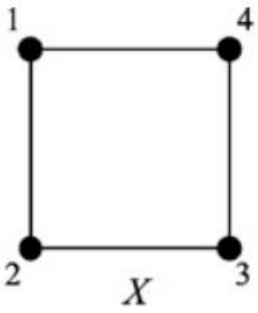
# 《MATLAB 仿真与科学计算》作业

Chen 等[1]的 A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems 提出了一种新的、基于集合的求解离散优化问题的粒子群算法。文中以两种著名的组合优化问题——旅行商问题(traveling salesman problem, TSP)与背包问题为例，证明了所提出的粒子群算法的优势。

按照文中所介绍的步骤，使用MATLAB编写了粒子群算法用于求解TSP，并利用所提出的粒子群算法计算了一个算例进行验证。

## 1、基于集合的粒子群算法介绍

粒子群算法是针对在连续空间上的优化问题所提出的，而本文提出的基于集合的粒子群算法(set-based Particle Swarm Optimization, S-PSO)是用于处理离散的优化问题的。对于 TSP 的表示如图 1 所示。图 1 表示的是一个由 4 个点组成的 TSP，E 指的是全集，在 TSP 中指的是所有可能被连起来的边。E 有四个维度，以$E^1, E^2, E^3, E^4$来表示，分别代表从 1,2,3,4,四个点出发的边。X 指的是一个满足约束的可行解，X 也是一个集合，包含四个维度$X^1, X^2, X^3, X^4$，与 E 相同。




$E = \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$

$E^1 = \{(1,2),(1,3),(1,4)\}$　　$E^2 = \{(1,2),(2,3),(2,4)\}$

$E^3 = \{(1,3),(2,3),(3,4)\}$　　$E^4 = \{(1,4),(2,4),(3,4)\}$

$X = \{(1,2),(2,3),(3,4),(1,4)\}$

$X^1 = \{(1,2),(1,4)\}$　　$X^2 = \{(1,2),(2,3)\}$

$X^3 = \{(2,3),(3,4)\}$　　$X^4 = \{(1,4),(3,4)\}$

图 1 TSP 的表示

S-PSO 中粒子的速度更新公式如下：

$$V_i^j \leftarrow \omega V_i^j + c_1 r_1^j \left(PBest_i^j - X_i^j\right) + c_2 r_2^j \left(GBest^j - X_i^j\right) \tag{1}$$

$V_i^j$指的是第 i 个粒子在第 j 个维度上的速度，$X_i^j$是第 i 个粒子在第 j 个维度上的位置，$PBest_i^j$是第 i 个粒子在第 j 个维度上找到过的最优的解，$GBest^j$是整个粒子群在第 j 个维度上找到过的最优的解(也可以不在整个粒子群中搜索，只在某个特定邻域内搜索)。显然，速度更新公式(1)与连续空间的 PSO 中粒子的速度更新公式形式一样，但是速度、位置和算术运算的公式被重新定义了。

1)位置：粒子的位置指的是一个问题的可行解，是一个集合$X_i$，i 表示第 i 个粒子，$X_i \subseteq E, X_i = X_i^1 \cup X_i^2 \cup X_i^3 \cdots \cup X_i^n$，$X_i^j \subseteq E^j (j \in 1,2,\cdots,n)$。同理，$PBest_i \subseteq E, GBest_i \subseteq E$。

2)速度：粒子的速度被定义为一个带有概率值的集合$V_i$，i 表示第 i 个粒子，$V_i = \{e / p(e) | e \in E\}, V_i^j = \{e / p(e) | e \in E^j\}$。

3)系数*速度：即系数与速度的概率值相乘，当乘积大于 1 时就取 1。

$$cV = \{e/p'(e) \,|\, e \in E\}, p'(e) = \begin{cases} 1, \text{如 } c \times p(e) > 1 \\ c \times p(e), \text{其他} \end{cases} \tag{2}$$

4)位置-位置：即属于前一个位置但不属于后一个位置的元素。

$$A - B = \{e | e \in A \text{ 且 } e \notin B\} \tag{3}$$

5)系数*(位置-位置)：目的是将一个集合变为带有概率的集合，即变为粒子的速度的形式。E′指位置-位置后形成的集合。

$$cE' = \{e/p'(e)\,|e \in E\}, p'(e) = \begin{cases} 1, \text{如 } e \in E' \text{且 } c > 1 \\ c, \text{如 } e \in E' \text{且 } 0 \le c \le 1 \\ 0, \text{如 } e \notin E' \end{cases} \quad (4)$$

6)速度+速度：即两个带概率值的集合中的元素中带概率值较大的加入新的带概率值的集合。

$$V_1 + V_2 = \{e/\max(p_1(e), p_2(e)) \,|e \in E\} \quad (5)$$

图 2 是一个速度更新算术运算的例子，是一个包含 6 个点的 TSP。(a)$V_i$即第 i 个粒子的速度；(b)$V_i^j$即第 i 个粒子在第 j 个维度的速度；(c)$\omega V_i(\omega \in 0.7)$即(a)中的速度乘上系数ω；(d)$\omega V_i^j(\omega \in 0.7)$即第 j 维的速度乘上系数ω；(e)$X_i$即第 i 个粒子的位置，是一个可行解；(f)GBest指在整个粒子群中找到过的最优的解，与$X_i$形式相同；(g)$PBest_i$指第 i 个粒子找到过的最优的解，与$X_i$形式相同；(h)$GBest^1 - X_i^1$即在第一个维度下GBest与$X_i$的减法；(i)$PBest_i^1 - X_i^1$即在第一个维度下$PBest_i$与$X_i$的减法；(j)$c_2r_2^1(GBest^1 - X_i^1)(c_2r_2^1 = 0.5)$即系数$c_2r_2^1$与$(GBest^1 - X_i^1)$的乘法；(k)$c_1r_1^1(PBest_i^1 - X_i^1)(c_1r_1^1 = 0.4)$即系数$c_1r_1^1$与$(PBest_i^1 - X_i^1)$的乘法；(l)更新后的$V_i^1$。

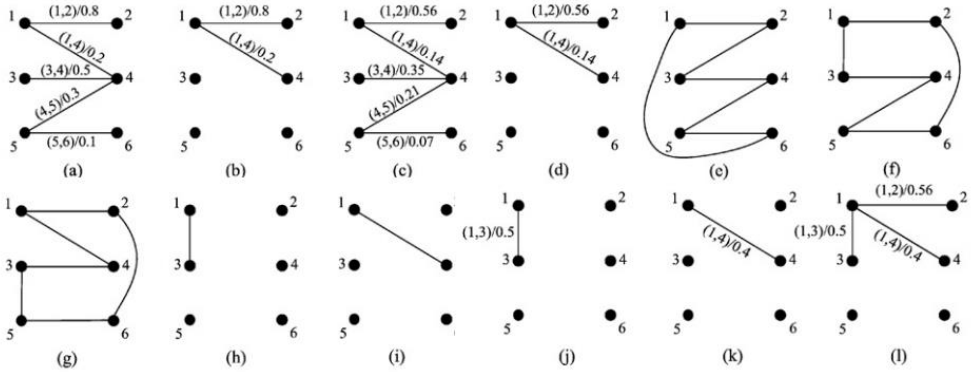



图 2 速度更新算术运算的例子

通过以上公式，粒子的速度更新运算就能够进行了。在更新完粒子的速度之后，粒子 i 将运用更新完的速度$V_i$来调整现在的位置$X_i$，形成新的位置$NEW_X_i$。

粒子位置的更新：首先产生一个随机数$\alpha$，对于上文得到的更新完毕的速度，将概率值大于$\alpha$的元素筛选出来。

$$cut_\alpha(V_i^j) = \{e|\, e/p(e) \in V_i^j \text{且 } p(e) \ge \alpha\} \quad (6)$$

对于 TSP，粒子位置的更新如图 3 所示。图 3 表示的是一个包含 6 个点的 TSP。$(a)X_i$指粒子 i 还未更新的位置；$(b)V_i$指更新完后的粒子 i 的速度；$(c)cut_\alpha(V_i)(\alpha = 0.5)$指粒子 i 的速度经过筛选后的结果；(d)(e)(f)(g)(h)(i)(j)是粒子位置的更新的过程：当$NEW_X_i$的构建尚未完成时，对于每个维度，首先考虑从$cut_\alpha(V_i^j)$中选取一条边进入$NEW_X_i$，随后更新$cut_\alpha(V_i)$；当$cut_\alpha(V_i^j)$为空时，考虑从$X_i$即粒子 i 还未更新的位置中选取一条边进入$NEW_X_i$，随后更新$X_i$，当$X_i^j$为空时，从全集 E 中选取满足条件的边进入$NEW_X_i$。直到$NEW_X_i$的构建完成为止。

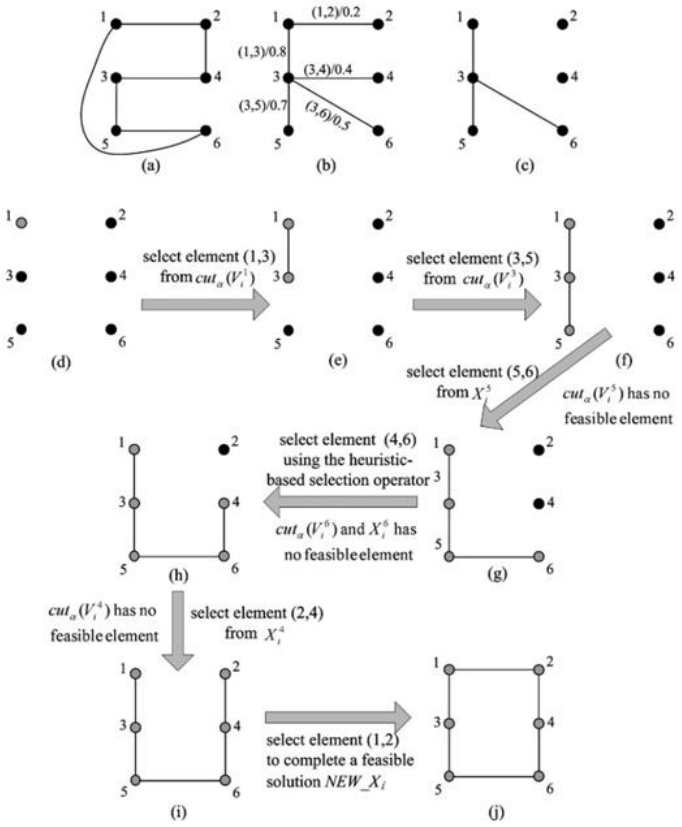



图 3 TSP 位置更新的例子

**2、基于集合的粒子群算法 MATLAB 实现**

按照文中提出的 S-PSO，编写了 MATLAB 程序用于求解 TSP。主要包括粒子群的初始化、粒子群中各个粒子速度与位置的更新等功能。在求GBest$^j$时，为了编程的简单，选择求整个粒子群在第 j 维的最优解，而没有考虑在某个邻域内进行搜索。

为了验证该算法的有效性，利用编写的程序求解一个 70 个点的 TSP(数据来自于 TSPLIB)。设置粒子群中粒子的数量为 40，根据式(1)，设置参数$\omega$=0.8，$c_1 = c_2 = 2$，设置迭代次数为 1000 次。TSP_PSO 是一个脚本文件，用于调用其他函数，控制程序的进行，如图 4 所示。

TSP_PSO.m

```
clear;clc

numPoints=70;%求解的TSP包含的点的个数
numSwarm=40;%粒子群中粒子的数量
numIteration=1000;%迭代的次数
omega=0.8;%omega、c1、c2速度更新时所需参数
c1=2;
c2=2;

[~,points]=readTSPFile('st70.tsp');%读取所需求解的TSP中包含的点的坐标值
points=points(1:numPoints,:);

dists=calDists(points);%计算各点之间的距离，得到一个矩阵
swarm=initializeSwarm(numPoints,numSwarm,dists);%粒子群的初始化，计算各个粒子的位置
swarmPBest=calPBest([],swarm);%记录每个粒子的最优解
GBest=calGBest([],swarm);%记录整个种群得到过的最优解

for i=1:numIteration
    swarm=updateSwarm(swarm,swarmPBest,GBest,dists,omega,c1,c2);%更新粒子群，即更新粒子的速度和位置
    swarmPBest=calPBest(swarmPBest,swarm);%记录每个粒子的最优解
    GBest=calGBest(GBest,swarm);%记录整个种群得到过的最优解
    drawnow%drawnow可以将每次迭代的图形绘制出来
    plotGBest(GBest,points,i);%画图，将最优解的路线绘制出来
end
```

图 4 TSP_PSO

图 5 是迭代 100 次时的最优路线，最优路线的长度是 691，而 TSPLIB 中公布的最优解的长度为 675。由此可见，在问题规模较小时，算法收敛了，并与最优解差距不大。

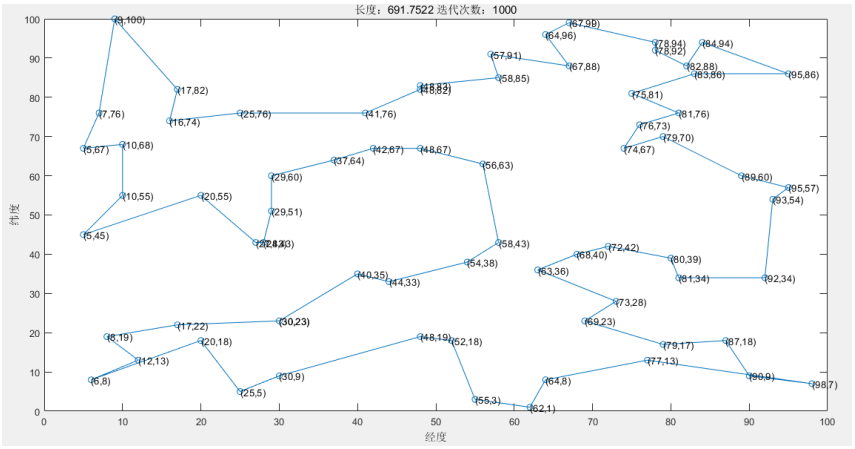



图 5 70 个点 TSP 迭代 1000 次

尝试使用 S-PSO 求解较大规模的 TSP。利用编写的程序求解一个 280 个点的 TSP(数据来自于 TSPLIB)。图 6、图 7 分别是迭代 3 次与迭代 100 次时的最优路线，迭代 3 次时最优路线的长度为 17659，而迭代 1000 次时的最优路线的长度是 4177。而 TSPLIB 中公布的最优路径的长度为 2579。由此可见，当问题规模较大时，虽然算法收敛了，但是离最优的结果还有一定距离。

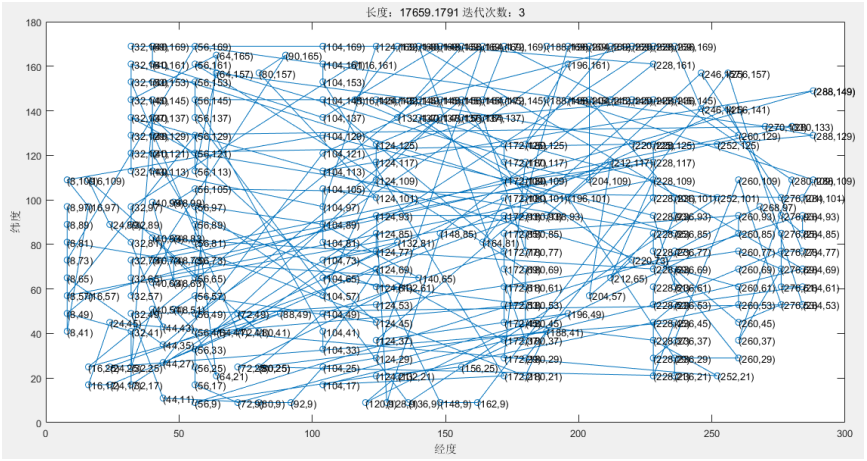



图 6 280 个点 TSP 迭代 3 次

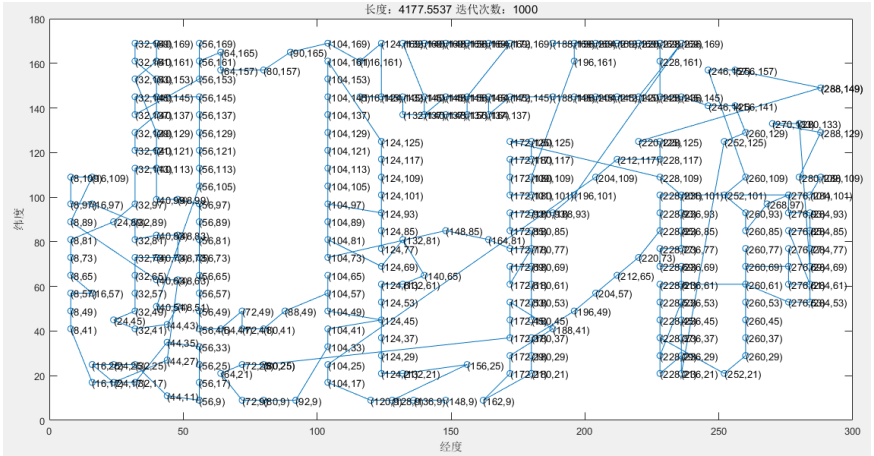



图 7 280 个点 TSP 迭代 1000 次

# 参考文献

[1] Chen W, Zhang J, Chung H S H, et al. A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems[J]. IEEE transactions on evolutionary computation, 2010,14(2):278-300.