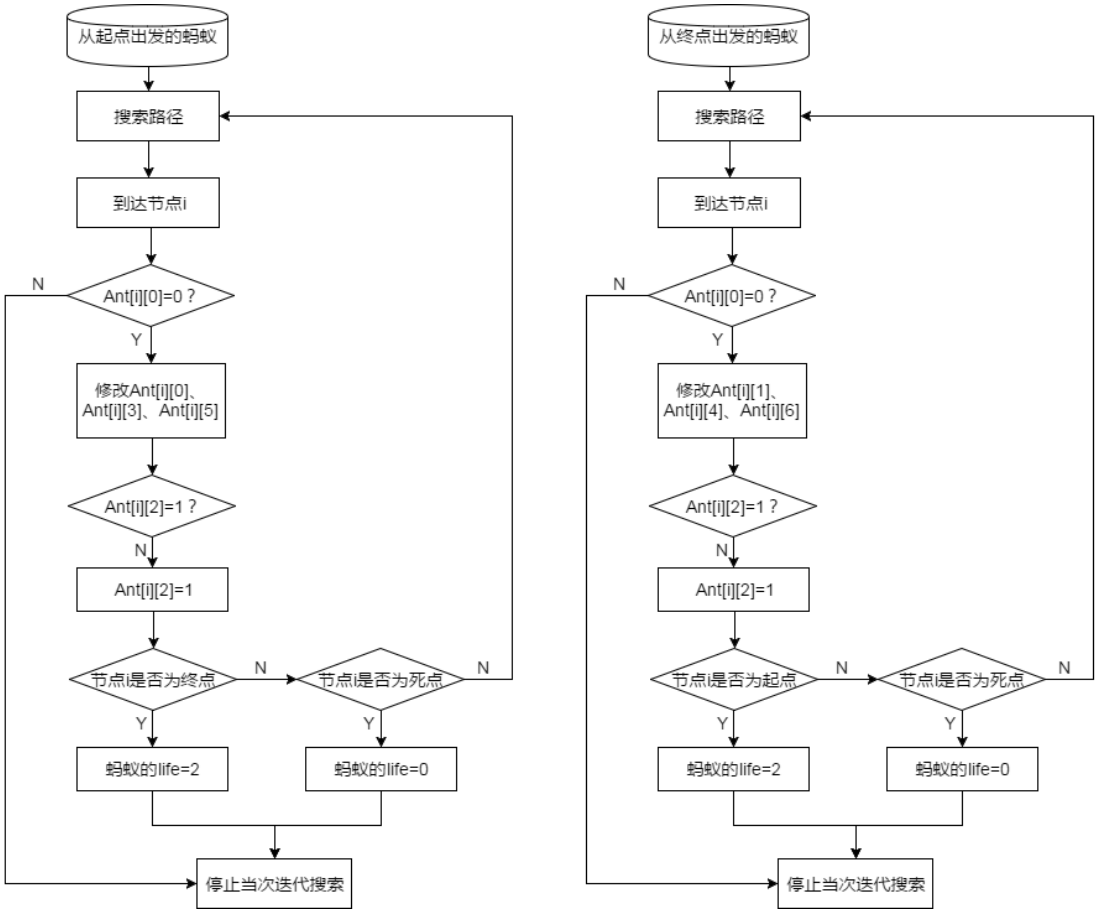

图 3.2 双向搜索过程

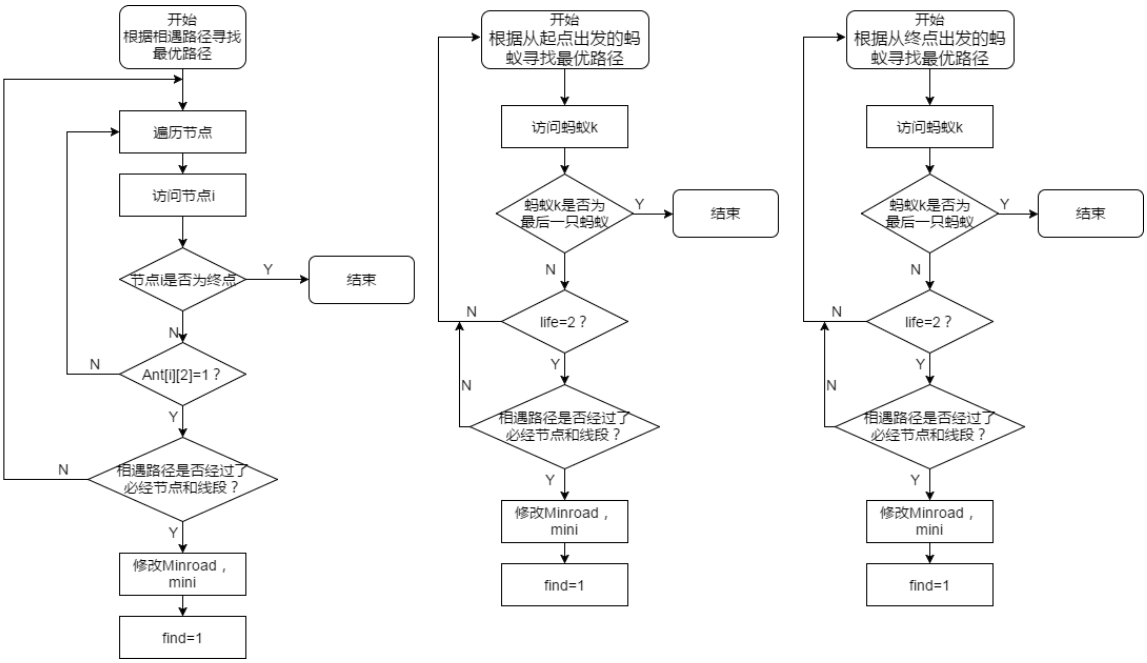

图 3.3 全局最优路径的确定

- **算法实现结果**

编程语言：C++

实验平台：win7 64 位系统，VS2013

针对赛题给出的地图，在满足约束条件的前提下，我们的算法共实现了以下两种情况：

情况 1：以经过最少的节点到达终点为目的

首先判断蚂蚁是否能在 9 个节点内到达终点，如果能，则输出最优路径；如果不能，则输出参考路径。我们定义“参考路径”为：经过最少的节点，到达终点。本题的输出结果如图 4.1 所示。其中，经过的储物间数量为 11 个（含起点和终点），参考路径为：S->N2->N4->N5->N12->N6->N7->N8->N14->N13->E，包含了必经节点和路径，且没有经过不能经过的路径。

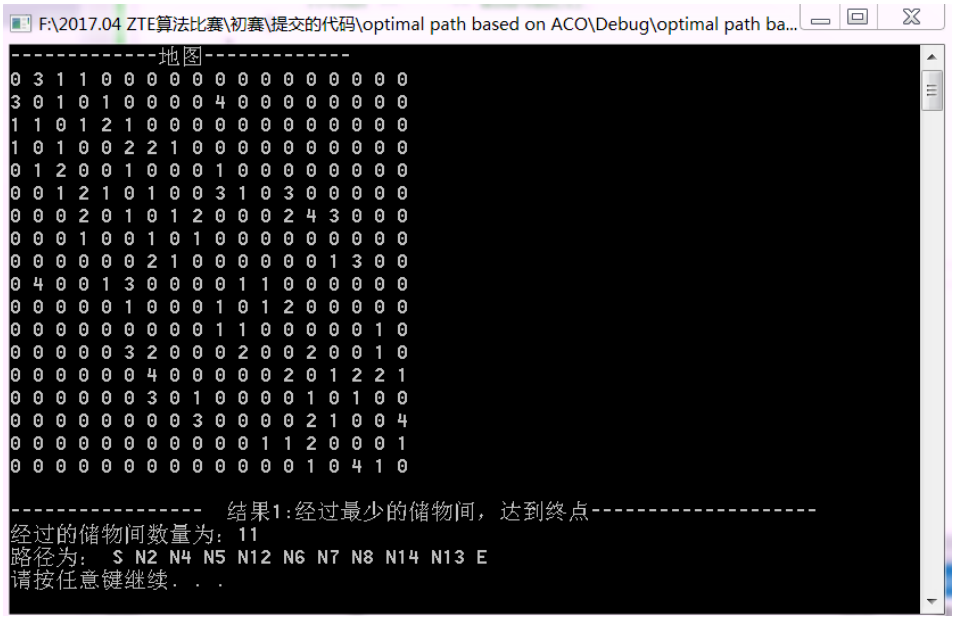

图 4.1 以经过最少的节点到达终点为目的的参考路径

情况 2：以最少花费为目的

如果之前已经判断出蚂蚁不能在 9 个节点以内到达终点，那么还可以将花费最少的路径作为参考路径。本题的输出结果如图 4.2 所示。其中，经过的储物间数量为 12 个（含起点和终点），参考路径为：S->N2->N4->N5-> N6->N7->N8->N14->N13->N12->N16->E，包含了必经节点和路径，且没有经过不能经过的路径；花费为 13。

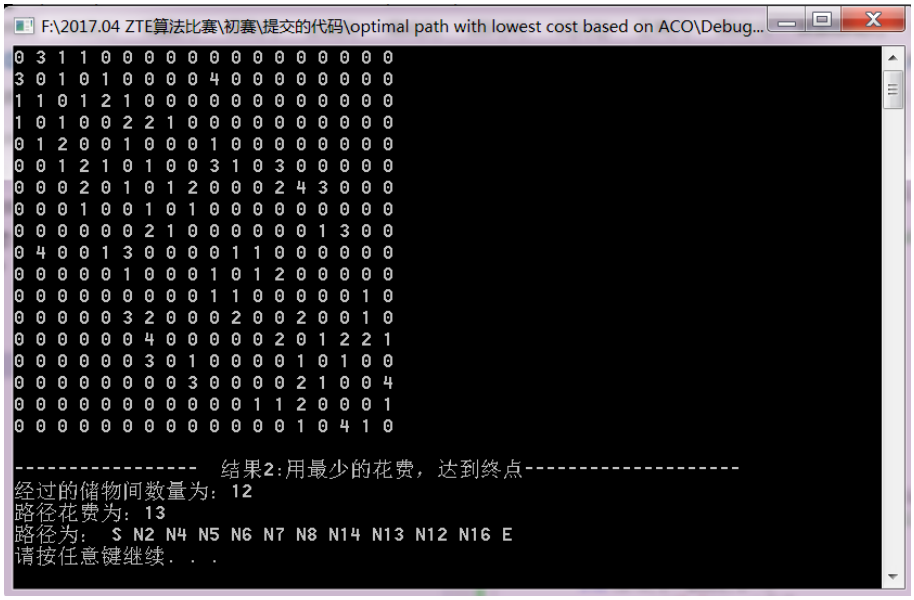

图 4.2 以花费最少到达终点为目的的参考路径

- **算法评价**

我们的算法具有如下优点：

1. 考虑了多种情况的参考路径

当不存在满足全体约束条件的最优路径时，我们分别给出了以经过最少节点到达终点为目的的参考路径，和以花费最少为目的的参考路径。

2. 采用双向搜索策略，提高了并行计算能力

设计了双向搜索策略，将蚂蚁等分分配在起点和终点，并行寻优提高蚂蚁的搜索效率，同时，通过相遇机制增强了蚂蚁之间的协作能力。

3. 信息启发因子与方向启发因子相结合

设计了两种启发因子：信息启发因子和方向启发因子。信息启发因子反映了每个节点的信息素对蚂蚁下一步路径选择的作用强度，方向启发因子表面了启发信息对蚂蚁下一步路径选择的影响程度。在两种启发因子的共同作用下，增大了蚂蚁选择较优路径的概率，从而有效地加快了算法的收敛速度。

4. 信息素局部更新与全局更新相结合

传统蚁群算法因为不能实现全局更新而易出现停滞的现象，因此，我们提出了一种信息素局部更新和全局更新相结合的方法。两种信息素相结合后除最优路径上的信息素会得到更新外，花费较少的节点之间的信息素也会得到强化。这一方面加强了最短路径上的信息反馈，加速了算法的收敛；另一方面，也对算法的停滞现象产生一定的抑制作用。

5. 具有传统的蚁群算法的优点：

1) 具有较好的鲁棒性：对模型稍加修改，就可以求解很多不同类型的问题。

2) 具有较快的计算能力：由于蚁群算法是一种基于群体的随机搜索算法，具有较快的计算能力，所以在求解大规模问题时，可以很明显的减少算法的计算时间。

3) 具有正反馈机制：对于某条路径，当蚂蚁经过的次数相对较多时，该路径上的信息素浓度就相对较大，以后蚂蚁选择该路径的概率就会相对较大，从而促进该条路径的蚂蚁继续增多和信息素的继续变大。这样能够促使蚂蚁集中在最优解的路径上。

4) 易于其他方法相结合：蚁群算法很容易与其他的启发式算法相结合，达到算法间的优势互补，提高算法的求解效率和求解能力。