# 算法的若干细节

目录

[算法的若干细节 1](#_Toc95325697)

[1 位置的编码与解的编码 1](#_Toc95325698)

[2 速度与位置的重编码 2](#_Toc95325699)

[3 适应度函数的设计 2](#_Toc95325700)

[4 用帕累托前沿来替代粒子群优化中的全局最优 3](#_Toc95325701)

[参考文献 4](#_Toc95325702)

## 1 位置的编码与解的编码

粒子的编码和粒子的位置的编码通常具有相同的含义，但是为了区分粒子的速度的编码，我们统一使用位置的编码这一词汇。

常见的PSO（粒子群优化）算法仅能解决单辆车的TSP（旅行商问题）。在本算法设计中，允许多辆车同时配送。

一般的PSO算法对TSP的粒子的位置的编码如下：

其中 代表一个需求点的编号，从 到 恰构成一个 的全排列。这恰好也是TSP中解的编码，因为它可以代表TSP中单辆车寻游所有结点一周。

在本例当中，位置的编码仍然如上。但用于PSO的位置的编码与最终展现结果的解的编码产生了分离。

给定一个位置的编码，就可以计算出在规定好单辆车最大载重量为 ，并且已知每个需求点的需求量时的车辆配送方案。以 5 个结点为例，当其粒子的位置的编码为：

一个可能的问题的解的编码为：

这代表共需要3辆车来配送，其中每辆车的配送路径的非精确表示（精确表示在“适应度函数的设计”一节）分别为：0→2→1→0，0→3→4→0，0→5→0.

在程序当中，会用 来替代 ，因为前者在表示上更加简洁

从粒子的位置编码到问题的解编码，遵循在满足需求的前提下车辆使用数量最小的原则。具体运算过程参见“适应度函数的设计”一节。

## 2 速度与位置的重编码

标准PSO对于速度的编码是和位置的编码维度相同的向量，如下：

其中，任意分量 由于随机数参与计算而可以是小数。在将速度加到位置上后，位置编码也就会存在小数。然而在路径优化问题中，不允许速度和位置中出现小数，遂速度编码与位置编码需要重新设计。

这个过程分为如下6步：

1. 按照原始的速度 更新位置 为 ，即
2. 找到 中的最小值 与最大值
3. 对 中的各分量利用 施以区间变换，将 变换为
4. 对 中的各分量进行四舍五入使得 成为 ，即变连续型数据为离散型数据
5. 此时 中的分量都为 间的正整数，但可能仍然无法构成 的全排列。此时施以最后一次变换，对其中重复的数字只保留第一个，其余的由未出现的数字填补，以保证不重不漏构成全排列
6. 最后，反过来对速度 进行修正，

按照上述步骤，即施以变换 , , 就能最大程度地保证原始速度向量中的信息，又可保证速度与位置编码的合法性。

## 3 适应度函数的设计

*% 多目标适应度函数，计算粒子群 particle 的配送总时间和总成本*

fit = fitness(particle)

*% particle 粒子群，每行是一个粒子（允许只有一个粒子）*

*% fit 适应度矩阵，每行表示一个粒子的适应度，包含 T 总时间和 Z 总成本两元素*

以上是精简后的适应度函数的原型。可以看出，适应度函数对输入的每一个粒子，计算其总时间和总成本。具体步骤为，对于每一个粒子：

首先，按照车辆载重要求，累计一辆车可以服务的车辆数，直到可以将粒子的位置中所有的需求点都满足。此时形成一个向量 . 其中每个元素代表该辆车服务的需求点的个数，而 则代表了所需车辆的个数。这与上面讲的解的编码是一致的。

其次，对于每一辆车，求出其最短路径。例如，如果第1辆车服务两个需求点，这两个需求点对应的编号为3，5. 则该车的非精确路径应该为：0（配送中心）→ 3 → 5 → 0. 由于并非任意两个点之间都有直接边相连，因此，最终该车的行驶路径可能为0 → 2 → 3 → 1 → 5 → 0. 其中2和1仅为途径结点，而非该车的配送结点。

然后，根据上述最短路径，可以计算出：

* 总运输距离。每辆车的运输路径有了，根据欧几里得公式，就可以计算每辆车的运输距离。所有车辆的运输距离加总即为总运输距离。
* 总消杀次数。每辆车的运输路径有了，两两节点间就可以根据风险矩阵计算是否需要消杀。这样所有车辆消杀次数之和也就易得。

最后，根据总运输距离与数据集中给定的车辆行驶速度，就可以计算出总配送时间，. 根据总运输距离与总消杀次数，再加上数据集中给定的车辆行驶的固定成本和可变成本就可以计算出总成本，.

## 4 用帕累托前沿来替代粒子群优化中的全局最优粒子

粒子群优化只能解决单目标优化问题，缘于其全局最优粒子只能保存一个值。参考[1]等对粒子群优化的改造，可以发现其全局最优都被一个“容器”替代，这个容器包含许多最优的结果，也就是非支配解集，也即常说的帕累托前沿。

在本算法中，每次迭代的时候，计算出当前的帕累托前沿。假设当前帕累托前沿为：

其中每行代表一个非支配的粒子的位置，利用fitness() 函数计算其对应的适应度矩阵如下：

则其平均适应度为 ，如果这个平均适应度中的元素均小于上一次迭代时的平均适应度中的对应元素，就更新当前的“全局最优”。如此一来，便能通过迭代不断获得更优的帕累托前沿解。

## 参考文献

[1] 张利彪, 周春光, 马铭, 等. 基于粒子群算法求解多目标优化问题[J]. 计算机研究与发展, 2004(07): 1286–1291.