离散粒子群优化，由旅行推销员举例说明问题

文章·2004 年 1 月

DOI:10.1007/978-3-540-39930-8_8

1 位作者：

莫里斯·克莱克

独立顾问

134 篇出版物 20,141次引用

查看资料

8.离散粒子群优化，由旅行商问题说明

莫里斯·克莱克

法国电信研发中心

经典的粒子群优化是一种在连续定义域上查找数值函数最小值的强大方法。作为一些二进制

由于版本已经成功使用，尝试为离散 PSO 定义框架似乎很自然。为了更好地了解两者的力量

以及这种方法的局限性，我们详细研究如何使用它来解决
众所周知的旅行商问题，原则上对于
这种优化启发式。结果表明离散 PSO 肯定不如
与某些特定算法一样强大，但是，另一方面，它可以很容易地被
针对我们没有良好的专门算法的任何离散/组合问题进行了修改。

8.1 关于“经典”PSO 的几句话

“经典”PSO 的基本原理非常简单。一组移动的粒子
（群体）最初被“扔”到搜索空间内。每个粒子都有
以下功能。

1. 它有位置和速度。
2. 它知道自己的位置，以及该位置的目标函数值。
3. 它会记住迄今为止找到的最佳先前位置。
4.它知道其邻居的最佳先前位置和目标函数值
（变量：当前位置和目标函数值）。请参阅下文了解邻居定义。

从现在开始，我们将认为一个粒子是它自己的邻居之一，并且
所以3和4可以结合起来。

有很多方法可以定义“邻域”[1]，它是一组粒子
与给定的类别相关，但我们可以区分两个类别。

- “物理”邻域，考虑距离。在实践中，
  每个时间步都会重新计算距离，这是相当昂贵的，但有些
  聚类技术需要此信息。
- “社交”邻里，只考虑“关系”。实际上，对于每个粒子，其邻域被定义为位于

  从一开始，就不会改变。请注意，当过程收敛时，
  社会邻里变成了物理邻里。

在每个时间步长，给定粒子的行为是以下两者之间的折衷：
三种可能的选择：

- 遵循自己的方式，
- 走向其先前的最佳位置，
- 走向最佳邻居的最佳先前位置，或走向最佳
  邻居（变体）。

这种折衷由以下等式形式化：

$$v_{t+1} = v_t + c_1(p_{i,t} - x_t) + c_2(p_{g,t} - x_t) \qquad (8.1)$$

$$x_{t+1} = x_t + v_{t+1}$$

在哪里

vt := 时间步 t 处的速度
xt := 时间步 t 的位置
pi,t := 迄今为止在时间步 t 找到的最佳先前位置
pg,t := 最佳邻居之前的最佳位置，在时间步 t
c1 , c2 , c3 := 社交/认知信心系数

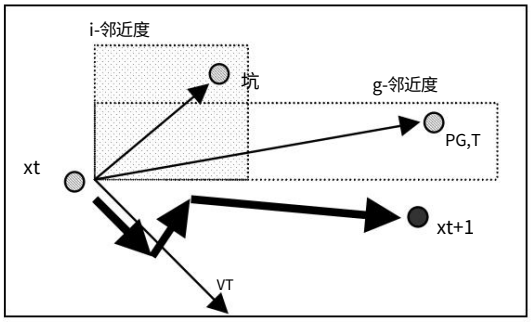


图 1.三种可能动作的加权组合

三个社会/认知系数分别量化：

·粒子现在对自己的信任程度，
·它对自己的经验有多信任，
·它对邻居的信任程度。

值得注意的是，社会/认知系数通常是在某些给定间隔内、每个时间步长和每个速度上随机选择的
成分。这意味着像“走向其先前的最佳位置”这样的规则应该
可以理解为“走向其先前最佳位置周围的区域（邻近区域）”。在经典 PSO 中，这些邻近区域是超平行区域。

当然，人们已经做了很多工作来研究和推广这种方法[2，
3、4、5、6、7、8]。特别是，在下面的讨论中，我使用“nohope/rehope”
技术如[9]中定义，收敛准则在[10]中证明。

8.2 离散粒子群算法

那么，使用 PSO 真正需要什么？

·位置/状态的搜索空间S = {si}

· S上的成本/目标函数f ，将S映射到一组值S → C = c ，并且f的最小值是解状态。 {我}，

· C上的一个阶，或者更一般地，一个半阶，这样对于每对元素
C 的评论（ ，，，ij ）我们可以说我们有 c < c或ij c≥c。 _ _

·如果我们想使用物理邻域，我们还需要距离d
搜索空间。

通常， S是实数空间 右D ，f是数值连续函数。然而，
等式。 8.1 并不意味着情况一定如此。特别地， S可以是有限的

状态集和f一个离散函数,并且一旦您能够定义
遵循基本的数学对象和运算,您可以使用 PSO:

·粒子的位置
·粒子的速度
·减法（位置位置)→速度 ·外乘（实数速度)→ 速度 ·加法
（速度速度） → 速度 ·移动（位置速度） →位置

,
,

为了说明这一论断,我编写了另一个旅行推销员算法。这个选择故意与 PSO 的“通常”使用相去甚远,所以
我们可以了解这种方法的威力,但也了解其局限性。这
目标当然不是与 LKH [11] 等强大的专用算法竞争,
但主要是说“如果你没有针对你的特定算法
离散优化问题,使用 PSO:它有效”。我们处于有限的情况下,所以,
期待任何有关 NFL（没有免费午餐)定理 [12] 的评论,让我们
立即说它在这里不成立,至少有两个原因:可能的目标函数的数量是无限的,并且算法确实通过修改一些参数（和

甚至可选地使用几个不同的目标函数）。

请注意,一些离散 PSO 版本已经被定义并使用
成功地[13, 14],特别是“PSO for TSP”的改进版本[15]。

8.3 TSP 的 PSO 元素

8.3.1 位置和状态空间

令G = {EG ,VG } 为我们在其中寻找哈密顿量的加权图
循环。 EG是加权边（弧)的集合， VG是顶点（节点)的集合。
图节点编号为 1 到N,因此VG的每个元素可以看作是
标签i,i ∈{1,..., N}。 EG的每个元素都是一个三元组
(i, j,wi, j ,i∈VG , j ∈VG ,wi, j只 ε R + 。当我们寻找周期时,我们可以考虑
是N+1 个节点的序列,全部不同,除了最后一个等于第一个
一。这样的序列在这里被称为N 循环并被视为“位置”。所以
搜索空间定义如下:所有N 循环的有限集。

Machine Translated by Google

8.3.2 目标函数

让我们考虑只有当所有弧( )1 , i i+时才像“可接受”的位置 xnn K nn(, , 神经网络+我1), n Vεn n , 1 = N+ 1 这是

nn确实存在。在图中,每个现有弧都有一个

价值。为了定义“成本”函数,一种经典的方法是完成

图,并用任意值l创建所有不存在的弧 超级 足够大到

例如,确保没有解决方案可以包含这样的“虚拟”弧

> + 升 N 种 最小 1) -) (8.2)

sup 最大功率 ij, ( )

最小 w (((,)

因此,每个弧(nn都有我+一个值,一个实值或一个“虚拟”值。现在,在每个位置,可能的目标函数可以简单地定义为

$$FX() = \sum_{n=1}^{N-1} w_{n_i, n_{i+1}} \quad (8.3)$$

该目标函数具有有限数量的值,其全局最小值为确实达到了最佳解决方案的位置(N 循环)。

8.3.3 速度

我们想要定义一个运算符v ,当它在一次期间应用于某个位置时步骤,给出另一个位置。所以,这里,它是N个元素的排列,即说出换位列表。令v为该列表的长度。然后速度定义为

$$v = ((i_k, j_k)),\ i_k \in V,\ j_k \in V,\ k = 1, \ldots, \|v\| \quad (8.4)$$

或者,简而言之 v = i k((k, j)),意思是“交换节点(等等,最后节点() 我1), j, 然后是节点我2, j2 , ‖在‖, j在‖”。请注意,我们需要两个括号,因为它是一个对的列表。

当独立应用于任何位置时,两个这样的不同列表可以生成相同的结果。我们称这种速度为等价,并使用符号≅来表示

表达它。因此,如果1 v并且例如,我们有2是等价的,我们说1 v ≅ v。2

((1,3),(2,5)) ≅ ((2,5),(1,3))。事实上,在这个例子中,它们不仅是等价的,

它们是相反的(见下文):当使用速度在搜索空间上移动时,

这就像一个“球体”:跟随两个相对的点可以到达同一点

路径。零速度是相当于∅ 的速度 , 空列表。

8.3.4 速度的相反

它的定义是

$$ø = vi\ vk\ 1 \left(\left(\| + \| \quad , j \|维克 + 1 \right)\right) \tag{8.5}$$

该公式的意思是"进行与v 中相同的转置,但相反,或者-参见下面的 Ad-德尔"。很容易验证我们有ØØv = v （以及v ⊕ Øv ≅ ∅的"速度加速度"）。 ,

8.3.5 移动（加法）"位置加速度"

设x为位置， v为速度。位置x = x + v通过应用找到
v到x的第一次转置,然后是结果的第二次转置,依此类推。

8.3.5.1 示例

$$x = (1,2,3,4,5,1)$$
$$在 = ((1,2),(2,3))\ 1,2\ , \tag{8.6}$$

将v应用于x,我们依次得到

$$(2,1,3,4,5,2)$$
$$(3,1,2,4,5,3) \tag{8.7}$$

8.3.6 减法"位置减去位置"

设1 x和2 x为两个位置。 x − x之差定义为速度 2 1
v,由给定算法找到,因此将v应用于1 x给出"由给定算法找到"是必要的,因为正如我 2 × 。条件
们所见,两个速度可以
即使它们具有相同的大小,也是相等的。特别地,该算法是
选择以便我们有 $x_2 - x = \leftarrow_1 x - x\ 1\ 2\ (\quad)$, 且x1 = x2 ⇒ v = x2 − x1 = ∅ 。

8.3.7 添加"速度加速度"

设1 v和2 v为两个速度。为了计算1 v ⊕ v我们定义列表 2
转置首先包含1 v的元素（对或转置）,后跟2 v的元素。或者,我们"收缩"它以获得更小的 ,

等效速度。特别是,该运算被定义为v ⊕ Øv = ∅ 。

那么，我们有 $\|v_1 \oplus v\| \leq \|v\| + \|v_1\| \quad \|_2\|$ ， 但我们通常没有 $v_1 \oplus v = v \oplus_2 v$ 。 1

8.3.8 乘法“系数乘以速度”

令c为实数系数， v为速度。有几种情况，看情况
关于c的值。

8.3.8.1 情况 c = 0

我们有cv = ∅ 。

8.3.8.2 情况 c ∈]0,1]

我们只是“截断” v。让cv为小于或等于cv的最大整数所以我们定义cv ((i j)) k , =1,..., $\| \|$ 。
= kk, $\|$简历$\|$ 。

8.3.8.3 情况 c > 1

这意味着我们有 $c = k + ck \varepsilon N c \varepsilon^{*}$, [0,1[。所以我们可以定义
简历 = $\underbrace{\oplus \oplus \oplus \oplus}_{k \text{ 次}}$ VCV 。

8.3.8.4 情况 c < 0

通过写cv = (− c)Øv ，我们只需要考虑前面的情况之一。笔记
如果c是整数，我们有v ≅ v2⇒ cv ≅ cv ，但在
一般情况。

8.3.9 两个位置之间的距离

设x和x为两个位置2这些位置之间的距离被定义
通过d(x ,1 2x) = x ‖ x 1‖ 。注意：这是一个度量，因为我们确实有（x3是任何第三个位置）
tion)：

$$\|x_2 x x - x_1\| \quad \| \quad \|_{1\,2} \quad \|$$
$$\|x_{2\,1} x_{1\,2} - \| = 0$$
$$\|x_2 x x x \leq \|_1 - \| - \|_2 \quad \|_3\| \quad \|_3 \quad \|$$

8.4 算法“PSO for TSP”。核心和变体

8.4.1 方程

我们现在可以重写方程。 8.1如下：

$$v_{i,t+1} = \oplus_1 c_v v_{i,t} \oplus_2 c_p (p_{i,t} - x_{i,t}) \oplus_3 c_g (p_{g,t} - x_{i,t}) \tag{8.8}$$

$$x_{i,t+1} = x_{i,t} + v_{i,t+1}$$

实际上，我们假设$c_1 = c_2 = c_3$没有实验发现不同的选择在某些类别的问题上给出了明显更好的结果。更确切地说，对于给定的问题，经过多次尝试后你可能会找到更好的选择，但是有没有已知的规则可以“提前”找到它。因此，选择$c_1 = c_2 = c_3$不会改变算法以任何重要的方式。通过定义中间位置

$p_{i,t} = p_{i,t} + (p_{g,t} - p_{i,t})/2$，我们最终使用以下系统：

$$v_{i,t+1} = \oplus_1 c_v v_{i,t} \oplus_2 c_p (p_{i,t} - x_{i,t}) \tag{8.9}$$

$$x_{i,t+1} = x_{i,t} + v_{i,t+1}$$

该方程的优点是可以作为选择的指南“好”系数 [10]，即使本文给出的证明应用于可微系统。确实可以这样使用它，但是用这样的直接而简单的方法，群体规模可能需要相当大才能避免陷入局部最小值。因此，对核心算法的一些修改对于提高性能非常有帮助。特别是，No-Hope/ReHope 过程可能非常强大，我们将在下一节中描述。

8.4.2 NoHope 测试

8.4.2.1 标准 0

如果一个粒子必须“朝向”距离为 1 的另一个粒子移动，那么它要么根本不移动，或者它到达与第二个完全相同的位置，取决于社会/信心系数。当群体规模达到小于N(N-1) 时，可能会发生所有移动均按照式（1)计算的情况。 8.9为空。在这种情况下，绝对没有希望改进当前的最佳解决方案。

8.4.2.2 标准 1

[9] 中定义的 NoHope 测试是“群体太小了吗？”。这需要计算每个时间步的群体直径,这是昂贵的。然而，在
像这里所介绍的那样的离散情况,只要之间的距离
两个粒子往往会变得“太小”,粒子变得相同（通常
首先按位置,然后按速度）。因此,在每个时间步,都有一个“减少”的群体
被计算,其中所有粒子都是不同的,这不是很昂贵,并且
NoHope 测试变成“群体是否减少太多？”,比如减少 50%。

8.4.2.3 标准 2

添加了另一个标准：“群体是否太慢？”。这是通过将所有粒子的速度单独或全局地与阈值进行比较来完成的。
在该算法的一个版本中,该阈值实际上在每个时间步长都会被修改，
根据目前得到的最好结果以及弧的统计分布
价值观。

8.4.2.4 标准 3

定义了另一个非常简单的标准：“太多时间没有改善
脚步”。然而,实际上,标准 1 和 2 似乎就足够了。

8.4.3 ReHope流程

一旦“没有希望”,蜂群就会重新扩大。前两种方法
这里使用的灵感来自[9]和[16]中描述的内容。

8.4.3.1 惰性下降法（LDM）

每个粒子都会回到其先前的最佳位置,并从那里开始随机且缓慢地移动 (v = 1),并在找到更好的位置或出现某个位置时立即停止。 ‖ ‖
最大移动次数M 最大限度 达到（在下面的示例中Mmax = N ）。
如果当前群体小于初始群体,则由一组新群体完成
随机选择的粒子。

8.4.3.2 能量下降法（EDM）

每个粒子都会回到之前的最佳位置,并从那里缓慢移动
(‖ ‖=1)只要它在最多M 次移动中找到更好的位置即可。如果当前 最大限度
swarm 比最初的小,它是由随机选择的一组新粒子完成的。您可能会偶然找到解决方案,但它的成本比

LDM。理想情况下,仅当组合方程 1 时才应使用它。 8.9+LDM好像
失败。

8.4.3.3 局部迭代调平（LIL）

这种方法更强大……而且更昂贵。在实践中,应该是
仅当我们知道有更好的解决方案时才使用,但组合方程： 8.9+
EDM 找不到它。

想法如下。目标函数有无限种可能
相同的全局最小值,因此我们可以在本地临时使用它们中的任何一个来指导
一个粒子。对于粒子的每个直接物理邻居y （距离为 1）
x,使用以下公式计算临时目标函数值f (y) t
算法。

8.4.3.4 LIL算法

·查找距离为1 的所有邻居。
·找到最好的一个， ymin 。
·将临时目标函数值f分配给y $_t(y)= (f (y_{min})+ f (x))/2$

并且,根据这些临时评估， x移动到其最佳立即数
邻居。

在 TSP 中,这种算法的成本为 在 $(_{)2}$。如果重复次数过多,则会大大增
加该过程的总成本。这就是为什么只有在以下情况下才应该使用它：
如果您确实想要最好的解决方案,那么确实“没有希望”。在实践中,即使不使用 LIL,l
算法通常也会找到一个好的算法。

8.4.4 自适应ReHope方法（ARM）

上述四种方法（NoHope ReHope/LDM/EDM/LIL)均可
以自适应方式自动使用,具体取决于有多少个时间步
从上次改进的解决方案开始。一个可能的策略给出在
如下表。

表 8.1。 ReHope 可能的策略

无时间步数 改进	瑞霍普型
0-1	没有希望
2-3	惰性下降法
4	能量下降法
>4	局部迭代调平

8.4.5 皇后区

我们不是对每个粒子使用最好的邻居/邻居以前的最好的,而是
可以使用一个额外的粒子来“总结”邻域。这个方法是

[9] 中定义的（独特的)queen 方法和 [17] 中描述的多聚类方法的组合。对于每个邻域，我们迭代地构建
质心并将其作为最佳邻居（方程 8.8 中的pg,t ）。

8.4.6 特优粒子

为了加快处理速度，算法还可以使用特殊的额外粒子
它存储迄今为止找到的最佳位置。这不是绝对必要的，因为这个位置在至少一个粒子中也被记忆为“先前最好的”，但它可能
避免两个 ReHope 进程之间的整个迭代序列。

8.4.7 并行和顺序版本

该算法可以在（模拟)并行模式或顺序模式下运行。
在并行模式中，在每个时间步，计算所有粒子的新位置，然后全局移动群体。在顺序模式下，每个粒子都是
一次移动：粒子 1，然后粒子 2，...然后粒子N，然后再次粒子
1 等。请注意等式。 8.9 隐含地假设了并行模式，但实际上有
在给定的顺序机器上，性能没有明显差异，并且第二种方法稍微便宜一些。

8.5 例子和结果

8.5.1 参数选择

本文的目的主要是展示 PSO 原则上如何使用
一个离散问题。因此，我们对参数使用一些默认值。
当然，通过“优化”当然可以找到一组更好的参数。
最优化”（事实上，PSO 本身可以用来寻找参数的最佳值
空间）。

8.5.1.1 社会/认知系数

∈如果没有明确提及其他内容，则在我们使用的所有示例中（如果使用 NoHope/[0,1] c1
ReHope，则通常为 0.5，如果不使用，则为 0.999）。 c2是随机选择的
每个时间步长为[0,2]。满足[10]中定义的收敛准则。这
准则仅针对可微的目标函数得到证明，但认为它在这里也适用并非没有道理。至少在实践中确实如此。

8.5.1.2 群体规模和邻域规模

理想情况下,最佳群体和邻域大小取决于本地群体的数量
目标函数的最小值,例如nlocal_min ,以便群体可以有
围绕每个最小值的子群。通常,我们根本没有这个
信息,除了最多有N-1 个这样的最小值这一事实。一个可能的
方法是在一开始使用一个非常小的群体,只是为了了解目标函数定义的景观（参见下面的示例）。另外,统计
可以使用弧值分布来估计nlocal_min 。在实践中,我们
通常使用等于N-1 的群体大小 S（参见下面的结构化搜索图）。

关于每个粒子使用三个速度来计算新速度这一事实的一些考虑（尚未完全形式化)表明,一个好的邻域
大小应该只是 4（包括粒子本身）。

但所有这些仍然是“经验法则”,这是 a p 方法的主要局限性:在给定的示例中,我们事先不知道什么是最佳参数,通常必须尝试不同的值。这就是为什么,特别是一些

自适应 PSO 版本目前正在开发中。

8.5.1.3 绩效标准

通常,文献中报告的绩效指标过于具体
数据。例如，“程序在 XYZ 机器上花了 3 秒来解决来自 TSPLIB 的实例 xxxx”。不幸的是,这些数据并不容易与
另一个使用机器 ZYX 的性能测量。因此,我们认为最好采用如下衡量标准:“需要 7432 次巡演评价才能达到
一个办法”。

然而,完全计算 N 循环上的目标函数值或
交换两个节点后更新它并不完全相同:第一种情况是关于
贵N/4倍。因此我们使用两个标准:位置评估的数量和算术/逻辑运算的数量。它仍然不完美,对于

即使使用相同的语言,相同的算法也可以以不同的方式编写
效率较高,而其他则较低。但是,如果您知道您的机器是 xxx
Mips 计算机,您可以估计运行该示例需要多长时间。

8.5.2 一个玩具示例作为说明

在此示例中,我们使用 TSPLIB 中的一个非常简单的图（17 个节点）,称为
br17.atsp（参见附录）。最小目标函数值为39,有
有几种解决方案。虽然它只有几个节点,但它的设计是为了
找到最好的旅行之一并不那么容易,所以它仍然是一个有趣的例子。

8.5.2.1 景观是什么样子

我们随机定义一小群,例如五个粒子,其中pbest是最好的
一。对于每个其他粒子pi ,我们现在考虑线性序列
pk = pi + k (pbest − pi) (N −1), k = 0,..., N −1 绘制图形
k N
( ) ( ) ( ) ( ) fp − fp 。 k 它让我们对景观有一个了解（见图 2 和
3）。景观显然相当混乱,所以我们肯定需要使用
NoHope/ReHope 过程非常频繁。此外,皇后选项可能效率不高（质心不能提供更好的位置）。

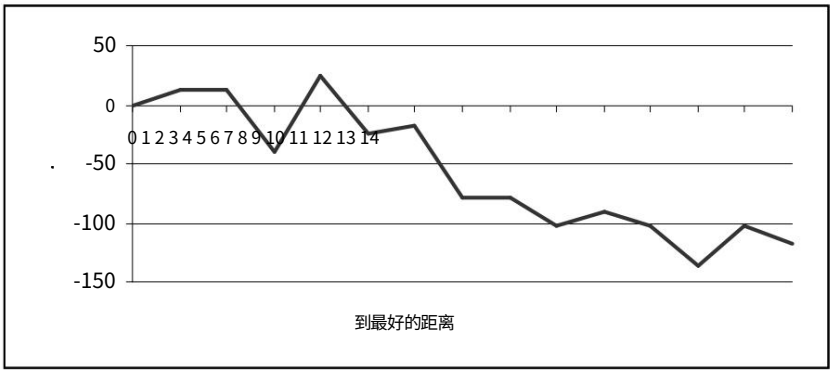


图 2.搜索空间景观的一部分

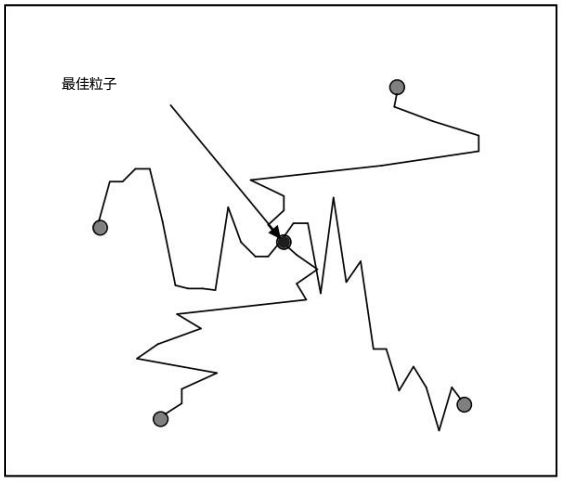


图 3. 5 群的主要部分

8.5.2.2 结构化搜索图。蜂群如何移动

假设我们知道一个解x 。我们考虑了所有可能的位置
搜索空间,并绘制地图(d(x x) f (x)) 。在这张地图上,在每个时间步,
我们突出显示粒子的位置,以便我们可以看到它是如何移动的。我们
根据等价关系定义一些等价类
x x d(x , x) d(x x) 。正如我们所看到的,不仅是全球范围内的群体
向溶液移动,但有些粒子也倾向于向溶液移动
等价类的最小值（就目标函数值而言）。它
意味着即使它没有找到最好的解决方案,它至少找到（并且通常
相当快)有趣的准解决方案。这也意味着如果群体很大
足够了,我们可以同时找到多个解决方案（该属性用于
多目标优化 [18, 19]）。

在我们的示例中,由于位置数量相当大（ $16! \approx 21^{12}$ ）,我们绘制
仅其中少数（4700）,作为图4的图序列的“背景”
5. 我们清楚地看到这个例子的设计有点困难,因为
距离 4 和 8 之间有一个“间隙”。这里,我们使用 16 个粒子组成的群,
它会在一次“标准”运行中找到三个解决方案,如果我们修改,还会找到其他一些解决方案
参数（见附录）。

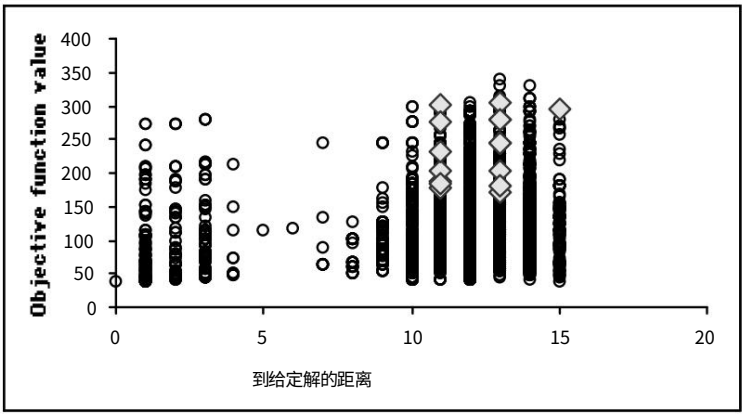
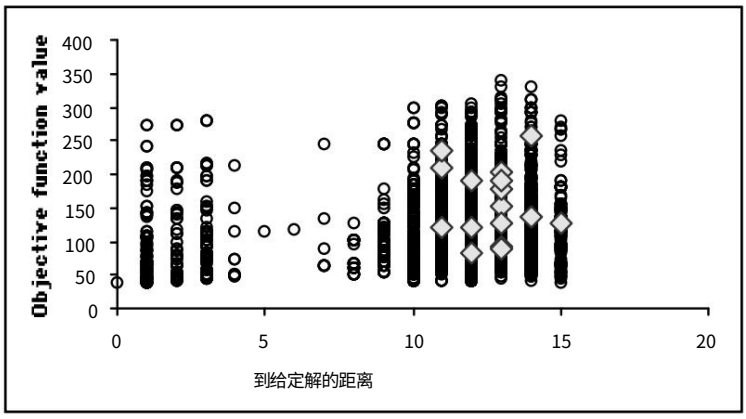
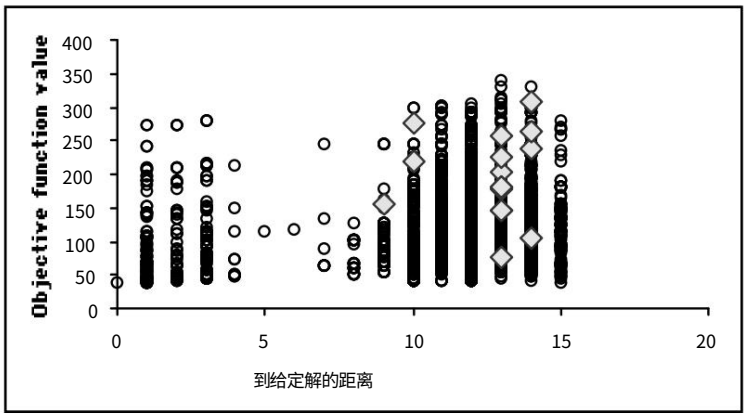
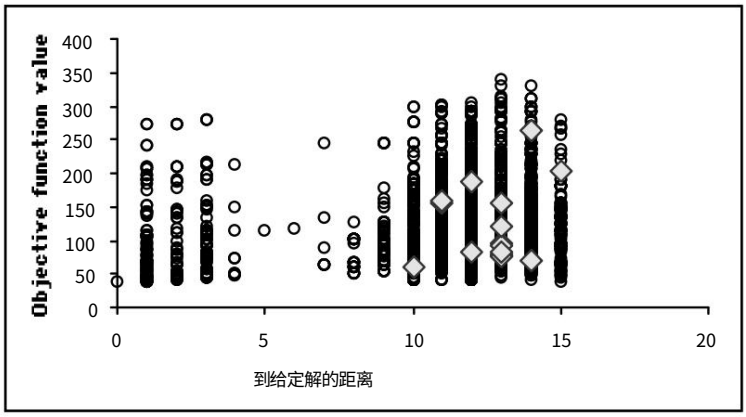
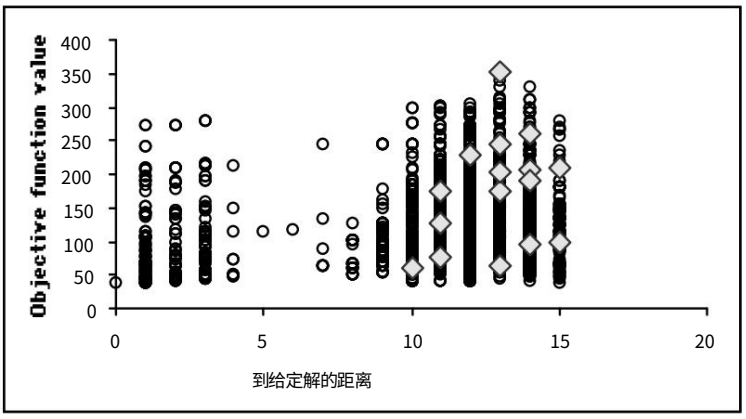


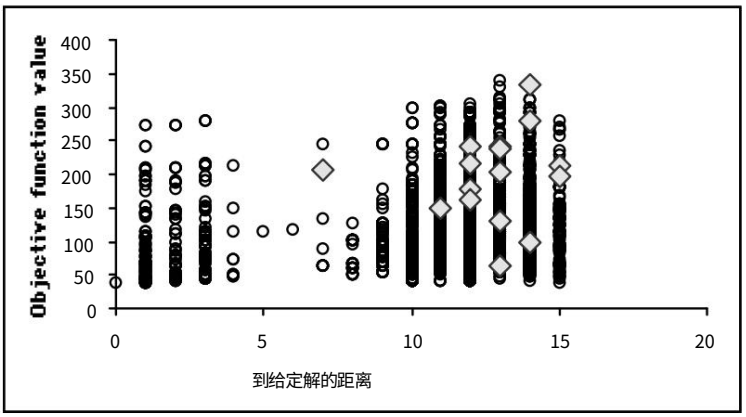
图 4.结构化搜索地图上的初始位置（群大小 = 16）

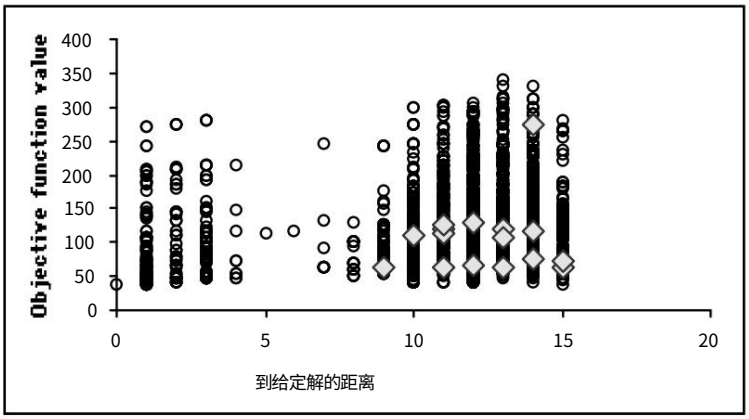












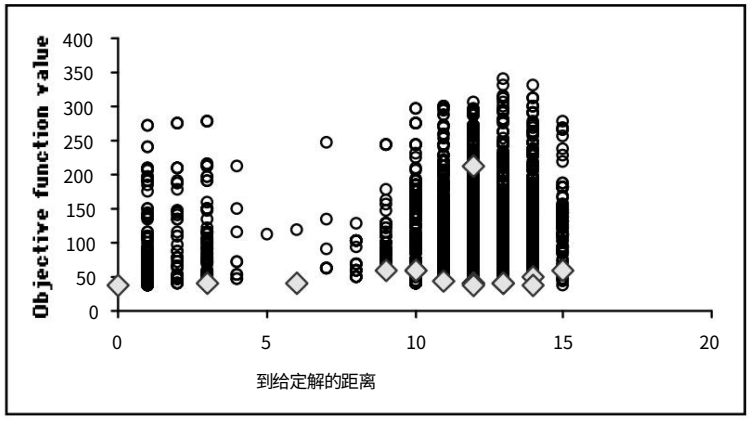


图 5.群体如何在结构化搜索地图上移动（有时在 1
和 480）。跨越距离 9 和 4 之间的差距需要一些时间

8.5.3 一些结果和讨论

表 8.2。在图 br17.atsp 上使用离散 PSO 的一些结果

一群 尺寸	兜帽 尺寸	c1	随机的 c2	重新希望 类型	最佳解决方案
16	4	0.500	]0,2]	手臂	39 （3个解决方案）
16	4	0.500		手臂	39
16	4 与 皇后区	0.500	]0,2]]0,2]	手臂	39 （3个解决方案）
8	4	0.500		手臂	39
		0.500		手臂	44
1	1	0.999		不	47
	4	0.999		不	66
128 32 16	4 4	0.999	]0,2]]0,2]]0,2]]0,2]]0,2]		86

表 8.2 （续）

罩型	旅游评价	算术/逻辑运 算
社会 身	7990	4.8M
体 社会 或	7742	6.3M
	9051	5.8M
身体的		
社会的	4701	2.8M
社会的	926 至无穷大	0.6M

社会的	41837 至无穷大	33.2 M 至∞
社会的	14351 至无穷大	10.8M 至∞
社会的	2880 至无穷大	1.9M 至∞

从表8.2可以看出：

·使用“物理”邻里可能需要较少的旅游评估，但也很多
如果我们考虑算术/逻辑操作的总数（必须在每个时间步重新计算距离），那么它比“社交”选项更昂贵。

事实上，通过社交邻居你可以找到更多的解决方案，这并不是一条普遍规则。

·仅使用 ReHope 方法（在这种情况下，使用s粒子进行T时间步长是
相当于仅使用一个粒子进行sT时间步）不足以达到
最佳解决方案。

·仅使用核心算法，根本没有ReHope，不足以达到
最好的解决方案，除非你可能使用一大群。

·使用皇后并不有趣。

结果不是特别好也不是特别差，主要是因为我们使用了通用的 Re-Hope 方法，该方法不是专门为 TSP 设计的。然而，这是在
目的。这样，只需更改目标函数，您就可以使用离散
针对不同问题的 PSO。

附录

图 br17.atsp（来自 TSPLIB）

在原始数据中，对角线值等于 9999。这里，它们等于 0。
对于 PSO 算法来说，它没有任何改变，并且简化了可视化程序。
名称：br17
类型：ATSP
评论：17_city_problem_（重复）_选项：_39
尺寸：17
EDGE_WEIGHT_TYPE：明确
EDGE_WEIGHT_FORMAT：FULL_MATRIX
EDGE_WEIGHT_SECTION

```
0 3 5 48 48 8 8 5 5 3 3 0 3 5 8 8 5
3 0 3 48 48 8 8 5 5 0 0 3 0 3 8 8 5
5 3 0 72 72 48 48 24 24 3 3 5 3 0 48 48 24
48 48 74 0 0 6 6 12 12 48 48 48 48 74 6 6 12
48 48 74 0 0 6 6 12 12 48 48 48 48 74 6 6 12
8 8 50 6 6 0 0 8 8 8 8 8 8 50 0 0 8
8 8 50 6 6 0 0 8 8 8 8 8 8 50 0 0 8
5 5 26 12 12 8 8 0 0 5 5 5 5 26 8 8 0
```

5 5 26 12 12 8 8 0 0 5 5 5 5 26 8 8 0
3 0 3 48 48 8 8 5 5 0 0 3 0 3 8 8 5
3 0 3 48 48 8 8 5 5 0 0 3 0 3 8 8 5
0 3 5 48 48 8 8 5 5 3 3 0 3 5 8 8 5
3 0 3 48 48 8 8 5 5 0 0 3 0 3 8 8 5
5 3 0 72 72 48 48 24 24 3 3 5 3 0 48 48 24
8 8 50 6 6 0 0 8 8 8 8 8 8 50 0 0 8
8 8 50 6 6 0 0 8 8 8 8 8 8 50 0 0 8
5 5 26 12 12 8 8 0 0 5 5 5 5 26 8 8 0

离散 PSO 找到的一些解决方案

表 8.3 的每一行给出了最小游览的节点序列（常见
总重量=39）。例如,根据上面的矩阵,第一次游览给出
以下权重:5+0+3+0+0+0+8+0+0+0+6+0+12+0+0+5+0=39。

表 8.3。 br17.atsp的一些解决方案

3 14 11 13 2 10 15 7										6 16 4				5		9 17 8 12 1	
6	7 15 16 5				4 17			9		8 10 2 13 11 14							3 12 1
7	6 15 16 5				4		9	8 17 10 11 13 2 14 3 12 1									
9	8 17 5			4 15 6				7 16 13 10 11 2								3 14 12 1	
12 3 14 10 2 13 11 17 8										9		5		4		7	6 16 15 1
12 3 14 10 11 2 13 17 8										9		4		5 16 6 15 7 1			
12 6		7 15 16 5					4		9 17 8				2 11 13 10 14 3 1				
12 8		9 17 5				4 16 15 7						6 10 2 11 13 14 3 1					
12 14 3				2 10 11 13 17 9								8		5		4 15 6	7 16 1
12 14 3 11 2 10 13 15 6 16 7														4		5	9 8 17 1
12 15 16 7					6		5	4		8 17 9 11 10 2 13 14 3 1							
12 16 15 7					6		4	5	9	8 17 10 2 11 13 14 3 1							
12 16 15 7					6		5	4	8	9 17 10 11 2 13 14 3 1							
12 16 7				6 15 5				4		9 17 8 11 10 13 2 14 3 1							
12 17 8				9	5		4 15 16 7					6		2 10 11 13 3 14 1			
12 17 8				9	5		4 15 16 7					6		2 10 11 13 3 14 1			
14 3 10 13 11 2 17 8										9		4		5 15 6 16 7 12 1			
14 3 10 13 11 2								8 17 9				4		5 15 6		7 16 12 1	
14 3 10 11 2 13 8 17 9												4		5 15 7		6 16 12 1	
16 6		7 15 4				5		9		8 17 13 11 10 2						3 14 12 1	
16 6		7 15 4				5		9		8 17 13 10 2 11 3 14 12 1							
17 9		8	4	5	7	6 16 15 10 11 2 13 14 3 12 1											
12 14 3				2 10 13 11 16 6								7 15 4				5 17 9	8 1

参考

[1] Kennedy J.，“小世界和大思想:邻域拓扑的影响
关于粒子群性能”，进化计算大会，
华盛顿特区,1999 年,第 14 页。 1931-1938。

[2] Angeline PJ，“使用选择来改进粒子群优化”， IEEE
进化计算国际会议,阿拉斯加安克雷奇,五月
1998 年 4-9 页， 84-89。

[3] Eberhart RC、Kennedy J.，“使用粒子群理论的新优化器”，
第六届微型机械与人类科学国际研讨会,名古屋，
日本,1995 年,第 14 页。 39-43。

[4] Kennedy J.、Eberhart RC，“粒子群优化”， IEEE 国际
神经网络会议,澳大利亚珀斯,1995 年,第 14 页。 1942-1948。

[5] Kennedy J.，“粒子群:知识的社会适应”,国际进化计算会议,印第安纳州印第安纳波利斯，
1997 年,第 14 页。
303-308。

[6] Kennedy J.，“粒子的行为”,《进化论 VII》,圣地亚哥,加利福尼亚州,1998 年，
p。 581-589。

[7] Shi Y.,Eberhart RC，“粒子群优化中的参数选择”，
进化规划七， 1998，

[8] Shi YH,Eberhart RC，“改进的粒子群优化器”,国际
进化计算会议,阿拉斯加安克雷奇,1998 年 5 月 4 日至 9 日,第 14 页。
69-73。

[9] Clerc M.，“虫群和蜂后:迈向确定性和适应性
粒子群优化”,进化计算大会,华盛顿特区,1999 年,第 1951-1955 页。

[10] Clerc M.,Kennedy J.，“多维复杂空间中的粒子群爆炸、稳定性和收敛”， IEEE
Transactions on Evolution-ary Computation,卷。 6, 1, 2002 年,第 6 页。 58-73。

[11] Helsgaun K.,Lin-Kernighan 旅行的有效实施
推销员启发式,罗斯基勒大学计算机科学系，
丹麦,1997 年。

[12] Wolpert DH,Macready WG,搜索没有免费午餐,圣达菲研究所,1995 年。

[13] Kennedy J.,Eberhart RC，“粒子群算法的离散二进制版本”,系统、人类和控制论会议， 1997
年,第 11 页。 4104-4109。

[14] Yoshida H.、Kawata K.、Fukuyama Y.，“考虑电压安全评估的无功功率和电压控制的粒子群
优化”，
IEEE 传输。电力系统,卷。 2001 年 4 月 15 日,第 15 页。 1232-1239。

[15] Secrest BR,Lamont GB，“粒子群优化中的通信
以旅行商问题为例”,粒子群研讨会
优化,印第安纳州印第安纳波利斯:普渡大学工程技术学院，
2001年，

[16] 何志,魏成,金波,裴文,杨凌,一种新的基于人口的增量模型
旅行商问题的学习方法”,进化计算大会,华盛顿特区,1999 年,第 1152-1156 页。

[17] Kennedy J.，“刻板印象:通过集群提高粒子群性能
分析”,进化计算大会， 2000 年,第 1507-1512 页。

[18] Coello Coello CA、Toscano Pulido G.、Lechuga MS,使用粒子群优化处理多个目标，
EVOCINV-02-2002， CINVESTAV，
进化计算组,2002。

[19] Hu X.,Eberhart RC，“使用动态邻域粒子群优化的多目标优化”，进化计算大会

(CEC 2002),新泽西州皮斯卡塔韦,2002 年,第 14 页。 1677-1681。