廣東工業大學

QG 中期考核详细报告书

题　　目　　论文复现

学　　院　　计算机学院

专　　业　　人工智能

年级班别　　23 人工智能创新班

学　　号　　3123004832

学生姓名　　李圣

2024 年 4 月 5 日

目录

一、论文采用模型解析

论文使用的模型为

$$x_i(t+1)=\frac{\sum_{j:|x_i(t)-x_j(t)|<1} x_j(t)}{\sum_{j:|x_i(t)-x_j(t)|<1} 1}. \qquad (1)$$

其中 **$x_i(t)$**是代理 **i** 在时间 **(**或迭代次数**) t** 的观点值

$j:|x_i(t)-x_j(t)|<1$ 表示满足该条件（观点值差值至多为 **1**）的代理 **i** 和代理 **j** 互为邻居，

$x_i(t+1)$等于所有邻居的观点值的平均值

二、论文第Ⅱ节略解析

命题一：如果 xi(0)≤xj(0),那么对于所有 t，xi(t)≤xj(t)

即每次迭代中 $x_i(t)≤x_j(t)$ 恒成立，作者在此处证明使用了数学归纳法，他用 $N_i(t)$ 表示所有代理 i 的集合，$N_j(t)$d 代表所有代理 j 的集合，$N_{ij}(t)$ 代表所有代理 i 和代理 j 的集合，对于任意 $k_1∈N_i(t)$，$k_2∈N_{ij}(t)$，$k_3∈N_j(t)$，已知 $x_i(0)≤x_j(t)$，易得 $x_{k1}(t)≤x_{k2}(t)≤x_{k3}(t)$，因此 $\bar{x}_{Ni}≤\bar{x}_{Nij}≤\bar{x}_{Nj}$,即三个集合的平均数同样满足不等式，根据模型（1）可得

$$
x_i(t+1)=\frac{|N_{ij}|\bar{x}_{N_{ij}}+|N_i|\bar{x}_{N_i}}{|N_{ij}|+|N_i|}\leq\bar{x}_{N_{ij}},
\quad\text{and}\quad
x_j(t+1)=\frac{|N_{ij}|\bar{x}_{N_{ij}}+|N_j|\bar{x}_{N_j}}{|N_{ij}|+|N_j|}\geq\bar{x}_{N_{ij}}
$$

其中论文使用|A|来表示集合 A 的基数，即集合元素的个数

由此可得该模型一般性排列结论：当 i<j 时，xi(t)≤xj(t)

命题二：将 x(t)排序，最小的 x_1 不会随迭代次数而变小，最大的 x_n 不会随迭代次数而变大,如果在某一次迭代后 $x_i(t)$ 与 $x_i(t+1)$ 的距离大于或等于 1，整个智能体系统将被分为两个子系统(一个是 1~i，一个是 i+1~n)

与其他模型不同的是，该模型的意见值的平均值不一定会被保存，且方差偶尔会上升

由此命题引出了定理一

- **定理一**：假如 $x(t)$ 出自于模型(1)，那么对于任何 i，$x_i(t)$ 都将在有限的次数内收敛到一个极限 x_i^*，此外，对于任意 i 和 j，我们都会有 $x_j^*=x_i^*$ 或 $|x_j^*-x_i^*|\geqslant 1$(后者即为分为两个子系统的情况)

作者在此处使用反证法，假设 $x(0)$ 已被排序，隐含条件是此时 $X(0)$ 相邻两个互为邻居。

由于 x_1 不会随迭代次数增加而减小，且 $x_n(0)$ 为其上界，因此它会收敛到一个值 x_1^*，设 p 是 $x(0)$ 收敛到 x_1^* 的最大索引，即 x_1 到 x_p 能收敛到 x_1^*，而 x_{p+1} 及其之后的代理均不能收敛到 x_1^*

作者需要证明如果 $p<n$，将会有一次迭代会使 $x_{p+1}(t)-x_p(t)\geqslant 1$，因此作者假设 $x_{p+1}(t)-x_p(t)<1$ 恒成立，即该模型将始终为一个系统

随后作者根据上述附加条件推导：

设定某值 $\epsilon>0$，并选定一个时间点，在这时间点之后对于 i 从 1 到 p，x_i 到 x_1^* 的距离小于 ϵ($|x_i-x_1^*|<\epsilon$)。因为 x_{p+1} 不会收敛到 x_1^*，对于某值 $\delta>0$，将会有更多的迭代次数时 $x_{p+1}>x_1^*+\delta$

对于这样的时间 t，将会有 $x_p(t+1)$ 至少为

$$\frac{1}{p+1}\left(\sum_{i=1}^{p+1} x_i(t)\right) \geq \frac{1}{p+1}\left(p(x_1^* - \epsilon) + (x_1^* + \delta)\right)$$

三、实验效果图的复现

关键有两个函数：

```
def update_opinions(opinions):# 更新代理观点
    new_opinions = np.zeros_like(opinions)
    for i, opinion in enumerate(opinions):
        neighbors = [neighbor for neighbor in opinions
if abs(opinion - neighbor) < 1]
        if neighbors: # 如果存在邻居
            new_opinions[i] = np.mean(neighbors)
        else: # 如果不存在，则保持不变
            new_opinions[i] = opinion
    return new_opinions
```

```
def simulate_model(L, agents_num, max_t=15):#迭代模拟
    opinions = np.linspace(0, L, agents_num)  # 初始观点
在[0, L]上均匀间隔
    opinion_t = []  # 存储每次迭代后的观点
    opinion_t.append(opinions.copy())
    for t in range(max_t):
        new_opinions = update_opinions(opinions)
        opinion_t.append(new_opinions.copy())
        opinions = new_opinions
    return np.array(opinion_t)
```

Fig.2：

描述为：不同簇在平衡时的位置，对于一个 L，有初始观点均匀分布在[0.L]的 5000L 智能体，及利表现出它们相对于 L/2 的位置。虚线表示初始观点在 0 到 L 分布的端点。

该图可以很好的展现出智能体收敛的稳定性，直观体现出每个 L 智能体分散情况

复现代码主要思想是循环遍历 L，对每个 L 算出最终收敛的集合 opinions，再遍历 opinions 通过计算每个智能体的邻居的平均数放入 temp，放前检查是否有重复，迭代完后将所得的坐标画为散点图

```
for t in range(max_t):
    new_opinions = update_opinions(opinions)
    opinions = new_opinions
for opinion in opinions:
    neighbors = [neighbor for neighbor in opinions if abs(opinion - neighbor) < 1]
    if neighbors:  # 检查邻居列表是否为空
        n = np.mean(neighbors)
        if np.any(x == n):  # 检查x中是否已经存在n
            continue
        x = np.append(x, L)
        y = np.append(y, n - L * 0.5)
```

由于设备和耗时限制，我在此每个 L 仅用 500L 个智能体，L 仅选取 0 到 25，复现图如下：

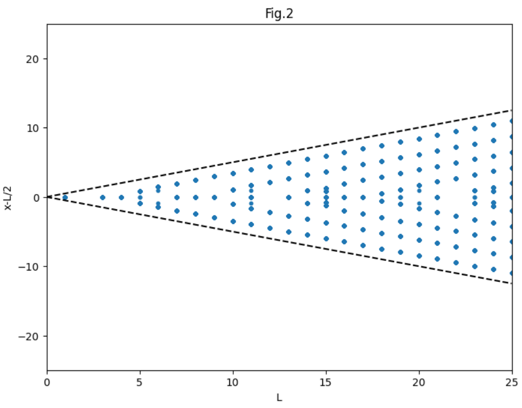

Fig.3

描述为：当初始意见在半无限区间上均匀分布时，每单位长度的密度为 100 的时间演化。代理组与其余代理分离，并收敛到相隔约 2.2 的集群

复现代码采用上限为 100 进行迭代，图只选取 0 到 8，传入参数 L=100，agents_num=100*L,

复现图如下：

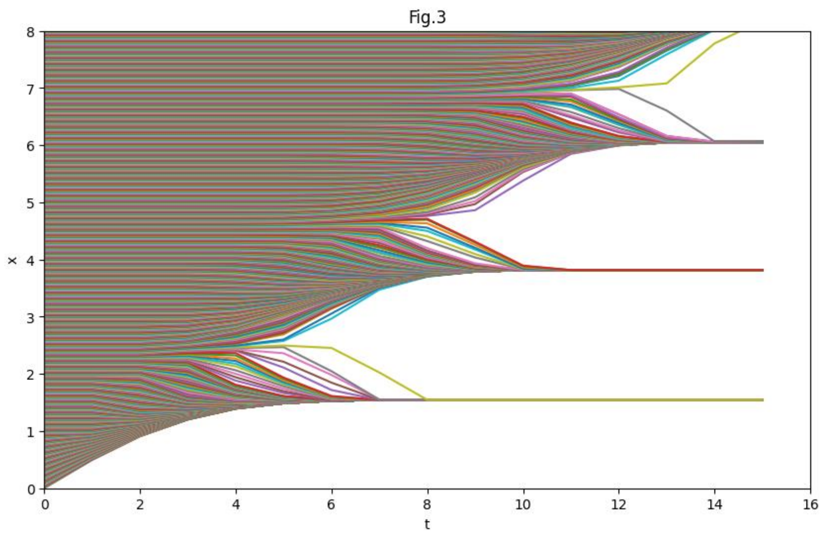

Fig. 4

描述为：临时“元稳定”状态的示例。最初，形成了两个彼此不相互作用的群体，但它们都与介于两者之间的少数代理相互作用。结果，两组之间的距离缓慢减小，最终小于 1。在这一点上，这些群体直接相互吸引并合并成一个集群。

复现代码传入参数 L=4，agents_num=1000,复现图如下：

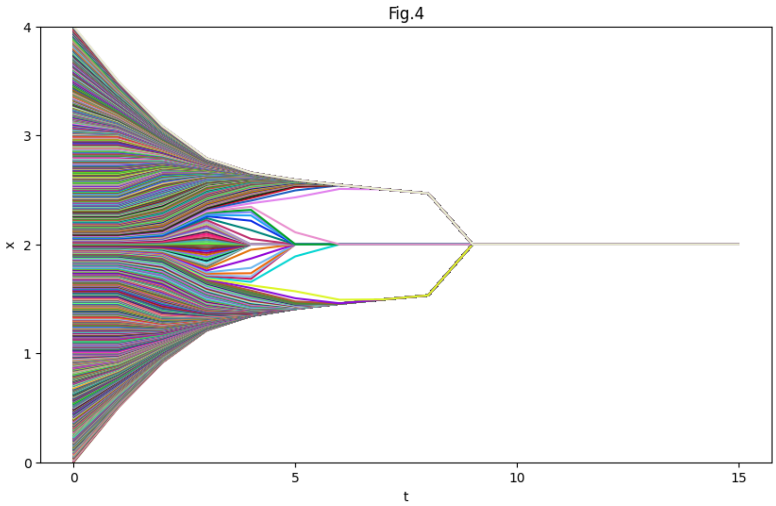

Fig.5

描述为：一个收敛到稳定平衡状态的例子，其中集群间的距离小于 2。初始的意见分布是通过在区间[0, 2.5]上取 251 个均匀间隔的意见，以及在区间[2.5, 3]上取 500 个均匀间隔的意见获得的。意见最终收敛到两个集群，分别有 153 和 598 个智能体，这两个集群之间相隔的距离是 1.6138，大于 1.2559（等于 1 加上 153/598）。当使用更多的智能体时，如果初始意见以相同的方式分布，即区间[2.5, 3]上的密度是区间[0, 2.5]上密度的十倍，那么会得到类似的结果。

复现代码将初始意见按描述进行分布

```
opinions = np.concatenate([np.linspace(0, 2.5, 251), np.linspace(2.5, 3, 500)])
```

复现图如下：