雪堆博弈与极小节点覆盖

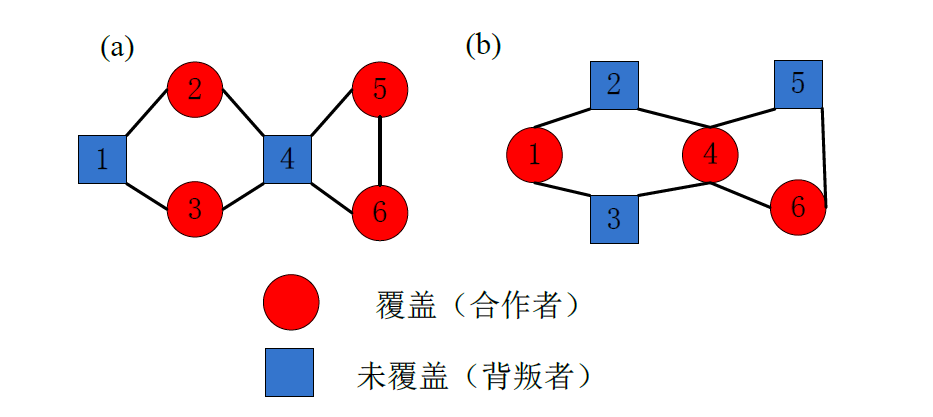
1. 问题重述

当雪堆博弈满足时，网络博弈的纳什均衡中的采用合作策略的节点构成极小节点覆盖。自己编程序验证这个结论，网络自定，节点数目不少于20。

1. 基本概念介绍

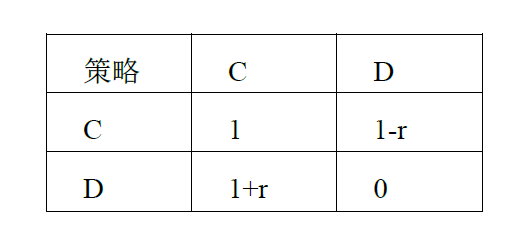
（1）极小节点覆盖

无向图G = (V, E)，其中节点集合V = {1, 2, … , N}，边集合E = {} 。假设是V 的一个子集使得对任意边∈ E，有i ∈ 或 j ∈ ， 也就是，边 至少有一端i 或j 为中的元素，则是网络的一个极小节点覆盖结果（）。 最小节点覆盖（MVC），表示为，是节点数最少的覆盖结果。下图为一个简单的说明网络，其中图(a)为网络的极小节点覆盖，而图(b)为网络的最小节点覆盖。



（2）雪堆博弈

博弈描述的是有利益冲突时相互作用的决策情况。如前所述，雪堆博弈可以进行个体进化。在雪堆博弈中，每个个体可以从两个策略中选择自己所采用的策略：合作C 或者背叛D。假设有两个司机分别被困在一个巨大的雪堆的一边，他们可以选择下车铲雪（合作C），或者呆在车里什么也不做（背叛D）。铲完雪回家将获得收益b，而铲雪堆将消耗劳动成本c。因此，如果两个司机合作，他们每人将收益b-c/2；两人都选择背叛的话将一直呆在车里而什么也得不到；如果一个人铲雪而另一个人呆在汽车里，合作者将获得收益b-c，背叛者收益b。一般地，约b-c/2 为1，则雪堆博弈可以用一个参数描述，耗费收益比r=c/(2b-c)，则雪堆博弈的收益矩阵为

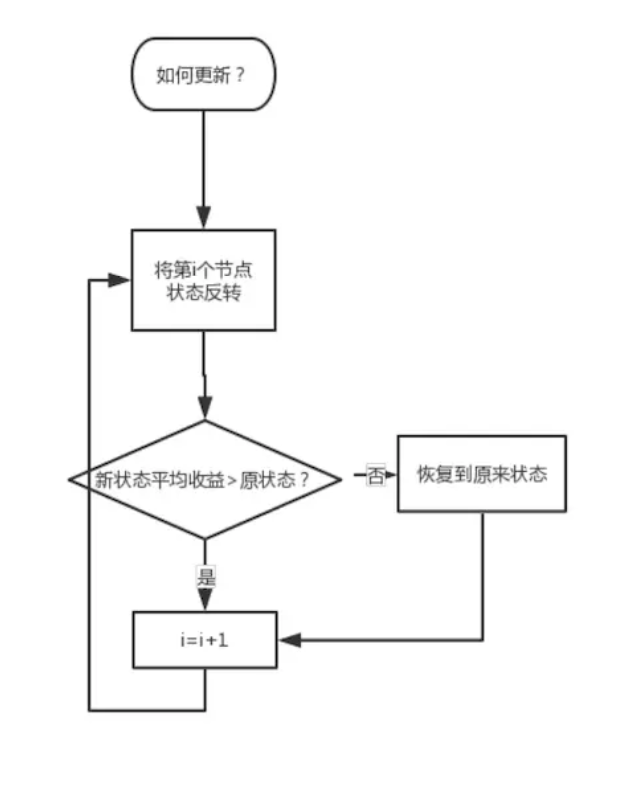
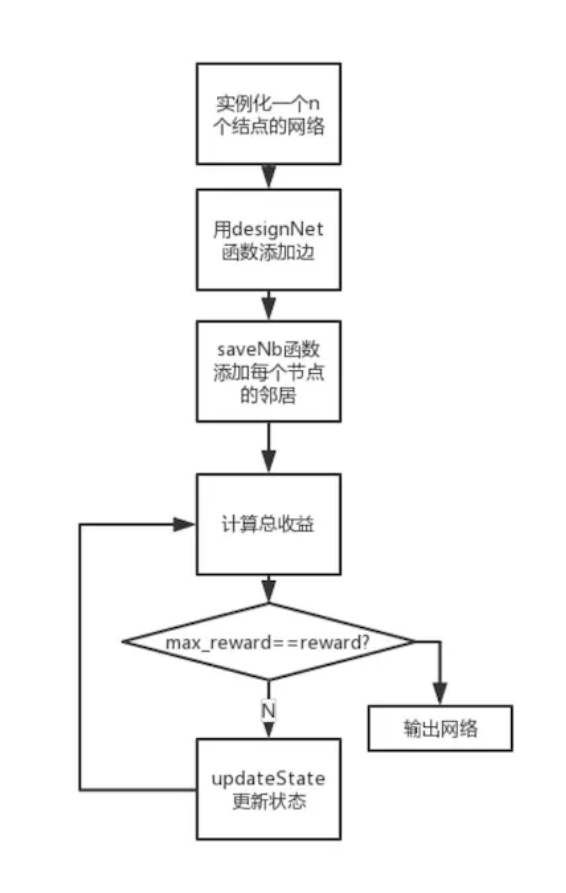


（3）问题背景

在复杂网络的囚徒困境和雪堆博弈中，初始时候每个个体（节点）以相同的概率1/2选择合作或者背叛策略。每轮博弈中，每个个体同时和周围的直接邻居进行博弈获取收益。如果两个人都采取合作策略，则两个人都得到R的收益。如果两个人都采取背叛策略，则两个人都得到P的收益。如果一个人选择合作，一个人选择背叛，那么合作者得到收益S，背叛者得到收益T。

1. 编程实现

（1）程序设计流程与思想



实现过程：

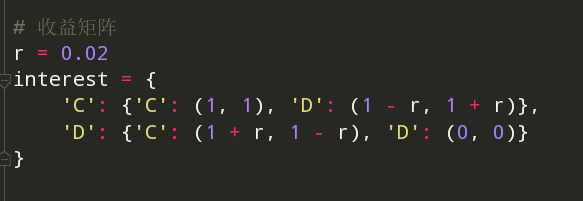
①给每个节点初始状态设置为C或者D

②依次对每个节点计算其采用C或者D的收益，改变其策略使其受益最大。

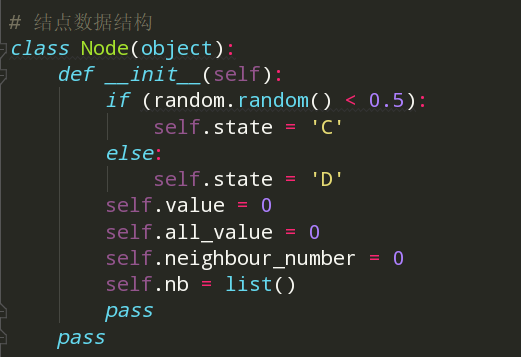
③重复②的过程到每个节点的状态不再改变。

（2）Python语言实现

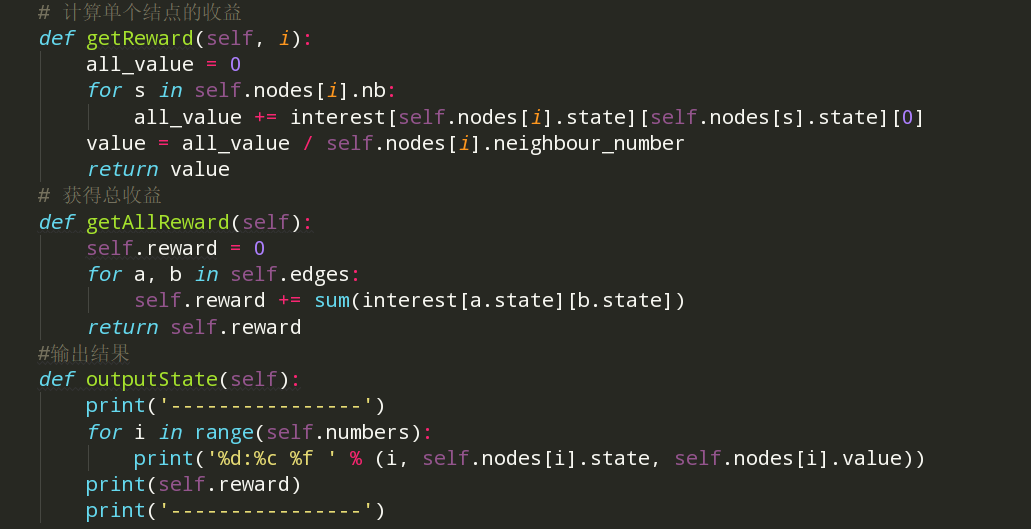
1.定义收益矩阵（取r=0.02）



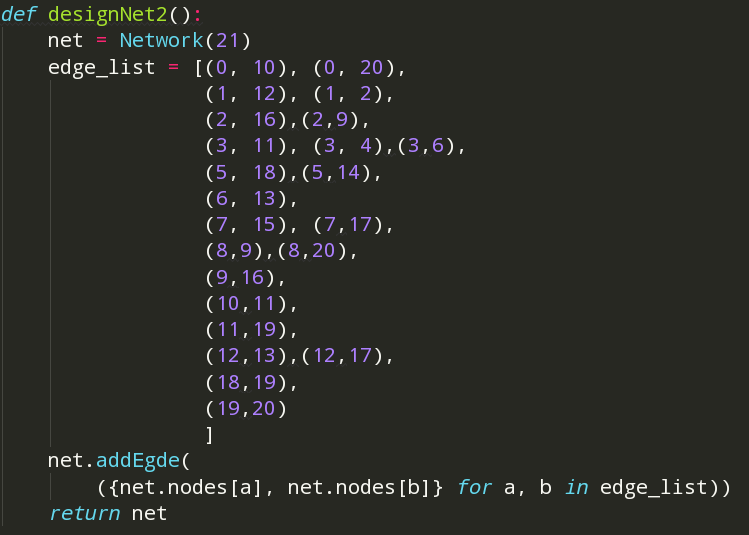
2.节点数据结构



3.计算策略收益并输出结果

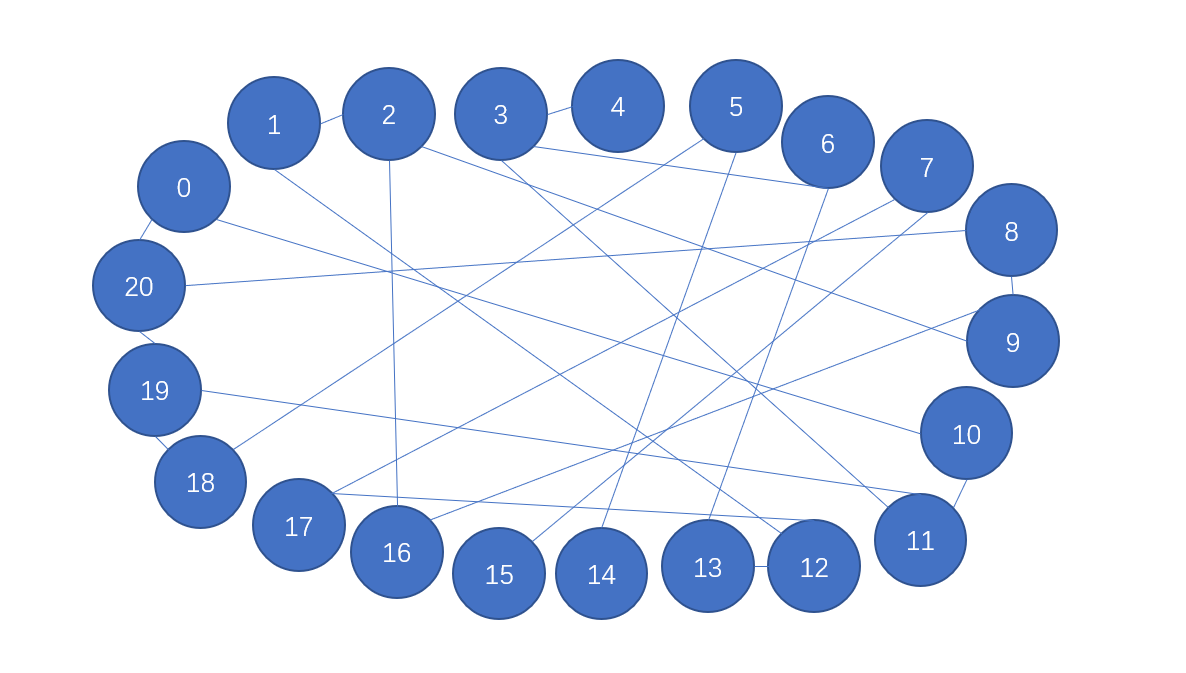


4.定义网络结构



1. 结果展示

（1）首先画出节点图



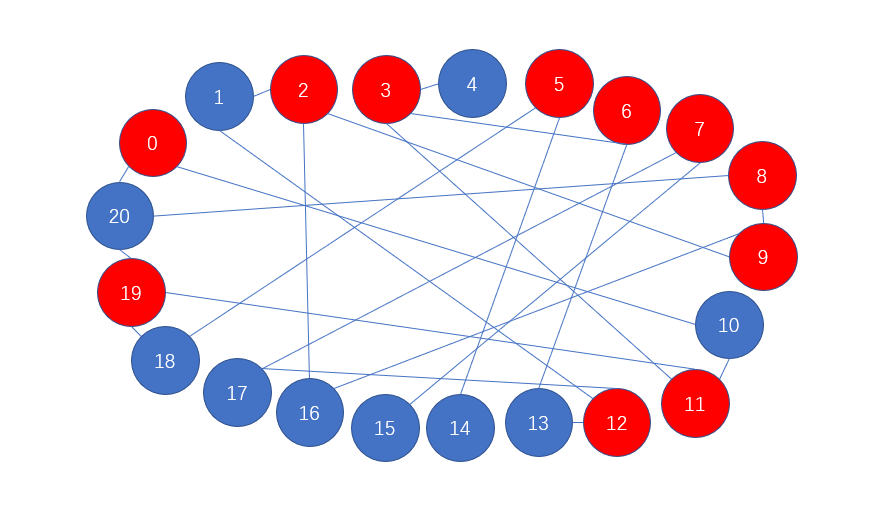
（2）运行算法得到结果

①下面结果表示每个节点采取的策略，C表示合作，D表示背叛，以下为一次运行时算法计算出的每个节点采取的策略。

0:C ，1:D ，2:C ，3:C ，4:D ，5:C ，6:C ，7:C ，8:C ，9:C ，10:D

11:C ，12:C ，13:D ，14:D ，15:D ，16:D ，17:D ，18:D ，19:C ，20:D

对应的图片如下图所示，红点表示合作，蓝点表示对立：



②再次运行算法，查看第二种方法。

0:C ，1:D ，2:C ，3:C ，4:D ，5:D ，6:C ，7:C ，8:C ，9:D，10:D

11:C ，12:C ，13:D ，14:C ，15:D ，16:C ，17:D ，18:C ，19:D ，20:C

对应的图片如下图所示，红点表示合作，蓝点表示对立：

